UNIVERSITE PARIS DAUPHINE

Université Paris-Dauphine

eddimo
ecole doctorale décision informatique mathématiques organisation

No. attribué par la bibliothèque

Les communications multi-parties et leur régulation dans les systèmes multi-agents: modèle et support

THÈSE

Pour l'obtention du titre de

Docteur en Informatique de l'Université Paris-Dauphine

(arrêté du 7 août 2006)

présentée et soutenue publiquement par

Julien Saunier

Composition du jury

Rapporteurs : Olivier Boissier
Professeur à l'ENS Mines Saint-Etienne
René Mandiau
Professeur à l'université de Valenciennes

Examinateurs : Yves Demazeau
Directeur de Recherche CNRS au LIG, Grenoble
Edwin Diday
Professeur à l'université Paris-Dauphine
Flavien Balbo
Maître de conférences à l'université Paris-Dauphine

Directeur de thèse : Suzanne Pinson
Professeur à l'université Paris-Dauphine

Laboratoire d'Analyse et de Modélisation de Systèmes pour l'Aide à la DEcision – UMR 7024

L'université n'entend donner aucune approbation ni improbation aux opinions émises dans les thèses : ces opinions doivent être considérées comme propres à leurs auteurs.

Remerciements

Je tiens tout d'abord à remercier mes encadrants, Suzanne Pinson et Flavien Balbo. Flavien a su à la fois m'encourager dans les voies que je choisissais, et critiquer, aiguiller quand il y en a eu besoin. Suzanne, qui a accepté de diriger ma thèse, et dont les nombreux conseils ont permis d'exposer au mieux les travaux de cette thèse, que ce soit dans les articles ou dans le document présent.

Ensuite, je souhaite remercier les membres du jury. En particulier, Olivier Boissier et René Mandiau ont suivi mes travaux, depuis le travail inachevé de la pré-soutenance jusqu'à accepter la charge de rapporteurs pour la version finale. Je les remercie pour leurs nombreuses remarques sur les travaux réalisés, mais aussi les discussions et nombreuses perspectives qu'ils m'ont apporté. Je remercie également Yves Demazeau pour avoir accepté la présidence du jury et pour l'intérêt qu'il a porté à ma thèse. Enfin, je tiens à remercier Edwin Diday pour sa disponibilité pour les discussions scientifiques autour de l'ADS et pour sa participation (malheureusement à distance) au jury.

La thèse est une expérience particulière, dont on sort plus qualifiés, plus fatigués, et plus dépendants à la caféine. La bonne ambiance du laboratoire est une aide précieuse pour la stabilité du moral, à défaut de toujours soutenir la productivité. A ce titre, je remercie donc tout ceux qui, permanents ou intermittents de la recherche, ont partagé des moments tout au long de ces quatre années, que ces moments aient été scientifiques ou amicaux. Je n'égrènerais pas de liste, de peur de manquer à l'exhaustivité de l'exercice, mais je n'en pense pas moins.

Last, but not least, je remercie mes parents et ma grand-mère, qui ne me prévoyaient probablement pas docteur mais m'ont tout de même soutenu jusqu'à cet aboutissement (qui n'est qu'un début). Je remercie mon frère, qui avait déblayé le chemin avant moi (même si cela ne l'a pas rendu plus simple), pour ses conseils et relectures. Je remercie mes amis, Cédric, Matthieu, pour avoir été là.

Et je ne remercierais jamais assez Cécile, ma femme !

à Cécile

La science ne consiste pas seulement à savoir ce qu'on doit ou peut faire, mais aussi à savoir ce qu'on pourrait faire quand bien même on ne doit pas le faire.
– Umberto Eco, Le nom de la rose.

Table des matières

Chapitre 5
Faciliter et réguler l'interaction : l'environnement comme régulateur de l'interaction

Chapitre 6
Déploiement de l'environnement d'interaction

Table des figures

Introduction

1 Cadre et motivation de la thèse

A l'origine, l'*Intelligence Artificielle* (IA) a pour objectif d'étudier les mécanismes de l'intelligence, et/ou de construire des systèmes capables de répliquer des activités intelligentes. Dans le cadre des *agents intelligents*, il s'agit de la capacité de l'agent à poursuivre ses propres buts et à exécuter les tâches nécessaires à leur réalisation, par un comportement rationnel et flexible dans son environnement.

Pris au sens fort du terme, le but de l'IA est de créer un programme qui, dans un monde arbitraire, ne ferait pas plus mal qu'un être humain. Au sens faible, il s'agit de trouver des méthodes d'ingénierie inspirées de comportements intelligents, ou les mimant. Pour cela, le chercheur s'inspire souvent d'observations du monde réel.

Les systèmes multi-agents mettent en oeuvre une distribution de la connaissance et des compétences au sein d'agents autonomes qui interagissent entre eux. Nombre de travaux dans cette communauté sont naturellement inspirés par les interactions animales et humaines, tout en conservant les contraintes liées au support artificiel.

1.1 Les systèmes multi-agents

Une bonne vue d'ensemble du domaine des systèmes multi-agents peut être trouvée dans [Weiss, 1999, Chap 1,2]. Dans ce livre, l'auteur refuse d'apporter une définition précise des agents, et plus globalement des systèmes multi-agents. Son argument principal en faveur des systèmes multi-agents est qu'ils sont le meilleur paradigme des systèmes informatiques distribués : de nos jours, de nombreux composants sont inter-connectés, allant de l'ordinateur au serveur en passant par le micro-onde ou le réfrigérateur. Chacun peut être considéré comme un agent faisant partie de l'environnement des autres.

Une devise du monde multi-agents est :

> *Il n'existe rien de tel qu'un système mono-agent*[1].
>
> [Wooldridge, 2002]

En effet, dans la majorité des systèmes informatiques non triviaux se trouvent déjà des sous-systèmes qui doivent interagir sans cesse afin de mener à bien leur tâche. Les systèmes

1. There is no such thing as a single agent system.

multi-agents se situent à un niveau d'abstraction supérieur, puisque les tâches à réaliser sont partagées entre des agents autonomes et non des modules logiciels.

Wooldridge donne la définition suivante d'un agent :

> Un *agent* est un système informatique qui est *situé* dans un *environnement*, et qui est capable d'*agir de façon autonome* dans cet environnement de façon à atteindre les objectifs pour lesquels il a été conçu [2].
>
> [Wooldridge, 2002]

Un agent est une entité informatique non nécessairement matérialisée, qui peut aussi bien être un programme logiciel qu'un robot autonome. Il est indissociable de son environnement, duquel il reçoit des informations via des capteurs, et dans lequel il effectue des actions grâce à des effecteurs.

Bien que cette dichotomie tende à s'estomper depuis quelques années, la littérature a longtemps été divisée entre deux grands types d'agents, les agents réactifs et les agents cognitifs. Les agents réactifs (voir par exemple [Ferber et Drogoul, 1992]) proviennent de l'*éthologie*, science du comportement animal. Ces agents opèrent selon un cycle perception/action, qui est l'équivalent d'un arc réflexe stimulus/réaction. Du point de vue de l'implémentation, les agents sont simples et réagissent de façon pré-progammée à chaque situation. Les agents purement réactifs ne disposent pas de mémoire des évènements passés, ne possèdent pas de représentation symbolique de leur environnement et n'utilisent pas de communications langagières de haut niveau. De fait, l'interaction proviendra souvent d'altérations de l'environnement, lesquelles seront perçues par la suite par les autres agents. Il s'agit du phénomène de *stigmergie*. L'exemple type de cette famille de systèmes multi-agents est la simulation d'une fourmilière [Drogoul et al., 1995], dans laquelle chaque fourmi est représentée par un agent. Les fourmis sont situées, perçoivent des phéromones et réagissent en fonction de leurs perceptions. La conjonction de ces comportements simples a montré l'émergence d'un corps social complexe permettant à la société d'exhiber une coordination efficace.

A contrario des agents réactifs, les agents cognitifs possèdent une architecture complexe, qui sera caractérisée par une représentation symbolique de leur environnement, une mémoire, et une capacité de décision. L'usage des symboles permet l'utilisation de langages de haut niveau disposant d'une sémantique. La mémoire et la capacité de décision permettent d'ajouter à leur cycle principal une phase de délibération (Perception/Délibération/Action), lors de laquelle l'agent décide en fonction de ses perceptions, de sa mémoire et de son environnement l'action qu'il va effectuer. Les connaissances et buts de l'agent influent ainsi sur son comportement.

Il est à noter que, factuellement, la plupart des agents actuels sont hybrides, c'est à dire possèdent des caractéristiques en provenance du monde réactif et d'autres du monde cognitif.

A l'origine, les systèmes multi-agents sont des systèmes coopératifs, l'objectif étant de diminuer la complexité de chacun des individus par la répartition des connaissances et des tâches de

2. An *agent* is a computer system that is *situated* in some *environment*, and that is capable of *autonomous action* in this environment in order to meet its design objectives

raisonnement. Les difficultés sont alors de permettre une organisation efficace de ces informations et capacités de raisonnement pour résoudre le problème donné, ainsi que la façon de les coordonner. Chaque agent étant autonome, cette coordination ne peut se faire que par l'interaction et l'échange efficace des informations.

Le paradigme multi-agents a évolué vers des systèmes où les agents peuvent être égoïstes et en compétition. Mais, même dans ce cas, si les agents sont mis en présence et dotés de capacités d'interaction, c'est toujours parce qu'il y a une nécessité de partage de l'information et des besoins mutuels.

1.2 L'interaction, à la croisée des chemins

Du point de vue du système multi-agents, nous parlerons de partage des informations, tandis qu'au niveau de l'agent la problématique est celle du besoin en information. Ceci donne les deux voies prises pour résoudre le problème : soit organiser un espace commun de l'information, dont les précurseurs sont les systèmes à base de tableaux noirs [Engelmore et Morgan, 1988], soit intégrer les informations au sein des agents pour les échanger quand il y en a besoin. Nous verrons par la suite que cela induit des problématiques différentes. Si le problème est toujours de retrouver l'information efficacement, dans le premier cas la difficulté est de discriminer directement les informations intéressantes, tandis que dans le deuxième il s'agit de trouver qui possède les informations ou compétences recherchées.

Par ailleurs, deux contraintes s'ajoutent. La première est l'infrastructure réelle dans laquelle les agents évoluent, qui implique des modes de communications différents suivant la fiabilité des communications. Par exemple, une faible bande passante et une connectivité intermittente peuvent rendre difficile l'utilisation de communications synchrones. La seconde est le support informatique lui-même, les réseaux étant fondés sur un ensemble pré-existant de protocoles et d'architectures.

Ces contraintes liées au support donnent un cadre aux interactions entre agents. Dans le monde réel, deux personnes en discussion peuvent offrir plusieurs niveaux d'information aux tierces personnes, allant du simple fait de savoir qu'elles échangent des messages au contenu de cet échange. L'adressage systématique des communications entre agents limite ainsi la prise en compte des principes des communications humains.

L'utilisation par les agents de langages de haut niveau et de modèles d'interaction complexes, issus des travaux dans les domaines de l'éthologie, de la cognition et des sciences sociales est fortement contraint par la réalité de leur support. Dans ce cadre, nos travaux ont pour objectif d'améliorer le support des communications de façon à mettre en oeuvre toute la richesse des interactions.

1.3 Cadre des travaux

La place de l'environnement dans les systèmes multi-agents a récemment bénéficié d'une forte dynamique internationale, illustrée notamment par un groupe de travail AgentLink [3] et par un workshop lié à la conférence AAMAS [4]. Ces évènements, auxquels nous avons participé, ont permis de fédérer les travaux de la communauté (voir par exemple [JAAMAS, 2007]).

Au sein de notre équipe, le concept de l'environnement comme support actif de la communication a été introduit dans [Balbo, 1999, 2000]. Ces premiers travaux reposent sur un système d'adressage des communications par des filtres dans le cadre d'un système d'aide à la décision. Ce modèle propose de déléguer à l'environnement la tâche de choix des récepteurs et de distribution des messages.

Par la suite, plusieurs axes de recherche autour de ce thème ont débuté au sein de l'équipe Agents Intelligents et Modèles Coopératifs du LAMSADE.

Nous avons tout d'abord unifié les principes régissant la flexibilité de la communication sous le nom de coordination fondée sur les propriétés (*Property-Based Coordination*), et trois thèmes (Fig. 1) sont explorés :

- La simulation
- Le modèle de coordination
- Le modèle d'interaction

Le premier thème a pour objectif d'intégrer au cadre de la simulation la *Property-Based Coordination*, en permettant à l'environnement non seulement de transmettre des messages, mais également de déclencher des comportements des agents en fonction de leur contexte.

Le second thème aborde la question du modèle formel de coordination entre processus, selon une approche de type espaces partagés. L'accent est alors porté sur la dynamique du modèle, par l'intégration de l'algèbre de processus.

Enfin, le troisième thème est celui sur lequel porte cette thèse, dont nous présentons maintenant les principales contributions.

2 Contribution

Notre approche s'inspire de travaux dans les domaines de la sociologie, de la psychologie et du travail coopératif. Ceux-ci étudient les principes sous-jacents de la communication humaine sous le terme de *communications multi-parties*, plus riches que les communications adressées classiques grâce à la prise en compte des besoins des récepteurs et du contexte de transmission. Nous nous appuyons sur ces principes pour proposer à la fois un modèle et une infrastructure adaptés aux système multi-agents. Ce travail se situe donc au niveau multi-agents, et non agent.

La principale contribution de cette thèse est ainsi d'envisager les communications d'un point de vue global, sans distinguer *a priori* interactions directes et indirectes. Ceci nous permet d'offrir

3. `http://www.cs.kuleuven.be/~distrinet/events/e4mas/`
4. `http://www.aamas-conference.org/`

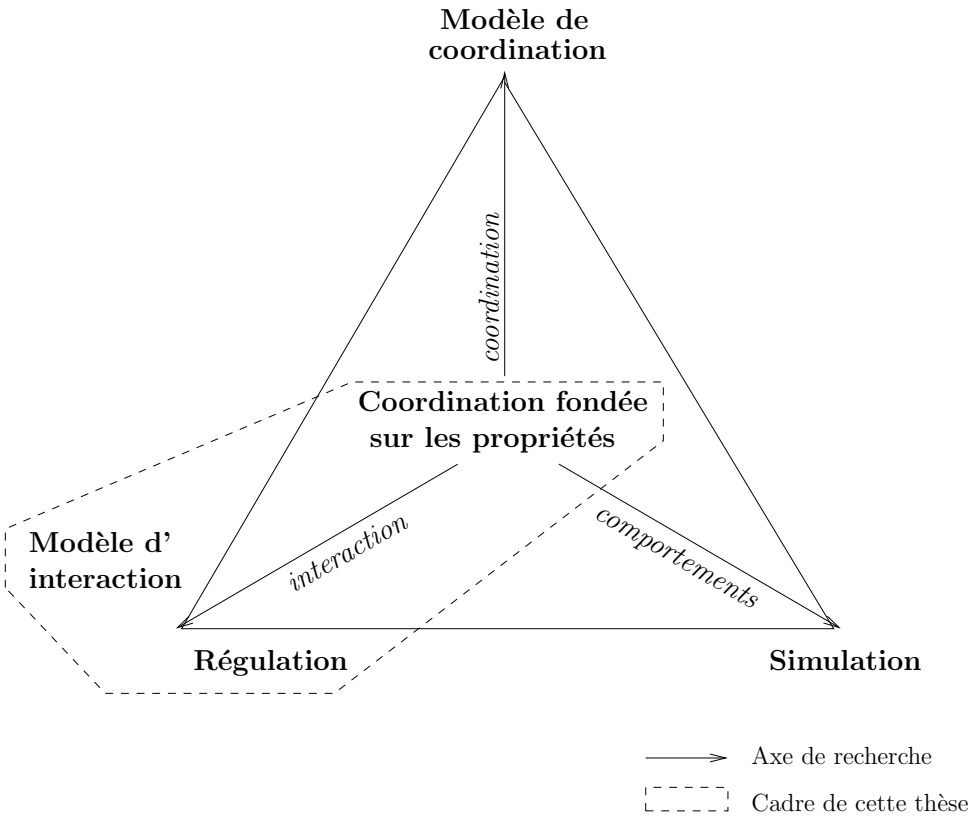


FIGURE 1 – Positionnement au sein des travaux de l'équipe

une solution standardisée pour le support et le contrôle des différents types de communication au sein d'un même système multi-agents, dans le but d'améliorer la propagation des informations et de favoriser les comportements proactifs.

Un modèle d'environnement pour faciliter les interactions

Nous introduisons le principe de *coordination fondée sur les propriétés*, selon lequel les agents ont des besoins en interaction, qui s'expriment sous forme de conditions sur l'état du système multi-agents. Notre modélisation est fondée sur un environnement explicite, qui est traduite par une infrastructure pour le support de l'interaction.

Nous avons choisi de formaliser le modèle d'environnement EASI - *Environnement Actif Comme Support de l'Interaction* - à l'aide du paradigme de l'analyse de données symboliques, afin d'unifier l'expression de l'état du système et des besoins des agents. Les entités du système sont décrites par des propriétés. Les besoins en interaction des agents -ses interlocuteurs ou les informations qu'ils recherchent- sont décrits par des conditions sur ces propriétés. Ils sont gérés dynamiquement par l'environnement sous forme de filtres. Les filtres permettent de prendre en compte des informations sur les agents, sur les messages et toutes autres informations contextuelles.

L'utilisation des filtres permet un adressage logique des données, qui est plus flexible que l'adressage classique point-à-point. Elle permet en particulier les communications de groupe, qui concernent un ensemble d'individus décrits de façon logique. Les filtres peuvent être déposés par l'agent émetteur d'une donnée ou par n'importe quel autre agent du système. Ceci permet de prendre en compte les besoins de l'ensemble des agents de façon unique et standardisée, par une conjonction des principes des communications directes et indirectes. Les filtres peuvent être déposés par l'environnement, et ainsi fournir un comportement standard des communications au niveau multi-agents.

Nous proposons des algorithmes pour gérer le processus dynamique de recherche des récepteurs. Ces algorithmes sont fondés sur la modélisation des propriétés. Ils s'appuient sur la conservation et la mise à jour de données sur la structure du système multi-agents.

Réguler et appliquer des politiques d'interaction

Dans les systèmes hétérogènes et ouverts, le concepteur du système multi-agents n'a pas le contrôle de la conception de tous les agents. Ceux-ci peuvent donc se révéler malveillants. Dans ce cadre, il est dangereux de laisser aux agents une totale liberté de manipulation des filtres.

Nous proposons une extension du modèle EASI pour la régulation du système multi-agents, grâce à l'adjonction de normes et lois de réception, et à la hiérarchisation des besoins et permissions. Ce nouveau modèle est appelé EARI, l'*Environnement Actif comme Régulateur de l'Interaction*. La hiérarchisation sous forme de politiques de priorité permet d'assurer différents types de comportement de l'infrastructure. Notamment, les règles de transmission des messages peuvent être obligatoires, ou il peut être permis de les transgresser. Il est aussi possible de mettre en oeuvre des priorités différenciées suivant les agents et la nature des filtres.

Déploiement de l'environnement de communication

Dans cette thèse, nous proposons une architecture fonctionnelle du modèle d'environnement de communication, de façon à faciliter la mise en place de systèmes multi-agents supportant les communications multi-parties. Chaque module est ensuite détaillé, ainsi que la façon dont cet environnement de communication s'intègre à une architecture d'environnement complète.

Nous avons réalisé deux implémentations du modèle. La première est un prototype fonctionnant sur une architecture client/serveur. La seconde a été intégrée à la plate-forme MadKit de façon à enrichir les interactions du modèle Agent-Groupe-Rôle par notre modèle d'environnement tout en conservant ses services initiaux (gestion du cycle de vie des agents, structure organisationnelle).

Validation

Nous avons validé notre modèle de plusieurs façons. La formalisation adoptée pour le choix des récepteurs est exprimable sous forme de logique des prédicats, nous avons donc réalisé

une première série de tests pour déterminer l'efficacité d'un système expert pour la gestion de l'environnement.

Nous avons ensuite validé les deux algorithmes fondés sur la structure du système multi-agents. Ils ont été implémentés sur un support *ad-hoc* et testés pour évaluer leur efficacité. Nous avons aussi vérifié l'impact du contrôle des communications sur le temps d'exécution du système.

Eléments d'exploitation des modèles

L'approche globale des interactions par une infrastructure dédiée permet la mise en oeuvre effective de modèles de communication riches, tels que l'écoute flottante, tout en offrant un support aux modèles classiques. Etant donné que le système flexible d'adressage que nous proposons a un impact sur la notion de protocole, nous étendons les protocoles existants par l'insertion de comportements proactifs opportunistes. Nous montrons ensuite comment instancier au sein de l'environnement un système multi-agents normé.

3 Organisation de la thèse

Ce document est organisé en 7 chapitres. Le premier chapitre forme une introduction à la problématique des communications multi-parties et identifie les concepts nécessaires à leur mise en oeuvre. Le second chapitre approfondit l'état de l'art en étudiant les supports de communication, et en les mettant en perspective avec les concepts issus du premier chapitre. Le troisième chapitre introduit une vue d'ensemble de notre modèle d'interaction, qui a pour but la facilitation des interactions. Le quatrième chapitre présente la formalisation de ce modèle et les algorithmes de gestion de l'environnement. Le cinquième chapitre étend le modèle au processus de régulation. Le sixième chapitre décrit une architecture fonctionnelle pour le modèle, ainsi que les deux implémentations réalisées. Enfin, le septième chapitre aborde la validation du modèle et son exploitation.

Chapitre 1 - Les communications multi-parties

Afin de placer cette thèse dans son contexte, le chapitre 1 propose un état de l'art sur les modèles de communication de groupe, et situe le cadre de leur support. Après une mise en perspective des notions de communication et d'interaction, nous introduisons un type particulier de communication, à savoir l'écoute flottante, et les difficultés inhérentes à son application. Ensuite, nous voyons comment la notion de communications multi-parties est un cadre général aux interactions du monde réel. Nous faisons ressortir les concepts nécessaires à l'unification des différents types de communication dans les systèmes multi-agents.

Enfin, nous mettons en exergue la corrélation de l'écoute flottante avec la prise en compte du contexte dans les interactions, ainsi que la nécessité d'introduire une part de contrôle à la fois au niveau agent et au niveau multi-agents.

Cette étude nous permet de définir quatre critères pour le support des communications multi-parties : la prise en compte des besoins de l'émetteur, la prise en compte des besoins des récepteurs potentiels, l'utilisation du contexte et la possibilité de contrôle.

Chapitre 2 - Les supports de communication : un état de l'art

Dans ce second chapitre, nous présentons un état de l'art des supports d'interaction dans les systèmes multi-agents, à l'aune des critères définis lors du premier chapitre. Nous étudions d'abord les interactions directes, qui sont actuellement le modèle majoritaire pour les systèmes multi-agents, et nous nous intéressons particulièrement au problème de la mise en relation des agents.

Ensuite, nous proposons un panorama des infrastructures de médiation. En particulier, nous étudions les plate-formes multi-agents, les systèmes *Publish and Subscribe*, les espaces partagés, et l'environnement. Nous montrons que la médiation est nécessaire au support des communications multi-parties, mais que les modèles actuels sont insuffisants pour ce support.

Chapitre 3 - Présentation générale du modèle d'environnement comme support actif de l'interaction

Dans le troisième chapitre, nous proposons un principe général de mise en oeuvre des communications multi-parties appelé *coordination fondée sur les propriétés*. Nous présentons ensuite une vue générale d'un système multi-agents utilisant les communications multi-parties, sur un exemple de cité digitale. Nous décrivons ainsi le fonctionnement effectif de notre modèle d'environnement.

Chapitre 4 - Formalisation du modèle d'environnement actif comme support de l'interaction

Le quatrième chapitre est une présentation du modèle EASI, "Environnement Actif comme Support de l'Interaction". L'environnement est utilisé comme médiateur de l'interaction, par la réalisation de la connexion avec les récepteurs. Notre formalisation s'appuie sur celle de l'analyse de données symboliques. Nous l'avons choisie d'une part pour son expressivité, et d'autre part pour l'efficacité de la recherche des récepteurs.

La modélisation des besoins des agents par des règles permet d'obtenir un modèle d'interaction prenant en compte les communications directes et indirectes.

Chapitre 5 - Faciliter et réguler l'interaction : l'environnement comme régulateur de l'interaction

Dans un contexte ouvert et hétérogène, il est nécessaire de faciliter les interactions, mais aussi de les contrôler. Le chapitre 5 étend le modèle décrit précédemment en ajoutant des règles de contrôle. Ce nouveau modèle est appelé EARI, "Environnement Actif comme Régulateur de

l'Interaction". Nous proposons la mise en place de priorités pour réguler et contrôler l'interaction. Nous montrons l'influence du choix des priorités pour obtenir différents comportements du système, que nous appelons politiques.

Chapitre 6 - Déploiement de l'environnement d'interaction

Le sixième chapitre introduit une architecture conceptuelle pour le modèle EASI, et décrit les modules fonctionnels nécessaires à sa mise en oeuvre.

Ensuite, nous introduisons deux implémentations du support d'interaction. La première est un prototype d'environnement de communication doté d'une architecture client/serveur que nous avons développé en Java. La seconde utilise la plate-forme MadKit, pour laquelle nous avons développé une interface fournissant les primitives d'utilisation de l'environnement.

Chapitre 7 - Expériences et éléments d'exploitation des modèles

Dans le septième chapitre, nous étudions expérimentalement deux implémentations d'EASI : la première est une série d'expériences sur un exemple jouet pour tester le processus de recherche des récepteurs à l'aide d'un système expert. La seconde est une série d'expériences sur nos algorithmes à l'aide d'un exemple issu de l'intelligence ambiante. Nous discutons les performances de ces deux implémentations.

Ensuite, nous abordons la question de l'ingénierie des protocoles d'interactions dans ce nouveau cadre, pour la mise en oeuvre de protocoles opportunistes. Enfin, nous montrons comment instancier au sein de l'environnement un modèle normé fondé sur la logique déontique.

Chapitre 1

Les communications multi-parties

Sommaire

– Pour un agent, interagir avec un autre constitue à la fois la source de sa puissance et l'origine de ses problèmes.

[Ferber, 1995, chap 1.5]

Un Système Multi-Agents (SMA) se caractérise tout d'abord par les interactions ayant lieu en son sein. Un agent interagit avec son environnement, ainsi qu'avec d'autres agents [5]. Comme le souligne Ferber, les interactions sont une source d'opportunités autant que de contraintes.

5. Les autres agents sont parfois considérés comme faisant partie de l'environnement de l'agent

1.1 Communication et interaction

Dans cette section, nous introduisons les notions de communication et d'interaction. Nous décomposons ensuite les différents "niveaux" intervenants dans l'interaction, de façon à déterminer le champs de notre étude. Enfin, nous donnons une première approche des deux principaux types d'interaction, directe et indirecte.

1.1.1 Première approche

La définition du terme *communication*, tout comme celle du terme *interaction*, n'est pas la même pour les communautés multi-agents et informatique distribuée. La communauté des Systèmes Multi-Agents (SMA) tend à s'intéresser aux concepts et à la finalité de haut niveau de l'interaction, tandis que la communauté informatique distribuée s'approche plus des problématiques réseau. Une première approche multi-agents de la communication est celle de Ferber [Ferber, 1995] :

> **Communication (1)** – Les communications, dans les systèmes multi-agents comme chez les humains, sont à la base des interactions et de l'organisation sociale. Sans communication, l'agent n'est qu'un individu isolé, sourd et muet aux autres agents, renfermé sur sa boucle perception-délibération-action. C'est parce que les agents communiquent qu'ils peuvent coopérer, coordonner leurs actions, réaliser des tâches en commun et devenir ainsi de véritables êtres sociaux.

Dans cette approche, la communication est définie en fonction de ce qu'elle permet. L'objectif de la communication est la coopération et la coordination des agents sociaux. Plus généralement, il s'agit d'un échange d'information, lequel peut être l'objectif final de l'opération de communication ou un moyen d'atteindre un effet de coordination. Il est alors nécessaire de voir comment deux agents peuvent communiquer ensemble.

La base de la théorie de la communication provient de [Shannon, 1948], telle qu'illustrée en figure 1.1. Elle correspond au niveau le plus bas, le réseau : la source encode un message, puis le transmet par un signal à travers un canal qui peut être bruité. Le signal est décodé par le récepteur à destination.

Pour des entités du monde réel, une définition fonctionnelle de la communication est la suivante :

> **Communication (2)** – La communication est l'échange volontaire d'informations provoqué par la production et la perception de symboles tirés d'un système partagé de symboles conventionnels [6]
>
> [Russell et Norvig, 2003]

6. *Communication is the intentional exchange of information brought about by the production and perception of signs drawn from a shared system of conventional signs.*

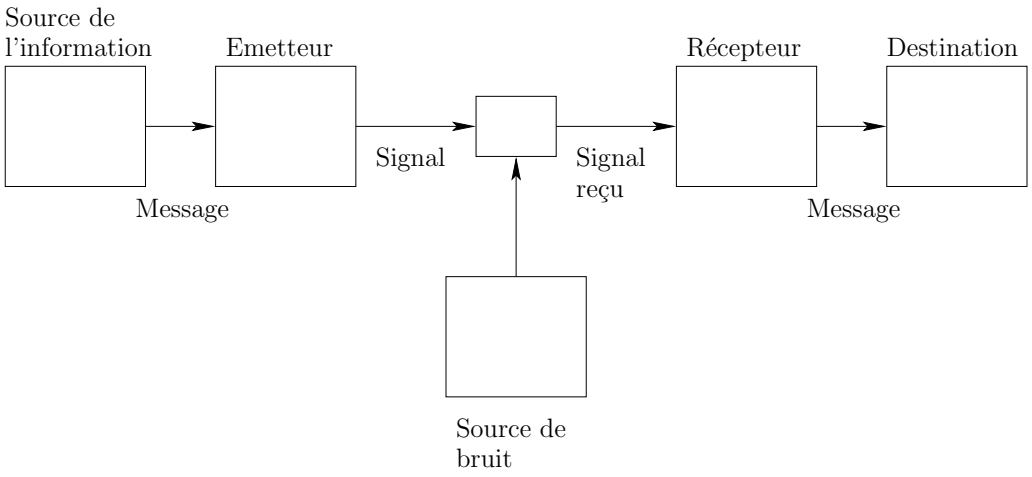


FIGURE 1.1 – Décomposition d'une communication [Shannon, 1948]

Nous retrouvons dans cette définition l'aspect de la production et de la perception de signaux. Nous notons une information supplémentaire, qui est celle de l'utilisation de symboles que doivent partager l'émetteur et le récepteur. Il y a donc une couche supplémentaire à considérer lorsque l'on se situe au niveau des agents, qui est celle du langage qui doit nécessairement être commun. En effet, le signal peut être reçu et décodé sans que l'information ne soit partagée, si les symboles utilisés ne sont pas reconnus par le récepteur. Il s'agit donc d'un double codage, au niveau réseau (les signaux) et au niveau agent (les symboles).

Il est à noter que Russell utilise le terme d'*échange* d'information, alors que le modèle original de Shannon ne donne pas de réciprocité de l'information. Ceci apporte un flou qui explique que les termes "communication" et "interaction" soient parfois utilisés dans la littérature de façon interchangeables. Nous choisissons donc, pour des raisons de clarté, de discriminer les deux sur cet aspect de l'échange. En conséquence, notre définition de l'interaction sera :

> **Interaction** – Les interactions sont des actions réciproques modifiant le comportement ou la nature des éléments, corps, objets, phénomènes en présence ou en influence.
>
> [Morin, 1977]

En ce sens, une communication est souvent une partie d'une interaction. Par exemple, lorsqu'un agent envoie une requête à un autre agent pour que celui-ci exécute une tâche, l'acte de communication engendre une modification des croyances de l'émetteur quant au récepteur, et des croyances du récepteur concernant l'émetteur. L'interaction prend donc en compte les effets de l'acte de communication.

Dans cette thèse, nous étudions les interactions entre agents, et non les interactions mettant en oeuvre un agent et d'autres entités (ressources, objets...). Lorsque nous utilisons le terme interaction, celui-ci implique donc nécessairement une communication. Cette communication

peut être directe ou indirecte (section 1.1.3), dans tous les cas elle implique un émetteur qui fournit une information, et un ou plusieurs récepteurs qui la reçoivent.

Nous avons vu en introduction que les agents réactifs avaient une fonction de perception, laquelle est remplacée ou étendue par la communication pour les agents cognitifs. L'intention de la communication est celle de l'émetteur, auquel cas le récepteur de l'information est passif. A l'inverse, la perception peut être un acte volontaire de la part de celui qui perçoit l'information. Il est à noter qu'en pratique, un certain nombre de plate-formes simulent la perception par des stimuli qui sont reçus de la même façon que le sont les messages (par exemple [Weyns et al., 2005], [Weyns et al., 2007]).

1.1.2 Les composants de l'interaction

Dans cette première approche des notions de communication et d'interaction se mêlent des considérations de bas niveau, comme les bruits sur le canal de transmission, et de plus haut niveau, comme les croyances des agents. Nous proposons donc une décomposition des différents concepts en fonction de leur proximité avec le support informatique lui-même.

Schéma général

D'un point de vue conceptuel, nous pouvons séparer les différents niveaux entrant dans le cadre de l'interaction. Le niveau le plus bas est celui du support physique, par exemple les protocoles réseaux utilisés. Ensuite, il peut y avoir une infrastructure particulière, dans le cas où l'environnement multi-agents est non-vide. Au dessus de cette infrastructure se situent les informations elles-mêmes, à savoir les agents, les messages et les objets de l'environnement. Enfin, le plus haut niveau d'abstraction est celui du modèle d'interaction utilisé. Notons que cette classification des concepts n'est pas directement liée à la notion de couches réseaux, le lien exact entre les deux est abordé en annexe A. La figure 1.2 illustre les différents niveaux, que nous explicitons dans la suite.

1.1.2.a Le support physique

Nous distinguons deux grands types de support physique : les réseaux classiques et les réseaux à communications limitées. Les premiers sont des réseaux de type LAN, Internet, dans lesquels il n'y a pas de limitation de la portée des messages à partir du moment où l'adresse du destinataire est connue.

Les seconds sont souvent liés à la réalité physique, pour des robots, grappes de capteurs, drones, dans le cas où les communications sont situées et/ou ne sont pas fiables. Les communications ne sont pas fiables s'il peut y avoir perte d'information, ou si leur portée est limitée par la topologie de l'environnement.

Le support physique est souvent en partie une hypothèse du domaine d'application du système multi-agents.

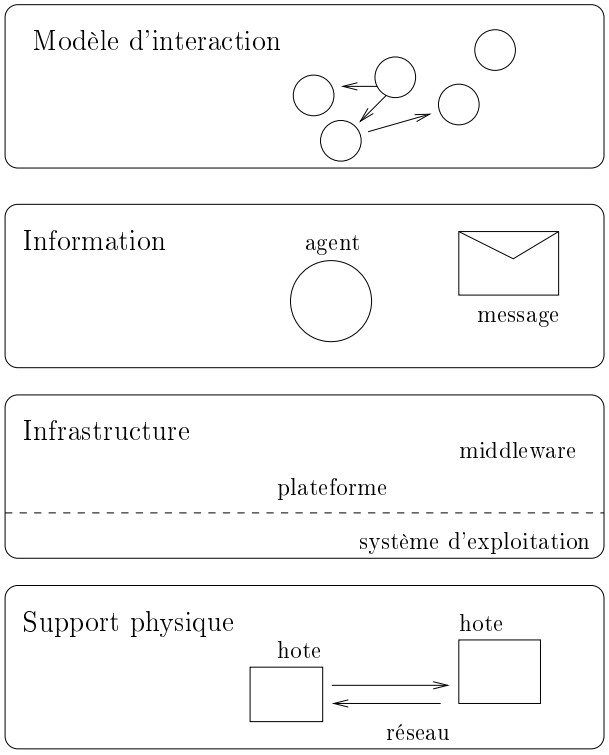


FIGURE 1.2 – Décomposition de l'interaction en niveaux

1.1.2.b L'infrastructure

Le support minimal faisant l'interface entre l'agent et la machine est composé du système d'exploitation et des protocoles du réseau. A cela s'ajoute généralement une plate-forme, qui permet de gérer des instructions de plus haut niveau. Les plate-formes multi-agents supportent en particulier :

- le cycle de vie des agents
- le transport des messages

A ces fonctions de base peuvent s'ajouter des services tels que la gestion des organisations, la gestion de ressources, des annuaires, que nous approfondissons en sections 2.2.1 et 2.2.2.

La gestion des accès à certaines ressources peuvent aussi être déléguées à des middlewares (ou *intergiciels*). Ces middlewares servent d'interface entre les agents et les autres composants de l'environnement informatique.

1.1.2.c L'information

Au niveau de l'information, il y a d'une part l'architecture de l'agent et d'autre part la structuration de l'information. L'agent doit être capable d'accéder à l'information et de la comprendre, ce qui implique une standardisation du format d'échange des messages. Les deux principaux langages de communication entre agents utilisés dans la communauté sont KQML [Finin

et al., 1997] et FIPA-ACL [FIPA, 2002b], et nous pouvons citer comme format d'échange de connaissances KIF [Genesereth et Fikes, 1992] et FIPASL [FIPA, 2000].

1.1.2.d Le modèle d'interaction

Les langages de communication ont une sémantique : un message induit un changement de l'état mental à la fois de l'émetteur et du récepteur. La réception du message provoque une modification de l'état du récepteur, et l'état de l'émetteur est modifié en fonction de cet effet attendu. C'est par la combinaison des communications et actions des agents que l'on obtient une interaction, et c'est par la combinaison de ces interactions que l'on obtient la coordination des agents.

Pour ce faire, les agents utilisent des modèles d'interaction communs, qui leur permettent d'être efficaces pour résoudre leurs tâches. Ce niveau d'abstraction est en général défini au niveau multi-agents. Il peut être matérialisé de différentes façons, par exemple par des protocoles.

Cadre de l'étude

Le choix du concepteur de systèmes multi-agents est transversal : choisir un modèle d'interaction implique une infrastructure de soutien, et inversement l'infrastructure et le support physique impliquent des contraintes. Ainsi, pour pouvoir améliorer les niveaux agent et multi-agents, il est nécessaire d'agir au niveau du support logique, autrement dit l'infrastructure.

Dans la suite de cette thèse, nous nous situons dans le cadre des réseaux classiques filaires, les problématiques identifiées et les solutions qui sont proposées ne sont donc pas *a priori* adaptées aux réseaux sans fils.

1.1.3 Interactions classiques : directes et indirectes

Un des choix fondateurs que doit faire le concepteur de Systèmes Multi-Agents est celui du modèle d'interaction, et à travers cela d'une infrastructure. Dans cette section, nous présentons de façon générale les deux principales familles de modèles d'interaction, les interactions directes et indirectes. Les infrastructures liées aux différents modèles d'interaction seront discutées en détail dans le chapitre 2.

1.1.3.a Interactions directes

La majorité des travaux utilise des *interactions directes*, dans la lignée des méthodes classiques des systèmes informatiques. Les interactions directes sont fondées sur la communication adressée point-à-point : un émetteur envoie un message à un récepteur localisé par son adresse. On retrouve ici de façon simplifiée la théorie de Shannon.

Nous illustrons les communications directes par la figure 1.3. Le média est déterminé au niveau du support physique et de l'infrastructure. Nous ajoutons un nouveau concept, le *contexte accessible*. En effet, les agents sont dans un contexte dynamique, composé à la fois de leur

représentation du monde et des objets et agents qu'ils peuvent observer directement. Il peut y avoir une différence de contexte entre l'émetteur et le récepteur, chacun possédant et pouvant observer des informations différentes.

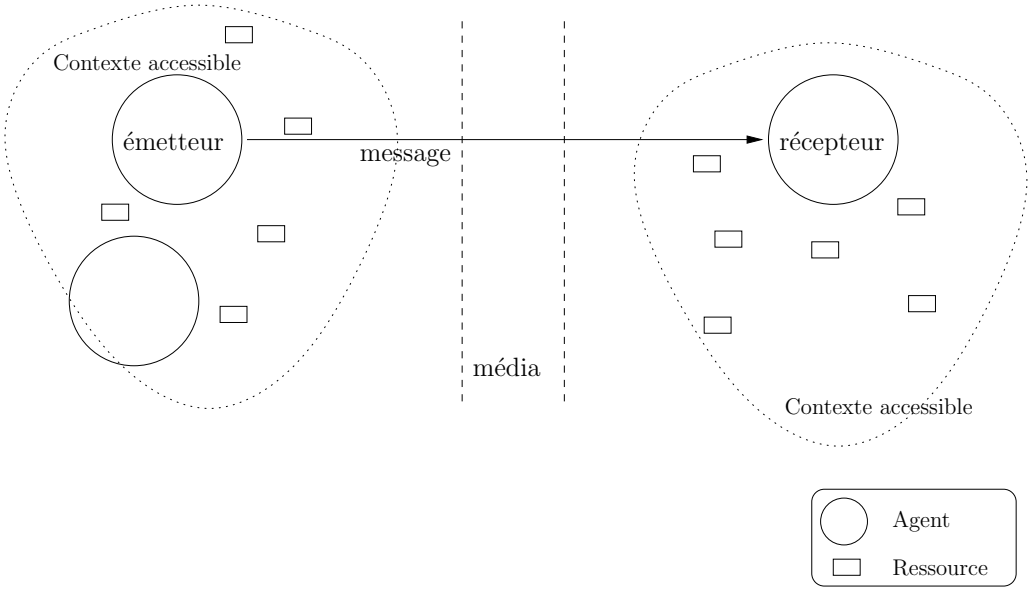


FIGURE 1.3 – Les communications directes

Les communications point-à-point correspondent à l'optique décentralisée des SMA. La gestion du temps permet d'introduire la notion de *protocole d'interaction*. Il s'agit de composer une succession de communications par la définition d'une succession temporelle valide de messages (par exemple [Huget, 2001]).

1.1.3.b Interactions indirectes

Hormis pour les communications locales, les contraintes architecturales des systèmes informatiques imposent des interactions point-à-point au niveau de la couche physique. Cependant, un certain nombre de travaux proposent d'ajouter au niveau de l'infrastructure un environnement logique pour faciliter les échanges d'information (figure 1.4). Cette famille de modèles est celle des *interactions indirectes*, qui repose sur un partage de l'information (voir par exemple [Carriero et al., 1986; Engelmore et Morgan, 1988; Omicini et Zambonelli, 1999]). Ainsi, au lieu de stocker l'information dans les agents, celle-ci est externalisée, ce qui permet à chaque agent d'émettre des informations accessibles à tous, et de récupérer les données qui l'intéressent. L'interaction est indirecte parce qu'il y a une médiation de l'acte de communication, qui provoque un découplage des actes d'émission et de lecture.

La difficulté n'est alors plus de chercher, comme pour les interactions directes, où est l'information. Elle est pour l'agent de réussir à retrouver efficacement quelles informations l'intéressent. Autrement dit, la connexion n'est plus entre deux agents mais directement entre une information

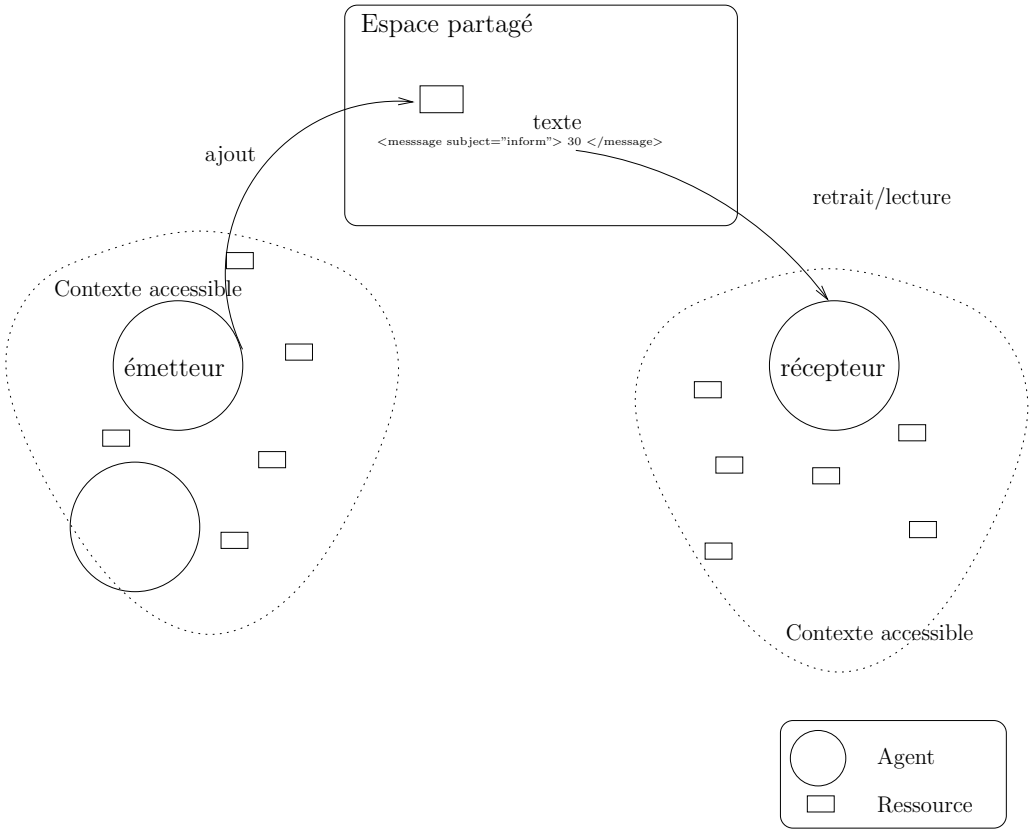


FIGURE 1.4 – Les interactions indirectes

et un agent. Cette connexion peut être rendue difficile par le volume des informations.

1.2 L'écoute flottante

Les modèles d'interactions directes sont, comme nous l'avons vu en section précédente, supportés par des communications point-à-point. Ces communications comprennent seulement deux interlocuteurs, l'émetteur et le récepteur, qui s'échangent des messages. Même lorsque plusieurs agents sont contactés par un même agent, par exemple dans le cadre d'un protocole, il s'agit en fait de plusieurs communications point-à-point exécutées parallèlement ou séquentiellement. Ainsi, les communications directes ne permettent pas de communiquer avec plusieurs agents en tant que groupe.

Par ailleurs, lorsqu'un message est émis par communication directe, l'émetteur est le seul décideur de quels seront les destinataires. Un agent tiers intéressé par ce message doit faire connaître son besoin à l'émetteur avant l'émission, ce qui implique plusieurs difficultés :

– La temporalité du besoin, qui peut être ponctuel, continu et/ou changeant, nécessite une mise à jour constante auprès de tous les agents potentiellement détenteurs de données intéressantes.

- Si les agents du système ne sont pas connus (système ouvert), les agents doivent pouvoir déterminer quels sont les agents potentiellement détenteurs de données intéressantes.
- L'émetteur doit être capable de comprendre et gérer les besoins des autres agents, et cette gestion représente un coût de traitement. De plus, les agents doivent être supposés honnêtes, dans le sens où il faut supposer qu'un émetteur connaissant un agent intéressé par son message lui transmet effectivement ce message.

Ces problèmes sont résolus dans les modèles de communications indirectes et les systèmes Publish/Subscribe [Eugster et al., 2003]. En effet, dans ces modèles, l'expression des besoins des récepteurs détermine le choix des destinataires. Par contre, l'émetteur des informations ne pourra pas choisir ses récepteurs.

Le choix d'un modèle, direct ou indirect, reflète donc la volonté de privilégier les besoins des émetteurs ou les besoins des récepteurs. Pourtant, de nombreuses interactions du monde réel sont une conjonction de ces besoins vis-à-vis de la communication. Elles étendent ainsi le modèle du dialogue limité à deux interlocuteurs. Le concept le plus utilisé jusqu'ici dans le monde multi-agents est l'*écoute flottante* [Balbo, 2004; Legras et Tessier, 2004; Platon et al., 2007b]. L'écoute flottante se rattache à la façon dont les agents communiquent.

Nous avons vu que l'interaction est composée de deux éléments, la communication et l'action qui en résulte. Cependant, la communication directe n'a pas d'effet sur d'autres agents que les acteurs particuliers de la communication que sont l'émetteur et le récepteur [Tummolini et al., 2004]. En effet, puisque la communication n'est pas observée par les autres agents, elle ne peut avoir d'impact sur eux.

L'écoute flottante implique des agents n'étant ni l'émetteur, ni l'un des récepteurs que l'émetteur a choisi. Il s'agit de permettre à des agents d'écouter les conversations qui les intéressent, ou au contraire de diffuser aux agents qu'ils jugent intéressant de contacter, sans pour autant les connaître individuellement, et ceci de façon dynamique. Un agent aura donc la possibilité, suivant des critères tels que son taux d'occupation du moment ou sa position, d'écouter ou non les messages qui passent "autour de lui".

1.2.1 L'écoute flottante : un phénomène naturel dans le monde réel

Le travail coopératif est un domaine d'étude situé à la croisée des sciences cognitives, de la sociologie et de la théorie des organisations. Il porte sur l'organisation des acteurs collaborant dans le cadre du travail. Un certain nombre d'articles dans ce domaine, comme [Dugdale et al., 2000; Rognin et al., 1998], ont souligné l'importance des phénomènes complémentaires avec la communication directe.

Les régulateurs de salle de contrôle de trafic aérien [Rognin et al., 1998] ont besoin, pour une bonne réalisation de leurs tâches, de mettre en oeuvre trois processus, l'attention mutuelle, la surveillance mutuelle et la communication. La communication est l'échange d'informations par voie verbale (écrite ou orale) ou non-verbale (gestes, regards). L'attention et la surveillance mutuelles représentent la possibilité pour chacun des opérateurs d'observer à la fois les commu-

nications et actions des autres agents, mais aussi les outils que ceux-ci utilisent. Cette capacité de diriger leur attention vers leur environnement leur donne une attitude proactive vis-à-vis de ce qu'ils observent, par exemple pour réparer une erreur commise par un collègue.

Dans [Dugdale et al., 2000], le problème est de simuler un centre d'appel du SAMU. L'observation du comportement réel des agents humains dans le centre a amené les auteurs à souligner l'importance de l'écoute flottante dans l'efficacité des agents. En effet, des informations du type "déjà occupé dans une conversation" peuvent apparaître d'une simple observation du fait qu'il se produit un échange, sans avoir besoin de connaître le contenu du message.

Il a également pu être observé que capter des bribes de messages, ici unilatérales, pouvait entraîner des actions opportunistes, par exemple comprendre le cadre global de la conversation tenue par son voisin et prévoir ce dont il aura besoin avant qu'il n'ait besoin de le demander explicitement.

Lorsque les régulateurs sont dans l'incapacité d'observer leurs collègues, la réalisation de la tâche est ralentie, ce qui permet de conclure que l'écoute flottante joue en faveur de l'efficacité du travail. Ainsi, lorsque l'activité dans la salle est à son plus haut niveau, et donc que l'écoute flottante ne joue plus puisque tous les agents sont occupés, on observe une diminution substantielle de la rapidité de traitement de l'ensemble des appels.

De cette étude, on déduit plusieurs remarques : d'une part, le fait de connaître l'existence d'une interaction entre deux agents peut modifier le comportement des agents alentours, et cela dans une logique de gain de performance. D'autre part, ce système d'écoute mutuelle flexible permet l'émergence de comportements opportunistes concernant le contenu même du message. En effet, si chacun des régulateurs poursuivait ses conversations de façon strictement point-à-point, les autres agents ne pourraient pas adopter ces comportements opportunistes par manque de connaissance de leur contexte.

L'idée principale qui a entraîné l'utilisation de l'écoute flottante dans les systèmes multi-agents est donc de donner plus de flexibilité aux communications de façon à améliorer la diffusion des informations.

1.2.2 Les systèmes utilisant l'écoute flottante

Nous présentons dans cette section les principaux systèmes fondés sur l'utilisation de l'écoute flottante.

Le modèle OTTO (*Organizing Teams Through Overhearing*) et son extension LOTTO (*OTTO for Large numbers of agents*) [Legras, 2003; Legras et Tessier, 2004] utilisent l'écoute flottante dans une politique d'aide et d'optimisation de la connaissance. Dans le contexte particulier d'agents autonomes utilisant des communications locales, les auteurs utilisent les messages diffusés par les autres agents pour mettre à jour les connaissances de chaque agent les recevant. L'écoute flottante n'est pas choisie par le récepteur, c'est à dire qu'il reçoit en diffusion tous les messages émis à proximité. Cette approche est appliquée dans le cadre de missions aériennes réalisées par des équipes d'agents autonomes.

L'écoute flottante permet la mise à jour dynamique de la représentation du monde (groupes, agents) par le biais de messages adressés à la cantonade. Un protocole opportuniste simple a été mis en place : lorsqu'un agent reçoit un message d'information contenant une information fausse selon ses croyances, il ré-émet un message donnant ses propres informations. A l'exécution, ce protocole a permis de valider l'hypothèse d'un gain de cohérence des croyances personnelles de l'ensemble des agents de l'équipe. La principale limite de cette proposition réside dans sa mise en oeuvre par la diffusion, qui n'est pas adaptée à des systèmes multi-agents de grande taille, et/ou dont les agents s'échangent de nombreux messages (section 2.1.3.a).

Dans STEAM (*a Shell for TEAMwork*) [Kaminka et al., 2002], le suivi des activités sociales est réalisé en observant les messages que les agents échangent. L'objectif est ici de surveiller le système multi-agents. Par comparaison entre les messages réellement captés et la probabilité de capter certains types de messages suivant l'état interne des agents, le système essaie d'obtenir une estimation de l'état actuel de l'agent et plus globalement de l'équipe.

Cette étude montre comment il est possible d'extraire des informations complexes grâce à l'observation des messages échangés. Les auteurs proposent également un protocole de suggestion opportuniste réutilisant les informations obtenues par la surveillance afin de faciliter la réalisation des tâches de l'équipe. Cependant, l'écoute flottante ne fait pas partie dans ce système du mode de communication "normal" entre agents. Il s'agit d'un privilège du surveillant qui centralise les informations. Les autres agents du système, qui sont ceux réalisant effectivement la tâche, ne peuvent pas écouter les messages, ce qui restreint fortement la possibilité d'en tirer profit. De plus, le surveillant doit s'inscrire auprès des agents dont il souhaite écouter les messages, ce qui contredit les travaux sur l'écoute flottante selon lesquels un message peut être entendu sans que l'émetteur ne le veuille ou même ne le sache (par exemple [Platon et al., 2007b; Tummolini et al., 2004]).

Le *channeled Multicast* [Busetta et al., 2002] met en oeuvre une diffusion ciblée par le biais de canaux de communication. Le système permet en effet aux agents de découvrir, diffuser et s'inscrire à des thèmes pour recevoir les messages correspondants. La mise en oeuvre est effectuée par un système d'inscription à des canaux, puis de diffusion par leur biais. Le choix des récepteurs prend donc en compte à la fois l'émetteur et les récepteurs, qui sont mis en relation par le canal. La difficulté pour mettre en oeuvre l'écoute flottante est dans l'unicité du critère de réception : les agents écouteurs sont contraints d'écouter tous les messages d'un canal, et non uniquement ceux qui les intéressent. Si leur intérêt ne correspond pas exactement au thème d'un canal, ils vont donc recevoir un certain nombre de messages inutiles.

Nous pouvons noter que Busetta met en évidence l'inadaptation des langages de communication actuels à la mise en oeuvre de l'écoute flottante. En effet, les actes de langage n'associent pas de sémantique à la notion de récepteurs multiples, dans ce cas tous les agents inscrits à un canal.

Le Mission Rehearsal Exercise [Traum et Rickel, 2002] est une application conçue pour simuler au mieux la réalité de missions militaires. Les auteurs ont axé leur recherche sur l'aspect multi-modal de l'interaction : messages, signaux, gestes, *etc.* L'application a nécessité de simuler,

entre autres, l'écoute flottante entre les différents agents. L'écoute flottante est nécessaire dans une optique de réalité virtuelle, car un grand nombre de comportements sociaux humains en contexte fortement interactionnel ne peuvent être décrits que par ce biais. Comme pour OTTO et LOTTO, la réalisation effective utilise la diffusion, ce qui en fait un système fonctionnellement coûteux. Le calcul de la réception d'un message prend en compte des critères fixes de l'environnement simulé, comme la distance à l'émetteur ou la force de son message.

Dans [Balbo, 1999; Balbo et Pinson, 2001], Balbo utilise la souplesse de l'écoute flottante dans le cadre du diagnostic des perturbations et du traitement des fausses alarmes dans un réseau de bus. Il définit le modèle ESAC (*Environnement comme Support Actif de la Communication*) dans lequel la diffusion des messages est directement assurée par l'environnement, comme support actif de communication. La diffusion des messages est assurée par le biais de filtres sur le contenu du message. Un agent, quelle que soit sa classe, peut déposer des filtres dans l'environnement, lesquels contiennent la priorité et la condition de réception : par exemple, certains messages envoyés par les agents *BUS* sont reçus par tous les autres agents *BUS* situés sur la même ligne que l'émetteur. Lors de l'envoi d'un message, l'émetteur donne le filtre d'émission qu'il souhaite appliquer au message afin de l'envoyer aux agents qu'il souhaite contacter.

Nous notons que ce type de filtre permet de contacter des agents sans que l'émetteur ne possède de connaissance sur eux, et ne nécessite pas d'autre intermédiaire que l'environnement. Par ailleurs, il est possible pour les agents de déposer des filtres d'interception de message, afin de recevoir les messages suivant leur contenu. Ce système a prouvé son efficacité dans la gestion des données incohérentes et le diagnostic des perturbations.

Enfin, dans le cadre d'une simulation de e-commerce, [Platon et al., 2006] montre que la diffusion d'informations par l'environnement permet d'augmenter substantiellement le nombre de contrats conclus. De nouveau, l'écoute flottante, par l'augmentation du niveau de connaissance des agents, permet à ceux-ci de moduler leur comportement et ainsi d'améliorer l'efficacité globale du système.

Nous pouvons résumer cette étude de la littérature en mettant en avant les différents avantages apportés par l'utilisation de l'écoute flottante dans les systèmes multi-agents :

- Au niveau de la représentation du monde, l'écoute flottante permet à l'agent d'obtenir une meilleure connaissance sur laquelle fonder ses actions. Non seulement le contenu des messages, mais la simple observation de l'émission d'un message améliore la connaissance de l'agent écoutant.
- L'information obtenue par écoute flottante peut être utilisée de différentes façons :
  - Inférence de connaissance au niveau multi-agents.
  - Mise en oeuvre de comportements proactifs par les agents.
- L'écoute flottante est nécessaire pour simuler les comportements de groupes humains.
- L'émetteur n'est pas le seul décideur des agents recevant ses messages, les autres agents peuvent aussi indépendamment choisir de recevoir un message.

1.2.3 Les modèles d'écoute flottante

Une première idée des communications multi-parties est de les définir comme des communications directes, auxquelles est ajoutée l'écoute flottante, c'est à dire la possibilité qu'ont les agents de recevoir une partie des messages qui ne leur sont pas adressés par l'émetteur. Nous étudions dans cette section les modèles prenant en compte les effets indirects des communications.

Un premier concept proche des communications multi-parties est la notion de *Behavorial Implicit Communication* (BIC) [Tummolini et al., 2004]. Dans le cadre de systèmes coopératifs de réalisation de tâches, les BICs sont l'ensemble des interactions observables de façon non explicite, *i.e.* l'information véhiculée par les actions ou les communications des autres agents. Ces informations sont transmises grâce à l'environnement. Nous retrouvons ici l'objectif de tirer parti d'informations traditionnellement indisponibles. Trois propriétés sont nécessaires pour la mise en oeuvre des BICs : l'observabilité des actions et de leur résultat, la capacité des agents à en déduire une information correcte (et éventuellement une action), et enfin la capacité des agents à agir en prévoyant les effets que cette observation va provoquer. Le cadre proposé nécessite des agents très coopératifs (permettant notamment l'accès à leur état intentionnel), et prend en compte toutes les possibilités d'actions, d'interactions et de connaissances, lesquelles sont difficilement modélisables *a priori* dans un système hétérogène et ouvert.

[Gutnik et Kaminka, 2005] ont proposé une modélisation des dialogues par des réseaux de Petri. Ces travaux sont orientés sur la fonction de surveillance par écoute flottante, l'objectif étant de déduire *a posteriori* les protocoles effectués à partir de messages captés. Ces travaux ont été appliqués sur des protocoles standards de la FIPA [FIPA, 2002a]. Cependant, l'écoute flottante elle-même n'est pas modélisée. Comme pour STEAM, l'utilisation de l'écoute flottante n'est pas intégrée au fonctionnement du système multi-agents observé.

A notre connaissance, le modèle de Platon [Platon et al., 2004] est le plus générique, considérant l'écoute flottante indépendamment du domaine d'application et des bénéfices que l'on souhaite tirer de son utilisation. L'introduction du T-compound comme patron de conception (*design pattern*) permet une représentation graphique de l'écoute flottante pour la modélisation des interactions, en s'appuyant sur les travaux existants dans le domaine objet.

Par ailleurs, dans [Platon et al., 2005] est mise en avant la nécessité d'implémenter l'écoute flottante sans passer par l'émetteur, mais par l'environnement. Ce travail est approfondi dans [Platon et al., 2006], de façon à permettre un niveau supérieur d'interaction, les *tag interactions*, dont le principe est illustré dans les figures 1.5 et 1.6.

Ce modèle s'appuie sur la notion de séparation du corps et de l'esprit des agents. Les deux entités sont séparées, l'esprit étant l'entité classiquement considérée comme l'agent, auquel est ajouté un corps virtuel qu'il ne contrôle que partiellement. Ce corps possède des étiquettes (*tags*), qui sont observable par les autres agents. L'environnement est en charge du corps, il valide ou non les influences de l'esprit. L'idée fondamentale est ici de permettre un contrôle des actions des agents, y compris au niveau de leur partie publique, c'est à dire ce qu'ils présentent à l'observation.

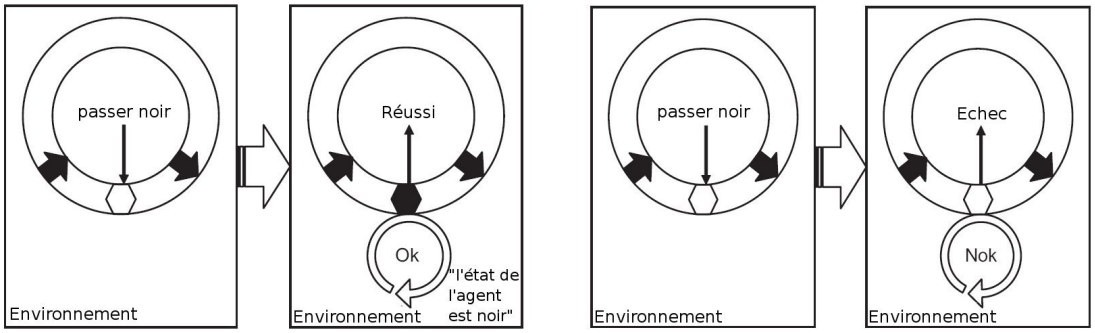


FIGURE 1.5 – Rôle de l'environnement dans les *tag interactions*

La figure 1.5 illustre ce principe. Les deux schémas de gauche montrent un agent tentant de changer la couleur de son étiquette vers la couleur noire. Cette modification est validée par l'environnement, grâce à un ensemble de règles qu'il possède. Ces règles entraînent le passage de la couleur de l'étiquette à noir, le signalement de l'action à l'esprit et l'annonce de cette modification aux alentours. Les deux schémas de droite montrent, pour la même action, une réaction de l'environnement différente : cette fois-ci, l'environnement refuse de modifier la couleur de l'étiquette, auquel cas il ne se passe rien au niveau du corps, et l'agent est informé de l'échec de sa tentative.

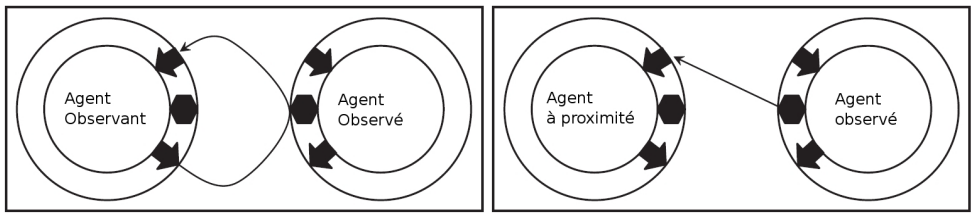


FIGURE 1.6 – Observation et perception spontanée dans les *tag interactions*

La seconde figure (Fig. 1.6) montre quant à elle la façon dont sont diffusées les informations observables. Le schéma de gauche montre une observation classique, l'agent observant tente d'observer par le biais de ses effecteurs l'état du second agent, et l'environnement fournit en retour cette observation au capteur de l'agent observant. Le schéma de droite montre une propagation de l'état de l'agent observé sans qu'il y ait requête de la part de l'autre agent. C'est notamment le cas lors d'un changement d'état d'une étiquette, mais cela peut aussi être dû à une règle propageant les états dès que deux agents sont suffisamment proches l'un de l'autre : ce sont des règles de l'environnement qui déterminent si les étiquettes sont diffusées ou non. Il s'agit de mettre en oeuvre une conscience des autres, comme lorsqu'une personne se rend compte soudainement que quelqu'un d'autre est entré dans la même pièce. Nous reviendrons sur ce concept particulier en section 1.4.

Les travaux sur les *tag interactions* sont donc en accord avec la notion de propagation de

l'information de façon éventuellement involontaire. Ces travaux montrent l'utilité de l'observabilité des agents, même lorsque ceux-ci sont purement logiciels. Cependant, ce modèle fait reposer toute propagation sur les règles de l'environnement, du fait de la séparation nette entre l'agent et l'environnement. Ainsi, une première limite du modèle est que l'agent ne peut exprimer ses besoins que de manière ponctuelle, par ses effecteurs, et non en continu. De plus, la forme des règles de propagation (quels sont les prédicats accessibles, quelle formalisation des règles ?) n'est pas explicitée.

MIC* [Gouaïch et al., 2005] est un environnement de déploiement formel fortement orienté vers l'informatique diffuse. Il s'agit d'un modèle indépendant à la fois des modèles d'agents et d'interaction, et de l'implémentation. De la même façon que précédemment, la diffusion et la propagation des objets d'interaction sont gérées par l'environnement. Une fois le message produit, il n'est plus sous la responsabilité de l'agent l'ayant émis, et le calcul des autres agents devant percevoir ce message est effectué par l'infrastructure. La contrainte forte de l'informatique diffuse est la connectivité intermittente de ses éléments, MIC* est donc prévu pour fusionner à la volée les données émises par chaque agent. Les capteurs et les effecteurs sont représentés formellement de la même façon que les messages, leur réification permettant à l'algèbre proposé de gérer de façon standardisée l'ensemble des concepts liés à l'interaction. A l'inverse des *tag interactions*, l'ensemble des perceptions d'un agent dépend de la corrélation entre ses capteurs et les effecteurs des agents émetteurs, l'environnement ne jouant qu'un rôle de calcul. Ce choix a deux conséquences : les règles de diffusion des messages dans l'environnement ne sont pas modélisées, et l'émetteur n'est pas assuré que ses messages sont transmis, puisque le récepteur contrôle entièrement ses capteurs.

La difficulté inhérente au support effectif des nouvelles formes d'interaction comme l'écoute flottante a conduit un certain nombre de travaux à ignorer cette problématique, ou à la poser comme hypothèse.

Par exemple, dans [Kamali et al., 2006], les auteurs proposent une nouvelle sémantique pour certains actes de dialogue dans le contexte de communications de groupe, c'est à dire les communications élargies du simple émetteur/destinataire à un ensemble d'auditeurs. Dans ce travail, les auteurs observent que modifier la sémantique des actes de langage en considérant les communications entre plusieurs agents en tant qu'unité, et non comme une multiplicité de communications un-à-un, permet d'améliorer les capacités d'inférence des agents. Par contre, la façon de mettre en oeuvre les hypothèses fortes que cette sémantique présuppose, comme la connaissance de l'émetteur et de tous les destinataires par tous les participants, n'est pas discutée.

Dans [Gutnik, 2005], les auteurs proposent des stratégies d'écoute flottante pour de grands groupes d'agents réalisées sur des simulations, mais n'abordent pas non plus la question du support dans le cadre de systèmes multi-agents réels. D'un autre côté, des travaux comme ceux de Huget [Huget et Demazeau, 2004] présentent les communications multi-parties tout en fondant leur support sur trois mécanismes classiques distincts, la diffusion dans un environnement spatial, les forums et les communications directes. La notion de communication unique et les avantages

qui en découlent sont alors perdus.

1.3 Une typologie des communications multi-parties

Nous avons vu que les approches mises en oeuvre dans des systèmes réels (diffusion, canaux, souscription...) présentent des limites fortes. De plus, à notre connaissance, il n'y a pas de modèles génériques ni d'infrastructure unifiée pour les représenter. Afin de proposer un nouveau modèle palliant ces limites, nous mettons en évidence les caractéristiques des conversations humaines.

1.3.1 De nouvelles caractéristiques définissant les communications

De façon générale, les communications multi-parties présentent la spécificité d'avoir un émetteur, des destinataires prévus et des récepteurs imprévus. Une étude plus fine est nécessaire pour extraire les différents types de rôles pris par les agents.

Kumar *et al.* [Kumar et al., 2000] ont étudié les propriétés des communications de groupe dans les sociétés humaines et leur application à des sociétés d'agents. Ils identifient un certain nombre de situations qui devraient être permises si l'on souhaite obtenir la richesse des communications du monde réel. Ainsi, il peut y avoir un ou plusieurs *destinataires*, connus (par exemple une mailing list) ou inconnus (par exemple un message radiophonique). De même, le récepteur ne connaît pas nécessairement l'émetteur du message, ce qui n'empêche pas sa portée, par exemple dans le cas d'un panneau "peinture fraîche". Les récepteurs effectifs peuvent ne pas avoir été prévus, par exemple un message sur un forum peut être lu par plus de personnes qu'initialement prévues. La communication peut porter l'intention de rôles particuliers, par exemple l'attente d'une participation active de la part d'un sous-groupe de récepteurs. Ces rôles ne sont pas nécessairement pré-attribués par l'émetteur, par exemple un enseignant s'adressant à "tous les étudiants ayant fait l'exercice 1".

Les auteurs ont identifié huit contraintes afférentes aux communications humaines :

- *Contrainte du destinataire* : les communications doivent pouvoir être adressées à des individus aussi bien qu'à des groupes d'individus. De plus, un groupe peut avoir des membres stables et connus ou, au contraire, ceux-ci peuvent être individuellement inconnus de l'émetteur.
- *Contrainte de l'émetteur* : les communications doivent pouvoir être émises par un individu aussi bien que par un groupe d'individus. Typiquement, un individu agit au nom d'un groupe lorsque l'émetteur est le groupe lui-même (par exemple une lettre d'une entreprise).
- *Contrainte de récepteur* : les communications peuvent être entendues par des récepteurs inattendus et par des écouteurs.
- *Contrainte d'acteur* : les communications doivent supporter des acteurs inattendus, qui sont éventuellement différents des récepteurs attendus d'un message.
- *Contrainte de conscience*[7] de l'acteur : les communications doivent supporter des émetteurs

7. Le terme original utilisé est *awareness*. Dans le cadre de ces contraintes, nous pourrions aussi traduire par

ne connaissant *a priori* pas le rôle des récepteurs.
- *Contrainte de conscience de l'émetteur* : les communications doivent pouvoir être émises vers un groupe dont l'émetteur ne connaît pas les individus membres.
- *Contrainte de conscience du récepteur* : les récepteurs n'ont pas nécessairement connaissance des autres récepteurs de la communication qu'ils perçoivent.
- *Contrainte d'origine* : les récepteurs peuvent ne pas avoir connaissance de l'émetteur d'une communication qu'ils perçoivent.

A partir de ces contraintes, il est nécessaire de distinguer ce qui relève de l'infrastructure de communication, et ce qui relève de la sémantique. La principale difficulté est la spécification des groupes, par d'autre moyens que l'adresse. La contrainte de l'émetteur est purement sémantique, puisqu'un message a, de fait, un seul émetteur. De la même façon, les contraintes de conscience sont principalement d'ordre sémantique, même si nous pouvons en tirer les spécifications suivantes : le médium de la communication permet un découplage de la transmission, et les agents ne sont pas omniscients quant à la communication. De plus, le médium doit être observable, de manière à permettre des récepteurs inattendus.

1.3.2 Les rôles des agents

À partir de ces contraintes, nous proposons 4 critères déterminant le rôle de l'agent dans la communication :
- L'*intention* : le récepteur est-il prévu et accepté ?
- La *connaissance* : l'agent est-il connu des autres participants ?
- Le *rôle attendu* : le récepteur a-t-il un rôle actif dans le dialogue ou est-il juste un auditeur passif ?
- L'*initiateur* de la réception : est-ce une initiative de l'émetteur ou du récepteur ?

Sur la base des travaux existants dans le domaine des SMA [Dignum et Vreeswijk, 2003; Huget et Demazeau, 2004], mais aussi dans les domaines de la psychologie et des sciences sociales [Branigan, 2006], ces critères vont nous permettre de définir les principaux rôles pouvant être joués dans la communication (Fig. 1.7) :
- Émetteur (*Speaker*) : l'agent est l'émetteur, il peut être connu ou non des participants.
- Destinataire (*Addressee*) : l'agent est un récepteur, il est prévu et connu par l'émetteur, et il participe activement au dialogue. Il reçoit le message par l'initiative de l'émetteur.
- Auditeur (*Auditor*) : l'agent est un récepteur, il est prévu et connu par l'émetteur, mais il est passif vis-à-vis du dialogue. Il reçoit le message par l'initiative de l'émetteur.
- Groupe de destination (*communication group*) : tous les agents du groupe sont récepteurs. Le groupe est prévu mais ses membres ne sont pas nécessairement connus par l'émetteur. Les agents peuvent être des participant actifs ou des participants passifs. Ils reçoivent le

connaissance ou attention. La problématique de l'utilisation et de la traduction du terme awareness est discutée plus avant en section 1.4.

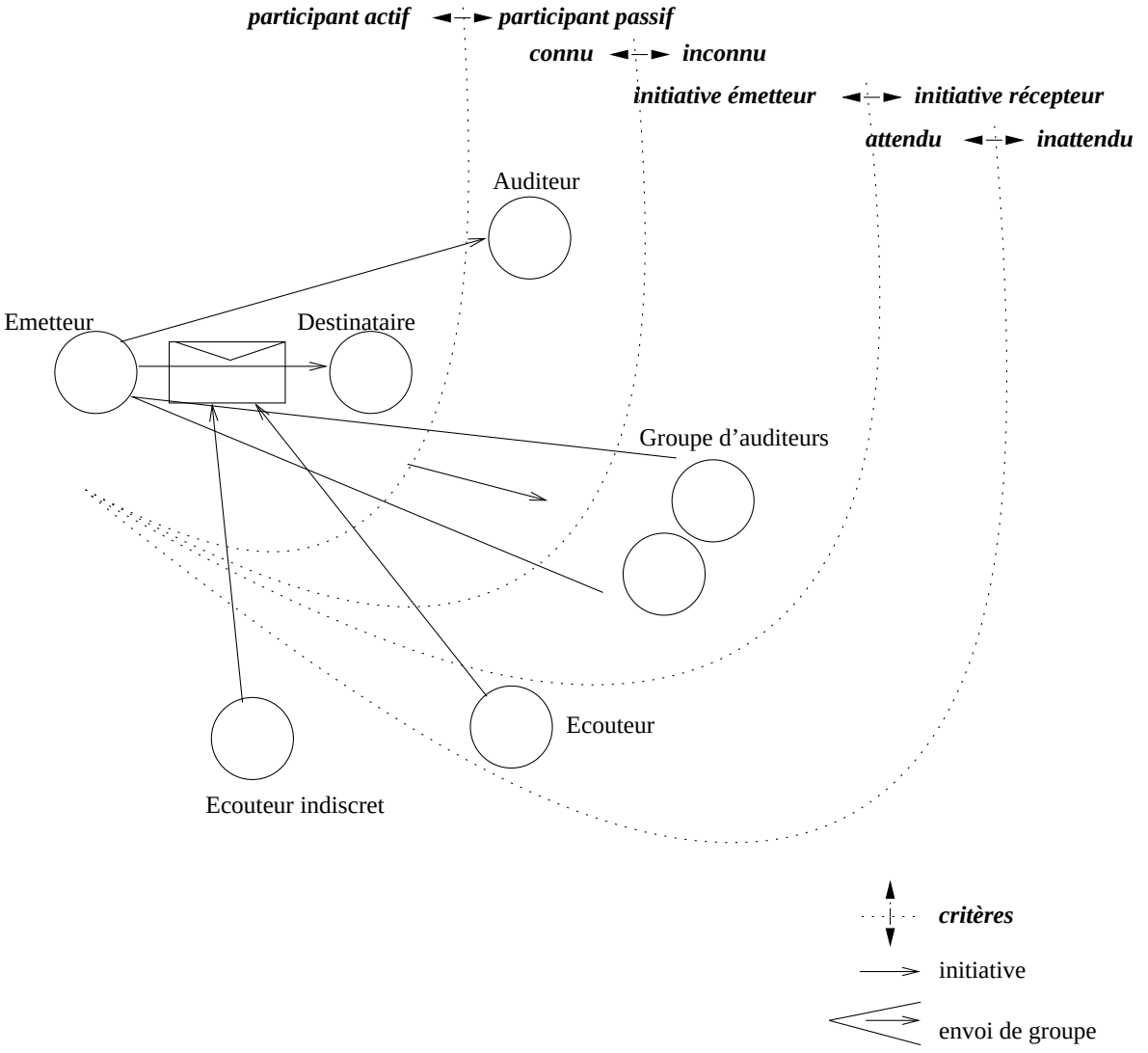


FIGURE 1.7 – Rôles des agents dans une communication multi-parties

message par l'initiative de l'émetteur.

- Écouteur (*Listener*) : l'agent est un récepteur, il est prévu et éventuellement connu de l'émetteur, mais il ne participe pas au dialogue. Il reçoit le message par sa propre initiative.
- Écouteur indiscret (*Overhearer*) : l'agent est un récepteur, il n'est pas prévu mais peut être connu de l'émetteur. Il ne participe pas au dialogue, et reçoit le message par sa propre initiative.

Hormis le rôle d'émetteur, dont la cardinalité est nécessairement de 1, chacun des autres rôles peut être joué par 0 à n agents. Nous considérons que la possibilité que les messages soient écoutés par d'autres agents est connue de tous les participants. A ce titre, nous n'étudions pas le cas de l'écoute frauduleuse (*eavesdropping*) et d'éventuels agents "espions".

Du point de vue dialogique, les agents ayant l'un des deux premiers rôles sont les participants actifs, i.e. prenant part à la conversation, les auditeurs sont les participants passifs, et les écouteurs sont des non-participants. Le cas "groupe de destination" est le moins classique, il s'agit d'une initiative de l'émetteur envers un groupe dont les membres ne sont pas nécessairement connus. Par exemple, faire une annonce au club photographie de l'université n'implique pas de connaître chacun de ses membres. L'émetteur connaît uniquement un certain nombre de conditions définissant le groupe recherché, comme l'appartenance à un groupe. Il est à noter que ce type de communications sur des conditions et non par adresses permet de faciliter l'interchangeabilité des agents : si un nouvel agent respectant les critères apparaît, il reçoit immédiatement les messages le concernant.

Le cas particulier où l'émetteur n'est pas connu des participants relève d'interactions indirectes, la médiation de l'interaction permettant l'anonymat. Deux exemples sont la stigmergie, modèle selon lequel les agents se coordonnent en fonction des modifications faites par les autres dans l'environnement (par exemple l'ajout d'une trace), et les espaces de tuples, que nous abordons plus en détail dans le chapitre 2.

Identifier les rôles permet de traiter les modèles existants comme des cas particuliers de communications multi-parties. Ainsi, les communications directes mettent en jeu un émetteur et un destinataire. Les communications indirectes, à la façon d'un espace de tuples, mettent en jeu un émetteur (connu uniquement si le message contient lui-même l'information) et des écouteurs, dans le sens où c'est à leur initiative qu'ils accèdent à l'information. Les communications locales, qui sont une diffusion limitée dans l'espace, mettent en oeuvre un émetteur, un ou plusieurs destinataires et éventuellement des auditeurs.

Il ressort de cette analyse un certain nombre de spécifications que doit avoir un modèle pour pouvoir mettre en oeuvre les communications multi-parties. Le point de rupture entre interactions directes et indirectes concerne l'initiative de la réception. Un système permettant la présence simultanée de destinataires et d'écouteurs doit résoudre la problématique de la mise en corrélation d'initiatives d'origines différentes pour la décision de distribution des messages. Autrement dit, les récepteurs d'un message sont la conjonction des récepteurs découlant de l'initiative de l'émetteur, et d'éventuelles initiatives de récepteurs. Par ailleurs, la possibilité de récepteurs imprévus, voir inconnus, implique que l'émetteur ne maîtrise pas nécessairement le canal de communication, par exemple une émission en clair sur un réseau wifi. Dans le cadre de réseaux filaires classiques, cela n'est réalisable que par le biais d'un middleware ou d'une architecture particulière.

Pour affiner ce principe, il est nécessaire de caractériser la façon dont s'exprime l'*initiative* permettant la connexion entre l'information et les récepteurs.

1.4 La conscience des autres

1.4.1 Notion de conscience des autres

L'écoute flottante dérive du principe de conscience des autres (*awareness*) [Heath et al., 2002; Robertson, 2002; Warren, 1999], c'est à dire la volonté et la capacité de percevoir des actions ou les effets d'actions dans son environnement. Elle a longtemps été considérée comme un état subi. Elle serait alors uniquement le résultat de stimuli extérieurs, le récepteur étant passif. Cependant, un certain nombre de travaux récents en psychologie et en sociologie discutent l'importance d'un état actif dans la conscience des autres.

Ainsi, dans [Heath et al., 2002] les auteurs montrent que la conscience des autres n'est pas seulement un état de disponibilité à l'environnement, mais aussi une capacité discriminante qui consiste à filtrer les informations issues de l'environnement qui ont une certaine importance. Ceci suppose donc de la part du récepteur une compétence de sélection [Robertson, 2002] selon la pertinence de l'information. De plus, même au niveau physiologique, [Warren, 1999] souligne qu'écouter est une activité de réception qui est le résultat d'une décision d'action. L'awareness est donc dirigée vers un sujet, par le biais d'un organe récepteur. Nous pouvons alors préciser la traduction du terme *awareness* par l'*attention*, qui porte l'idée d'action volontaire, plutôt que conscience des autres.

Fonctionnellement, ceci signifie que les stimuli proviennent de l'environnement. Ces stimuli peuvent ou non être évitables. Par exemple, un cri émit à proximité sera nécessairement entendu, du fait des règles de l'environnement physique qui transmet ce signal. Il est donc important de tenir compte de la topologie de l'environnement. Pour les signaux qui ne sont pas transmis automatiquement par les règles de l'environnement, la perception dépend de deux facteurs : la capacité à percevoir et la volonté de percevoir.

La capacité à percevoir relève de deux contraintes, la première étant celle du média par lequel transite l'information, et la seconde étant liée aux capacités de l'organe récepteur. Par exemple, la topologie de l'environnement affecte la transmission du signal, celui-ci devant parvenir à portée de l'organe de réception pour pouvoir être perçu.

La volonté de percevoir, quant à elle, est une décision du récepteur, c'est elle qui permet l'activation de l'organe de réception.

Par exemple, une discussion ayant lieu à une distance raisonnable peut être écoutée, mais si le point d'émission du message est trop lointain, la sensibilité de l'organe récepteur lié à la topologie de l'environnement empêche cette écoute.

Dans le cadre de la robotique [Bajcsy, 1988] et des systèmes multi-agents [Weyns et al., 2004], la notion de *focus* a été proposée pour modéliser l'attention des robots ou agents. Elle est en général étudiée du point de vue du récepteur. Il est possible d'envisager le fait d'adresser un message comme un *focus* de la part de l'émetteur. Le champs de ce *focus* est défini par l'identifiant du récepteur choisi. De façon plus complexe, le cas des communications de groupe illustre ce cas. En effet, diriger un message sans en connaître a priori les récepteurs, en fonction de son contexte de transmission, dénote une initiative de l'émetteur.

Le focus est donc une initiative qui provient soit de l'émetteur, soit du récepteur. Cette initiative s'exprime en fonction du contexte dans lequel le message se trouve, autrement dit de l'état courant de l'environnement de transmission du message.

Dans ce qui suit, nous allons voir comment utiliser le contexte dans le cadre des communications multi-parties.

1.4.2 Contextualiser les interactions

De façon générale, le contexte d'un agent est tout ce avec quoi il peut interagir par observation et/ou action. La question du contexte a été traitée extensivement dans la littérature et a fait l'objet de nombreuses conférences (voir par exemple [Baldauf et al., 2007; Benerecetti et al., 2001; Bucur et al., 2005; Coutaz et al., 2005; Fahy et Clarke, 2004]). Cette problématique rejoint celle de la perception, à savoir l'observation des entités et évènements de l'environnement par l'agent.

La gestion du contexte se heurte à trois difficultés principales [Coutaz et al., 2005] : (i) le contexte est non seulement un état mais également une partie d'un processus, ce qui signifie que la gestion du contexte doit tenir compte de la dynamique de l'environnement. (ii) L'adaptation et l'utilisation des données du contexte implique une fusion de ces données. (iii) La gestion du contexte doit être transparente pour éviter les conflits d'utilisation de modèles éventuellement différents.

Tout comme nous avons noté que nous étudions les interactions entre agents, et non celles des agents avec les autres entités non-agents, nous nous intéressons ici au lien entre contexte et communication. Plus précisément, le contexte est lié à notre problématique de deux façons différentes :

- Les interactions entre agents forment un contexte qui peut être observé par les autres agents
- Les informations contextuelles peuvent être utilisées pour la diffusion des messages

Le premier point est le principe même de l'écoute flottante, que nous avons discuté en section 1.2.

Par rapport à une communication, la notion de contexte est tout ce qui n'est pas le message, autrement dit le contexte de l'émetteur, le contexte de la transmission du message et le contexte du récepteur. Les agents doivent pouvoir utiliser ces trois constituantes du contexte d'un message en tant que contrainte(s) de réception. Dans les communications multi-parties, il s'agit de pouvoir modifier le choix de réception en fonction de valeurs dynamiques, par exemple recevoir les messages de "tous les étudiants situés dans une salle possédant un photocopieur". Que ce soit pour une initiative en émission ou en réception, l'agent ne sait pas a priori quels sont les agents qu'il doit contacter. Ce type de description de contrainte est souvent utilisé de manière implicite, par exemple en regroupant les agents par propriétés communes au sein de groupes [Gutknecht et Ferber, 2000], ou en souscrivant à des canaux particuliers [Busetta et al., 2002].

Dans le même esprit, les mécanismes de *publish and subscribe* [Eugster et al., 2003] permettent de diffuser les informations selon un ensemble de conditions sur le thème du message(*topic-*

based), son contenu (*content-based*) ou son type (*type-based*). La médiation est fondée sur des évènements, qui sont les transmissions de messages. Le contexte pouvant influer sur le choix des récepteurs est ainsi nécessairement limité au contenu du message lui-même.

Pour un agent, la récupération d'informations du contexte peut se faire de trois façons [Baldauf et al., 2007; Benerecetti et al., 2001] : soit par l'accès directs aux capteurs, soit par le biais d'un middleware, soit par la mise en place d'un serveur d'information contextuelle. La première approche permet à l'agent d'obtenir par lui-même l'information et de l'utiliser dans le choix des récepteurs.

Dans la seconde approche, de nombreux travaux proviennent du domaine des applications mobiles en environnement pervasif. Dans [Julien et Roman, 2004], les auteurs considèrent le contexte comme l'ensemble des noeuds pouvant apporter des informations. Une distance logique est utilisée pour calculer ce contexte. En fonction de cette distance, les communications sont ou non transmises aux noeuds limitrophes. Le contexte est donc composé de la distance à laquelle les informations sont conservées, ainsi que du message retransmis lui-même. Lorsque la distance change, le routage des messages est alors modifié pour prendre en compte le nouveau contexte.

Un certain nombre de middlewares, comme [Gouaïch et al., 2005; Julien et Roman, 2004; Mamei et al., 2003; Schelfthout et al., 2006] proposent des espaces de communications partagés, pouvant fusionner suivant la proximité des hôtes. Dans ces systèmes, la mise à disposition des éléments partagés dépend à la fois du niveau architecture et du niveau coordination. Les règles de propagation ne sont pas limitées à un seul calcul de distance. Elles prennent en compte la topologie des relations entre hôtes pour transmettre les informations. Cependant, les règles restent simples et ne permettent pas l'appariement entre plusieurs éléments, ce qui limite leur expressivité. Par exemple, dans EgoSpaces [Julien et Roman, 2004], les tests liés aux règles sont effectués unitairement sur les propriétés de l'hôte, du réseau, des agents et des tuples eux-mêmes, mais ne peuvent pas comparer une propriété d'un message avec celle d'un hôte. Du point de vue coordination, le contexte est représenté par des éléments de l'espace médié auquel les agents peuvent accéder mais qui ne peuvent pas être utilisés directement pour modifier leur façon de recevoir les informations.

Enfin, la notion de serveur d'informations contextuelles permet de rassembler les informations en provenance des capteurs pour factoriser leur utilisation. Cette démarche a pour avantage de déplacer une partie de la complexité computationnelle des clients vers le serveur. Certains serveurs d'informations comme CASS [Fahy et Clarke, 2004] proposent des principes de souscription aux informations dans la lignée du *publish and subscribe*. Nous verrons dans les chapitres 3 à 5 que nous nous inspirons de cette approche.

L'utilisation d'un middleware ou d'un serveur d'information permet de séparer d'une part l'acquisition et la gestion des données du contexte, et d'autre part le raisonnement sur ces données. En vue d'un support du contexte, cela correspond conceptuellement à distinguer deux couches différentes [Bucur et al., 2005], et à envisager de façon générique la notion de contexte.

Il existe de nombreux modes de représentation des données du contexte, parmi lesquelles nous pouvons citer les tuples, de format fixé ou non, les couples attribut/valeur, étant ou non associés

à des entités, et les structures de données *ad hoc*. L'utilisation unifiée et générique du contexte requiert une cohérence de représentation et une expressivité suffisante. Nous pouvons ainsi noter l'utilisation d'ontologies [Bucur et al., 2005] liant les attributs aux concepts, ou l'approche W4 (*Who, What, Where, When*) [Castelli et al., 2007] qui fixe des quadruplets (sujet, activité, localisation, temps) comme atomes de données.

1.4.3 Contrôler les interactions

Le principe décrit jusqu'ici permet aux agents d'agir sur le choix des récepteurs en fonction de leurs besoins en émission et en réception. La conformité à d'éventuelles règles de l'environnement est confiée aux agents. Cela s'avère suffisant si les agents sont honnêtes, ou si le concepteur du système multi-agents peut vérifier le comportement des agents. Dans le cadre de systèmes ouverts et hétérogènes, c'est à dire dont les agents ne peuvent être contrôlés a priori, cette conformité implicite n'est plus suffisante [Esteva et al., 2004; Okuyama et al., 2007; Paes et al., 2007]. Si les agents peuvent exprimer un besoin envers n'importe quel message, tous les messages leur sont potentiellement disponibles. Or, il peut être souhaitable de restreindre les possibilités de réception.

Nous avons vu en section 1.4.1 que la perception réagissait aux contraintes de l'environnement. Autrement dit, si les règles de l'environnement déterminent qu'un agent doit recevoir une information, l'agent est obligé de la recevoir. C'est le cas avec les communications adressées. Nous avons également vu que si ces règles peuvent s'exprimer de façon *positive*, elles peuvent aussi s'exprimer de manière *négative*, au niveau de l'environnement d'une part, et au niveau de l'agent d'autre part. L'environnement impose des règles de portée des messages et de capacité de perception, par exemple la distance sur laquelle l'air porte la voix et les capacités de l'oreille. Au niveau de l'agent, il s'agit de la décision d'action, décision qui peut être d'écouter, c'est à dire de porter une initiative, ou au contraire de refuser les messages, de diminuer son attention.

Il convient de distinguer plusieurs concepts se rapprochant du contrôle. Les *règles*, ou *lois*, sont obligatoires et ne peuvent pas être transgressées. Les *normes* [Khadraoui et Gateau, 2005] proviennent d'une logique sociale. S'il est donc conseillé de les suivre, les agents peuvent choisir de ne pas le faire. Le support des normes peut être considéré comme un service offert par l'infrastructure, et la surveillance de leur respect (et les sanctions liées) se fait *a posteriori*. Enfin, le contrôle du point de vue agent est sa capacité à gérer ses propres capteurs. Dans le modèle des *tag interactions*, par exemple, le contrôle de l'agent sur son corps est limité à des influences, tandis que celui de l'environnement est total.

Le contrôle à l'exécution des interactions des agents suppose une gouvernance, une médiation de ces interactions par un hôte de confiance. Ce principe est mis en oeuvre dans les institutions électroniques [Arcos et al., 2007; Esteva et al., 2004], lesquelles sont garantes du respect d'un certain nombre de règles définies par le concepteur du système multi-agents. Les principaux types d'application concernés sont ceux qui nécessitent une assurance de légalité des protocoles d'interactions, et pour lesquels on ne peut pas *a priori* faire confiance à l'agent, comme par

exemple pour les applications d'e-commerce.

Une architecture client/serveur est illustrée en figure 1.8. Chaque agent A_i de l'application délègue la gestion de ses communications à un agent spécialisé G_i de la plate-forme. La vérification de la légalité des communications est réalisée par les agents du serveur S_i, qui s'appuient sur une spécification des règles et protocoles autorisés.

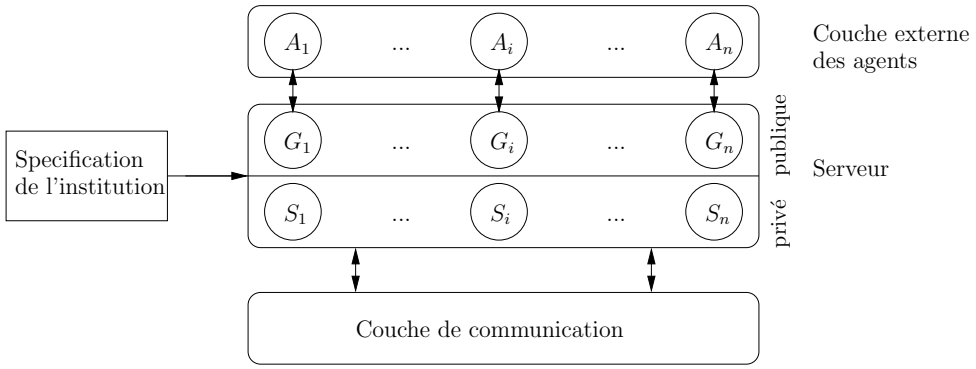


FIGURE 1.8 – Institution électronique et médiation [Esteva et al., 2004]

Cette architecture met parfaitement en oeuvre le contrôle au niveau SMA, mais elle n'est pas suffisante dans le cadre des communications multi-parties. En effet, si elle permet de limiter les interactions, elle ne donne aucun moyen de les faciliter.

L'approche du contrôle par la médiation a aussi été mise en oeuvre dans le cadre de lois locales [Zheng et al., 2007], lesquelles sont assurées par des modules exécutés sur les hôtes des agents. Dans ces travaux, l'architecture est totalement décentralisée, et l'objectif est de permettre une régulation globale à l'aide de contrôles locaux. L'absence de mise en commun des informations est néanmoins un facteur limitant, dans la mesure où le contexte de transmission n'est pas observable localement et doit être redemandé aux possesseurs de l'information à chaque fois qu'il est nécessaire.

1.5 Conclusion

Dans ce premier chapitre d'état de l'art, nous avons tout d'abord introduit le cadre général de la communication dans les systèmes multi-agents. En particulier, nous avons noté la nécessité de placer notre travail au niveau de l'infrastructure d'interaction. Nous avons aussi souligné la dichotomie existante entre les modèles de communication directs et indirects.

Ensuite, nous avons étudié la littérature sur la notion d'écoute flottante et ses avantages pour les systèmes multi-agents. L'écoute flottante permet une meilleure diffusion de l'information, et facilite la réalisation de différentes tâches, notamment la surveillance ou les pratiques opportunistes.

L'introduction des principes de l'écoute flottante dans les communications directes est à la base des communications multi-parties, par la délégation d'une possibilité supplémentaire de

réception des messages par les agents non concernés *a priori*. Cette possibilité s'apparente aux interactions indirectes, puisque le futur récepteur récupère lui-même l'information. Nous avons alors proposé une classification des rôles des agents dans les communications multi-parties, fondée sur les travaux existants dans les domaines des SMA, de la psychologie et des sciences sociales. Le rôle d'un agent dépend de quatre critères, à savoir l'intention de l'émetteur, la connaissance des participants, le rôle prévu dans l'interaction et l'origine de l'initiative ayant permis la réception.

Du point de vue multi-agents, les communications multi-parties sont des communications dont les récepteurs sont choisis par une conjonction des initiatives des émetteurs et des récepteurs. Du point de vue de l'agent, nous avons montré que l'expression des initiatives des agents leur donne la capacité d'*awareness*, autrement dit de diriger leur attention en fonction de leurs besoins.

L'*attention* crée deux nouvelles problématiques, qui sont celles de la prise en compte du contexte et celle du contrôle de la communication. Les besoins des agents peuvent s'exprimer en fonction de leur contexte courant, de celui de l'émetteur et/ou de celui du message. De plus, les agents peuvent ouvrir leur attention comme la restreindre en exprimant des besoins "négatifs", autrement dit refuser les communications de la même façon qu'ils peuvent les rechercher. Le concepteur du SMA doit selon le même principe pouvoir contrôler les interactions afin d'améliorer l'efficacité par la limitation du nombre de messages, et afin d'introduire et assurer des règles communes.

Une fois ces concepts – les communications multi-parties, l'attention, le contexte, le contrôle – définis, il est nécessaire d'étudier les différents types de support d'interaction existants en regard des fonctionnalités désirées. La problématique générale du support des communications multi-parties est celle de la sélection des récepteurs. Nous avons pu extraire un certain nombre de critères plus fins qui nous permettrons, dans le chapitre suivant, de classifier ces différents supports :

1. Besoins de l'émetteur : le choix des récepteurs prend-il en compte les besoins de l'émetteur ? Comment ce besoin s'exprime-t-il, les communications de groupe sont-elles possibles ?
2. Besoins du récepteur : le support permet-il la prise en compte des besoins des agents autres que l'émetteur, et comment ?
3. Contexte : Est-il possible de prendre en considérations des éléments du contexte, et lesquels ?
4. Contrôle : Est-il possible de réguler les interactions au niveau du SMA et/ou au niveau de l'agent ?

Chapitre 2

Les supports de communication : un état de l'art

Sommaire

Dans ce chapitre, nous abordons l'état de l'art des supports de communication dans les systèmes multi-agents, et comment ces supports permettent la mise en oeuvre des communications multi-parties. Le problème principal se résume ainsi : quels vont être les récepteurs d'un message compte-tenu des besoins de l'ensemble des agents. Pour décrire cela, nous utilisons les critères introduits dans le chapitre 1 : la prise en compte des besoins de l'émetteur, la prise en compte des besoins des récepteurs potentiels, l'utilisation du contexte et la possibilité de contrôle.

Nous donnons en début de chaque section les caractéristiques générales du modèle, puis nous les affinons avec des propositions permettant de pallier les limites induites par ces caractéristiques pour les besoins de la mise en oeuvre des communications multi-parties.

2.1 Les communications directes : un support implicite

2.1.1 Caractéristiques générales

Les communications directes sont des communications point-à-point réalisées en utilisant les adresses des agents. Pour pouvoir communiquer avec un agent, il faut connaître ou être en mesure de récupérer cette adresse.

Les communications point-à-point ne nécessitent pas de support particulier pour être mises en oeuvre. L'émetteur a une totale maîtrise de ses communications. Par la richesse des langages de haut niveau, elles permettent également la mise en oeuvre de processus de négociation et de coordination. Une autre caractéristique des interactions directes est que les communications sont éphémères : une fois transmise par le médium, l'information n'est plus disponible.

La figure 2.1 résume les caractéristiques des communications directes. Seul l'émetteur de la communication choisit les récepteurs de ses messages, ce qui implique qu'un agent qui n'est pas connu individuellement par l'émetteur ne peut être récepteur. De même, leurs besoins ne peuvent être pris en compte que dans la mesure où l'émetteur connaît ces besoins.

Critères	Communications point-à-point
Emetteur	Seul décideur
Récepteur	Non pris en compte
Contexte	Contexte émetteur
Contrôle	Agent : aucun
	SMA : aucun

FIGURE 2.1 – Caractéristiques générales des communications directes

L'agent émetteur prend en compte ses besoins et son contexte pour choisir les récepteurs, il n'a par contre pas directement accès au contexte de transmission ou à celui du récepteur.

Du point de vue du contrôle, les agents ne peuvent choisir de prendre en compte ou non les messages qu'une fois que ceux-ci sont reçus, donc *a posteriori*, sauf à fermer leurs connexions avec les autres agents. Au niveau SMA, la communication entre les agents est faite directement, il n'est donc pas possible de contrôler les interactions à l'exécution.

2.1.2 Problématique des communications directes

Malgré ses avantages, il existe une double difficulté liée à l'interaction directe. Pour cela, il faut revenir aux objectifs de l'interaction, qui sont l'échange d'information et la coordination. Les informations et capacités sont distribuées entre entités distinctes. Les agents doivent donc être capable de mettre en oeuvre des mécanismes pour obtenir les informations dont ils ont besoin. Pour cela, ils doivent résoudre deux problèmes :

- Où est l'information ou la capacité recherchée ?
- Cette information est-elle à jour ?

Ces problématiques sont celles du problème de connexion, dont la définition est la suivante :

> *[...] trouver les autres agents possédant les informations ou les capacités dont vous avez besoin.* [8]
>
> [Davis et Smith, 1988]

L'un des pré-requis de la communication adressée est de savoir quel agent doit recevoir le message pour obtenir l'effet requis. L'émetteur doit être capable de choisir le ou les bon(s) récepteur(s), selon des critères différents en fonction de ses besoins. Les agents doivent donc avoir accès à une connaissance sur les autres agents.

Dans [Balbo et al., 2002], les auteurs identifient trois types de critères pour le choix des récepteurs dans le cadre de l'allocation multicritère de tâches : les critères d'objectif, les critères accessibles du contexte, et les critères non accessibles du contexte. Concernant les critères d'objectif, l'agent contacté doit posséder l'information ou la capacité recherchée. Dans ce cas, la formalisation de la demande permet en elle-même de trouver le récepteur, et le choix fait correspondre les agents aux demandes. Si plusieurs agents correspondent à la demande, l'agent doit pouvoir discriminer entre les candidats. Il s'agit de trouver le ou les meilleurs partenaires pour l'interaction. Pour cela, l'agent utilise des critères dépendant du contexte. Nous distinguons les critères du contexte accessibles à l'agent, par exemple relatifs à la demande (efficacité de l'agent, confiance), et les critères non accessibles qui font partie du contexte du récepteur, par exemple sa disponibilité.

La prise en compte de critères autres que l'adresse pour le choix des récepteurs est donc liée au problème de connexion, qui lui-même engendre le problème de temporalité de l'information. Hormis dans le cadre d'un système parfaitement statique, les informations comme les besoins d'information ont une durée de vie.

Les informations que l'agent possède sur les autres agents peuvent nécessiter des mises à jour. Dans un système ouvert, un agent peut apparaître, auquel cas les autres agents du système doivent pouvoir le découvrir. Il peut également disparaître, auquel cas il ne pourra plus être contacté. De plus, ses propriétés peuvent changer : acquisition d'une compétence, dégradation des performances. La mise à jour des informations sera d'autant plus difficile que le système sera dynamique, et le mécanisme de choix des récepteurs doit en tenir compte.

Les besoins des agents peuvent être ponctuels ou avoir une durée. Ceci implique des problématiques différentes. Dans le premier cas, le problème de connexion est renouvelé pour chaque besoin. L'agent doit donc pouvoir manifester ce besoin, y compris en fonction de caractéristiques qu'il ne connaît pas, par exemple si une information change mais qu'elle n'est pas accessible directement par l'agent. Dans le cas où le besoin a une durée, le possesseur de l'information doit connaître tous les agents intéressés par l'information. On pourra associer à ce cas celui d'un besoin ponctuel mais non immédiat, qui doit être connu pour le choix des récepteurs.

Ces difficultés impliquent de trouver un compromis au niveau de la mise à jour des informations de connexion, qui doivent être le plus à jour possible sans grever l'efficacité des agents, et

8. [...]finding the other agents that have the information or the capabilities that you need.

plus globalement du SMA.

Dans la section suivante, nous faisons un état de l'art des différentes propositions de résolution du problème de connexion, et comment elles permettent d'améliorer la flexibilité de l'adressage et la prise en compte du contexte.

2.1.3 La sélection des récepteurs

Différents modèles d'interaction directe ont été proposés dans la littérature. Ils présentent des solutions pour la sélection des récepteurs par l'agent émetteur. Nous étudions en particulier les mécanismes de découverte d'agents, ainsi que les critères de sélection des récepteurs. Les modèles sont regroupés en trois catégories, suivant que le choix est réalisé par diffusion ou protocoles, en fonction des connaissances internes de l'agent, ou en fonction d'une connaissance externe.

2.1.3.a Un cas particulier : la diffusion

Les caractéristiques générales de la diffusion sont résumées en figure 2.2. Le message étant envoyé à tous les agents, les besoins de l'émetteur sont implicitement pris en compte. Par contre, puisque ceux-ci sont implicites, les récepteurs peuvent ne pas en tenir compte ou ne pas être capables de les inférer. Le récepteur est chargé de trier les messages qui l'intéressent en fonction de ses besoins et de son contexte. Enfin, il n'y a aucun contrôle, puisque les récepteurs ne peuvent pas choisir *a priori* de refuser des messages, et qu'il n'y a pas de structure de contrôle.

Critères	Diffusion
Emetteur	Implicite
Récepteur	Choix parfait
Contexte	Contexte du récepteur
Contrôle	Agent : aucun
	SMA : aucun

FIGURE 2.2 – Caractéristiques générales de la diffusion

L'envoi de tous les messages à tous les agents par diffusion (ou *broadcast*) est le moyen le plus sûr de contacter tous les bons récepteurs. La diffusion peut être réalisée de deux manières : (i) soit elle est l'effet du médium, par exemple la plate-forme multi-agents ou les communications sans fil, (ii) soit elle est un choix des émetteurs, auquel cas il s'agit d'une répétition d'une communication adressée à tous les agents.

De cette façon, l'espace de recherche des récepteurs est complet, et les informations sont connues de tous. Le contexte ne rentre pas en jeu dans la transmission des messages, par contre les messages en tant que contexte du système sont effectivement perçus par les agents. L'intérêt des messages étant calculé localement par chaque agent, ceux-ci peuvent discriminer les messages selon leurs besoins. L'écoute flottante est donc intrinsèquement parfaite. Par contre, comme les récepteurs ne connaissent pas nécessairement les besoins initiaux de l'émetteur pour savoir leur

rôle dans la communication multi-partie, la diffusion est insuffisante pour en assurer le support complet.

Mais la principale limite de la diffusion de l'information est que son efficacité a un coût très important :

1. Le coût en bande passante : Chacun des messages doit être dupliqué proportionnellement au nombre d'agents, et chaque agent supplémentaire ajoute ses propres besoins en communications.
2. Le coût de traitement : Chaque agent recevant chaque message, il doit calculer pour chacun d'eux s'il est intéressé par l'information. Au pire, une communication n'ayant d'intérêt que pour un agent devra cependant être traité par tous les agents du système. Ceci induit une surcharge d'occupation qui sera elle aussi dépendante du nombre d'agents et du nombre de messages échangés. De plus, dans le cas où la diffusion est gérée par l'émetteur, il faut prendre en compte le temps d'envoi des messages pour chacun des récepteurs.
3. Le problème de l'adresse : Dans le cas où ce sont les agents qui se chargent d'envoyer les messages, un nouvel agent entrant n'a pas de moyen de connaître les agents pré-existants, et ces agents n'ont pas de moyens de le découvrir. Dans ce cas, la problématique liée à l'utilisation des adresses reste entière.

Dans le modèle LOTTO [Legras et Tessier, 2004], discuté en section 1.2.2, les auteurs utilisent la diffusion à tous les agents pour mettre en oeuvre l'écoute flottante. Les communications sont limitées par leur portée, ce qui permet de limiter le nombre d'agents écoutant les messages. Cependant, les auteurs ne donnent aucune indication sur le coût des communications ainsi réalisées.

Pour pallier le problème de coût de la diffusion, l'objectif des modèles décrits par la suite est de gagner en efficience, c'est à dire de garder la meilleure efficacité possible tout en baissant le coût à l'exécution.

La gestion par protocoles Les agents peuvent utiliser les protocoles d'interaction comme moyen de résoudre le problème de connexion [Bergenti et Ricci, 2002] ; l'exemple le plus connu et le plus utilisé est le protocole *contract-net* [Davis et Smith, 1988]. Il a été conçu pour la résolution coopérative de problèmes entre les agents, sur le modèle des mécanismes de tractation et d'appels d'offre utilisés dans le commerce.

Dans ce protocole, un agent prend le rôle de manager (parfois appelé gestionnaire) et les autres de contractants. Dans le protocole d'origine, les rôles ne sont pas pré-attribués, et n'importe quel agent ayant un problème à résoudre peut prendre le rôle de manager. Des contraintes sur les rôles peuvent être ajoutées par le concepteur en fonction du domaine.

Il y a un seul manager M pour n contractants C. Le protocole se déroule de la façon suivante :

1. M : Annonce d'une tâche à réaliser.
2. C : Evaluation de l'annonce.
3. C : Réponse par la soumission d'une proposition ou le refus.

4. M : Réception et évaluation des offres.
5. M : Allocation de la tâche à réaliser à un (ou des) contractant(s).
6. C : Réalisation de la tâche et rapport au manager.

De nombreuses variantes de ce protocole ont été proposées dans la littérature [Aknine et al., 2004; Lee et Chang, 1999; Russell et Zilberstein, 1993; Sandholm et Lesser, 1995] ; l'ajout le plus courant étant d'ajouter une date limite de réception des messages, afin de ne pas bloquer l'ensemble du processus en cas de non-réponse d'un des participants. Le protocole tel que défini par la FIPA est donné en figure 2.3.

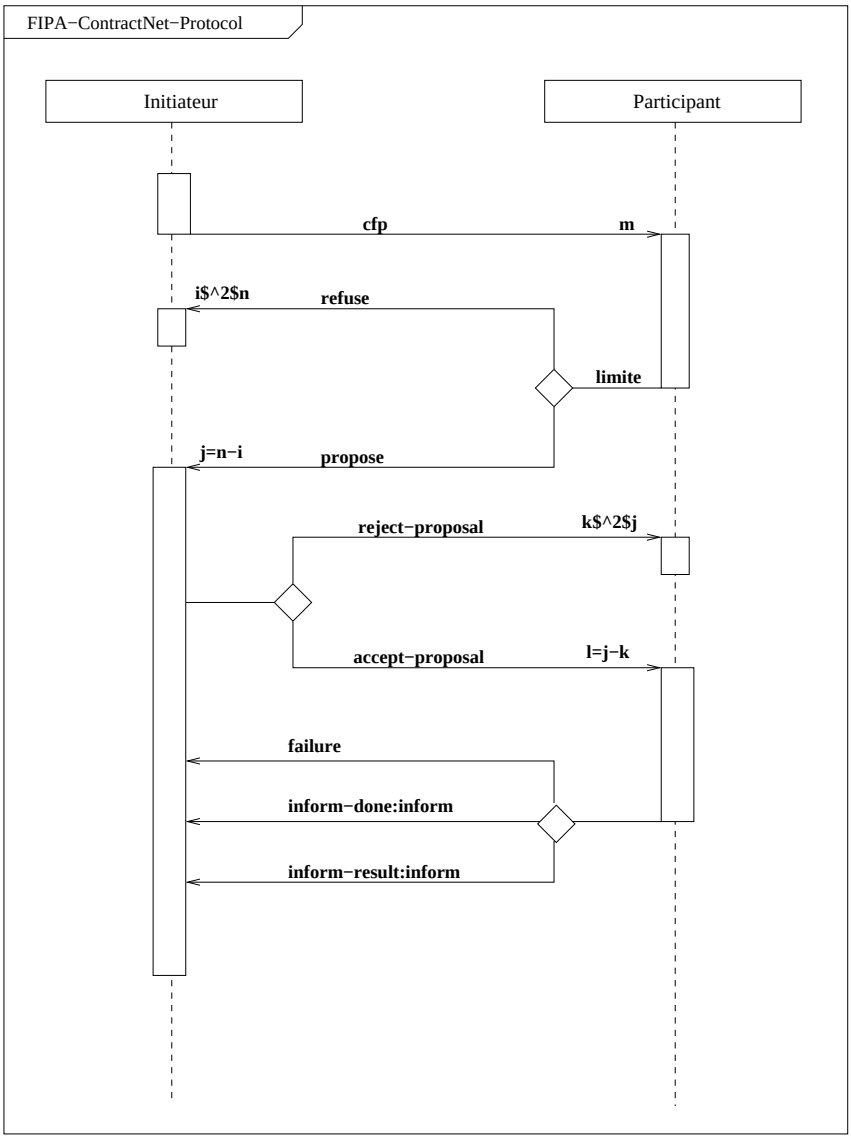


FIGURE 2.3 – Le protocole contract net.

Nous pouvons noter un double niveau de sélection de l'agent cible, le premier se situant

au niveau des étapes (1) (2) et (3), par la détermination des candidats potentiels, et le second lors des étapes (3) et (4) par le choix final en fonction de l'évaluation des propositions. La première sélection est opérée par les contractants, et la deuxième par le manager. La faculté de prendre n'importe quel rôle permet une redistribution par décomposition des tâches successives des agents. Une extension du protocole [Aknine et al., 2004] permet de pré-engager un agent dans plusieurs protocoles, ou de choisir d'autres contractants si celui choisi initialement ne donne pas de réponse ou une réponse insuffisante, pour limiter les coûts de désengagement.

La résolution de chaque nouvelle tâche implique une nouvelle sélection, ce qui fait du protocole contract-net une solution robuste. La disparition et l'apparition des agents dans le système multi-agents sont prises en compte à chaque nouvelle instance du protocole grâce à la diffusion de l'étape (1). Les mises à jour de propriétés sont effectuées immédiatement lors de la résolution distribuée.

Dans le cadre du problème de connexion, le mécanisme assure une seule utilisation de la diffusion complète (lors de l'annonce) pour chaque protocole. Le message d'annonce contient les critères de la tâche à remplir. Les contractants peuvent ainsi évaluer les propositions en fonction des critères de la demande et de leur propre contexte, ce qui assure que tous les agents potentiellement concernés sont en possession de l'information. Le choix final revient à l'émetteur, ce qui permet de prendre en compte une conjonction des besoins de l'émetteur et des récepteurs. C'est ce qui a fait son succès et sa facilité d'adaptation [Sandholm et Lesser, 1995].

Par rapport à une simple diffusion, le protocole contract-net permet donc une meilleure prise en considération des besoins et des contextes à la fois de l'émetteur et des récepteurs. Cependant, l'utilisation de la diffusion initiale implique toujours des problèmes de surcharge en bande passante et en traitement, comme le montre par exemple [Malville et Bourdon, 1998].

Hormis pour les systèmes multi-agents de faible taille, ou ne nécessitant que peu d'interactions, la diffusion complète n'est pas utilisable du fait de son coût important. Nous allons donc étudier les propositions permettant de réduire la diffusion à une sélection d'agents plus restreinte.

2.1.3.b Restreindre la diffusion grâce aux connaissances de l'agent

La première solution consiste à effectuer le choix des récepteurs de façon interne à l'agent. L'agent gère sa propre base de connaissances sur les autres agents et utilise ces informations pour effectuer ce choix. L'utilisation de réseaux d'accointances [Knoblock et al., 1994] illustre cette façon de procéder. Il s'agit d'utiliser les connaissances sociales de l'agent. Le terme d'accointance est également utilisé dans le cadre des réseaux pair-à-pair pour désigner l'ensemble des noeuds connus par un pair. Le principe est le suivant :

- L'agent possède une table d'accointances, regroupant ses connaissances sur les autres agents. Souvent, il ne connaît pas l'ensemble des agents mais seulement une partie de ceux-ci.
- Lorsqu'il a un besoin en communication, l'agent sélectionne les agents étant capable de

répondre à ce besoin.
- L'agent peut soit contacter tous les agents candidats, soit affiner sa sélection sur des critères autres que leurs capacités, grâce à ses connaissances et à son contexte.
- Ces calculs précèdent la mise en oeuvre d'un protocole contract-net. Les agents contactés répondent à cette sollicitation par une proposition, et l'émetteur effectue le choix final parmi les propositions.

Cette procédure de choix interne repose sur les connaissances de l'agent. L'organisation et le contenu de ces informations dépend des solutions proposées. Nous pouvons noter deux principaux types de classification des informations : une classification par compétence, que l'émetteur met en correspondance avec ses besoins, ou une classification inverse en fonction des besoins déclarés des agents.

L'avantage principal de ce modèle est de limiter les coûts de communication et de traitement corrélativement au nombre de récepteurs du message. Cependant, la contrepartie se situe au niveau du mécanisme de gestion des accointances et des données relatives à la sélection :

1. Dans un environnement ouvert, les connaissances sociales doivent bénéficier de protocoles de découverte des agents, ainsi que de mécanismes prenant en compte leur disparition. Chaque agent doit pouvoir répercuter ces modifications sur son propre réseau d'accointances.

2. Du point de vue des données elles-mêmes, des difficultés apparaissent au niveau du coût de mise à jour et au niveau de la nature des informations :
   - Les informations sont dynamiques et donc susceptibles d'êtres modifiées, ce qui a un impact sur le processus de sélection. L'ajout ou le retrait d'un service proposé, le changement de besoin d'un agent, doivent être propagés aux accointances pour que leurs connaissances soient à jour. Alors que les accointances diminuent le nombre de messages diffusés pour la réalisation des tâches, le processus de gestion dynamique des connaissances génère de nouvelles communications. Ce coût augmente en fonction du nombre d'agents appartenant aux accointances des autres agents et en fonction de la fréquence de mise à jour des informations. Au pire des cas, les agents communiquent donc plus pour mettre à jour leurs connaissances que pour réaliser leurs tâches. Si les mises à jour ne sont pas réalisées immédiatement, il y a un écart entre l'état courant du système et sa représentation, ce qui peut amener à des erreurs dans le choix du récepteur puisque celui-ci est fondé sur des informations ou besoins périmés.
   - L'adéquation des informations aux besoins en communication est complexe. D'une part, les agents ne diffusant pas leurs messages à l'ensemble des agents, la réponse aux besoins des agents dépendra des agents appartenant à son réseau d'accointances. Or, ce réseau ne contient pas tous les agents, ce qui diminue la robustesse du système. D'autre part, plus le processus de sélection des récepteurs est proche des besoins des récepteurs, plus il nécessite une quantité d'information importante sur celui-ci.

2.1.3.c Délégation de la sélection à une source externe

La résolution du problème de connexion peut être déléguée à un agent tiers. Dans [Sycara et Wong, 2000], nous trouvons une taxonomie des agents intermédiaires (*middle agents*). Ils sont utilisés pour assister les agents du système par la localisation et la mise en rapport du meilleur offrant avec le meilleur demandeur. Ce modèle est orienté-tâche, et s'articule autour de la correspondance compétence/besoin [Sycara et al., 2002]. Il s'agit en général d'une médiation de la recherche de contacts, et non d'une médiation de l'interaction elle-même.

La recherche d'information et de services, dans des systèmes ouverts, implique l'accès à des sources multiples, hétérogènes et distribuées. La solution proposée est donc de créer des intermédiaires, qui chercheront les meilleurs contacts par le biais d'annuaires descriptifs. Un agent ayant besoin d'entrer en contact avec d'autres agents n'a donc besoin de connaître que l'intermédiaire, lequel le met en relation - ou s'occupe lui-même - de la transaction souhaitée.

Le principe des agents intermédiaires est donc de connaître les services offerts par les fournisseurs, et de répondre aux demandeurs de la meilleure façon possible en choisissant selon la requête le ou les fournisseurs appropriés : il s'agit de la coordination fondée sur les capacités (*capability-based coordination*).

La gestion des informations est asymétrique entre les fournisseurs et les demandeurs. Les fournisseurs inscrivent et mettent à jour auprès des agents intermédiaires les services qu'ils peuvent offrir. On peut ensuite dissocier deux types de comportements, suite à la requête d'un demandeur :

- Les *matchmakers* renvoient l'adresse des services appropriés à la requête au demandeur, puis ce dernier contacte directement le ou les fournisseurs.
- Les *brokers* se chargent de demander le service à la place du demandeur, puis lui retransmettent le résultat une fois que le fournisseur a achevé sa tâche.

Ces comportements sont illustrés en figure 2.4 et 2.5.

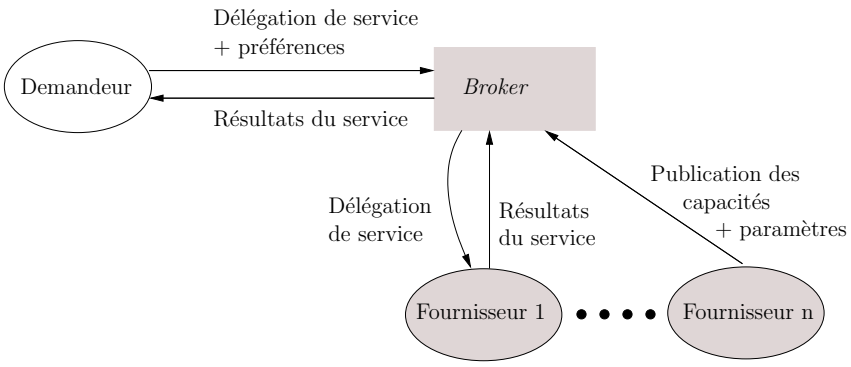


FIGURE 2.4 – Agent broker [Sycara et Wong, 2000]

Un exemple de *broker* couramment utilisé est celui des méta-moteurs de recherche sur internet, qui permettent à l'utilisateur d'effectuer une requête unique. Elle est ensuite gérée par le

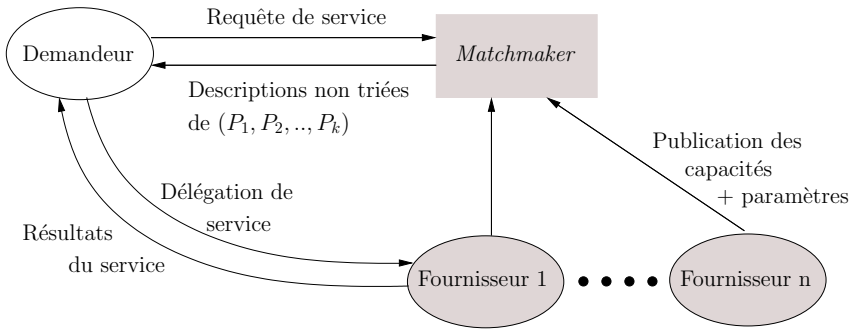


FIGURE 2.5 – Agent matchmaker [Sycara et Wong, 2000]

méta-moteur, lequel contacte et utilise les différentes sources avant de retourner le résultat au demandeur.

Les agents intermédiaires sont utilisés de trois façons, suivant la manière dont sont décrits les agents ou services :

- Pages blanches : Les adresses sont classées en fonction de l'identité des services.
- Pages jaunes : Les services sont classés suivant des taxonomies standardisées.
- Pages vertes : Les services sont décrits par un ensemble de méta-informations.

Le concept d'agent intermédiaire est aussi utilisé dans le cadre d'agents "avatars", capables d'intégrer, manipuler, rechercher et analyser des informations sur l'Internet, directement ou indirectement pour l'utilisateur [Daniel et al., 2005].

Du point de vue du coût en communication, l'avantage par rapport au stockage interne des informations est que la mise à jour doit uniquement être réalisée auprès des agents intermédiaires. Le modèle facilite également la tâche de recherche du récepteur. Par rapport aux réseaux d'accointances, les agents intermédiaires possèdent des connaissances sur l'ensemble des agents, ce qui permet une recherche plus exhaustive.

Les deux limites de ce modèle sont liées à la centralisation des informations et à la méthode de choix du récepteur. Du point de vue de l'adressage, ce modèle nécessite une connaissance des agents intermédiaires par tous les agents. Or, tous les messages étant traités par un petit nombre d'agents intermédiaires, il peut se créer un engorgement dans la résolution des problèmes de connexion, du fait des coûts de traitement et/ou de bande passante. Augmenter le nombre d'agents intermédiaires crée soit une redondance accrue, soit une fragmentation de l'information. Le service de pages blanches ne permet pas de mettre en relation un besoin et un fournisseur, et le service de pages jaunes limite la recherche aux seules compétences. Le service de pages vertes est le plus complet, mais il ne permet pas d'évaluer directement le contexte du récepteur.

Notons que lorsque la requête n'est pas ponctuelle mais persistante, le système se rapproche des mécanismes *Publish and Subscribe*, étudiés en section 2.2.3. De plus, certaines plate-formes comme RETSINA [Sycara et al., 2003] et JADE [Bellifemine et al., 2001] proposent un système de découverte par pages blanches et pages jaunes ; ce système est préconisé par la standardisation

FIPA (section 2.2.1).

2.1.4 Conclusion

Les communications directes sont une composante des communications multi-parties, recouvrant les cas où le choix du récepteur est effectué sur l'initiative de l'émetteur et où ses correspondants sont connus individuellement.

Le support des communications multi-parties par la diffusion est irréaliste dès que la taille du système devient importante ou que le nombre de communications augmente, tant au niveau du réseau que de la charge computationnelle supportée par les agents. De plus, les besoins et le contexte de l'émetteur doivent être explicitement définis dans les messages pour que les récepteurs puissent les prendre en compte.

Restreindre le nombre de récepteurs par la connaissance, interne ou externe, permet de dépasser cette contrainte ; par contre, les besoins des récepteurs ne sont pris en compte que dans la mesure où l'émetteur les connaît et les applique. A cet égard, la gestion par protocole de type contract-net et les agents intermédiaires permettent une mise en correspondance des besoins des agents. Cette mise en correspondance est orientée-tâche, ce qui signifie que c'est l'intersection des demandes et propositions qui crée le lien entre émetteur et récepteur(s). Si l'objectif est non seulement la réalisation de tâches, mais aussi la propagation de l'information aux agents intéressés, l'objectif sera de transmettre le message non plus à l'intersection mais à l'union des agents choisis par l'émetteur et des agents intéressés par ce message.

Avec les communications directes, le contrôle des interactions à l'exécution s'avère impossible tant au niveau agent qu'au niveau multi-agents. L'agent ne peut pas choisir de refuser un message, sauf à fermer sa connexion avec les autres agents. L'absence d'infrastructure au niveau multi-agents empêche le contrôle *a priori* des transmissions : comme nous l'avons vu en section 1.4.3, la seule solution pour mettre en oeuvre cela est la médiation de la communication.

2.2 Médiation des communications

Les modèles vus en section 2.1 ne nécessitent pas d'infrastructure d'interaction puisque le seul pré-requis est la capacité des agents à communiquer, laquelle peut être incluse dans l'architecture de l'agent.

Pour simplifier la conception des systèmes multi-agents, de nombreux travaux proposent de déléguer une partie des tâches en dehors de l'agent, au sein d'une structure spécifique. Cette structure prend alors en charge un certain nombre de services de base, les deux plus courants étant le cycle de vie des agents (exécuter et terminer les processus des agents) et la communication.

Nous nous intéressons dans cette section aux différents types de structures existantes du point de vue de la communication, en suivant les mêmes critères que précédemment, et notamment la façon dont les récepteurs sont sélectionnés.

2.2.1 La standardisation FIPA

La première approche pour simplifier la conception et l'interopérabilité dans les systèmes multi-agents est l'adoption d'une standardisation. La spécification de référence pour une infrastructure d'interaction est celle de la FIPA [FIPA, 2002a] (illustrée en Fig. 2.6).

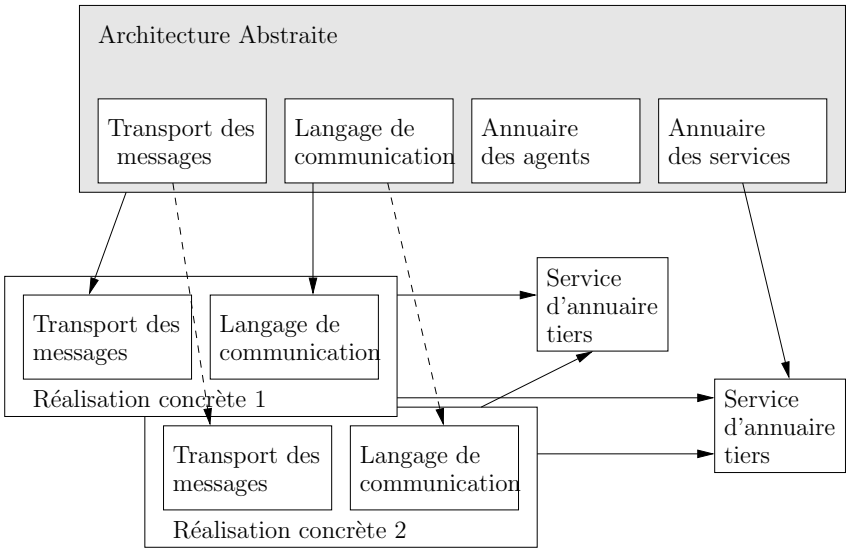


FIGURE 2.6 – Architecture FIPA

Elle définit le modèle de référence d'une plate-forme multi-agents, et notamment les rôles des agents clés nécessaires à la gestion de la plate-forme, ainsi que le contenu du langage de communication des agents.

Les rôles principaux sont au nombre de trois :

- Le système de gestion d'agents, qui contrôle l'accès et l'usage de la plate-forme par les agents, et fournit un service de pages blanches.
- Le canal de communication, qui officie pour le routage des interactions de base intra et extra plate-forme.
- Le facilitateur d'annuaire, qui fournit un service de pages jaunes à la plate-forme.

Le langage de communication agent préconisé est FIPA-ACL. Pour faciliter les interconnexions entre plate-formes, FIPA spécifie un transfert de message sous forme textuelle, en supposant que l'agent est capable de transmettre cette forme. Parmi les plate-formes supportant l'architecture FIPA, nous pouvons citer JADE, ZEUS, et FIPA-OS [9].

Notons que cette architecture est indépendante du modèle utilisé, ainsi que du langage d'implémentation. Son objectif est essentiellement l'interopérabilité et la facilitation des primitives de gestion des agents, notamment par l'incorporation d'une architecture d'annuaire à la façon des agents intermédiaires. Elle partage les forces et faiblesses des agents intermédiaires pour le support des communications multi-parties : la centralisation des informations au ni-

9. Présentation et références en Annexe B

veau de l'annuaire facilite la gestion des informations, mais les communications restent dans le paradigme des interactions directes.

Les spécifications FIPA sont une base sur laquelle il est possible d'ajouter des éléments, ainsi certaines plate-formes permettent la gestion de structures organisationnelles, dont nous voyons un exemple en section suivante.

2.2.2 Le modèle organisationnel

Il est possible de déléguer le processus de sélection des récepteurs à l'organisation du SMA. Il s'agit de regrouper les agents au sein d'une structure formelle qui possède un ensemble de règles. Un exemple de gestion structurelle par modèle d'organisation est le modèle Agent-Groupe-Rôle (AGR) [Ferber et Gutknecht, 1998].

Aalaadin, la méthodologie liée à AGR, inclut deux niveaux conceptuels : le niveau concret, avec les notions d'agent, groupe et rôle, et le niveau abstrait, décrivant les interactions valides et la structure des groupes, de façon totalement indépendante de l'architecture des agents. Un agent peut appartenir a plusieurs groupes, au sein desquels il possède un rôle. Le modèle AGR permet d'identifier les agents par le biais des groupes auxquels ils appartiennent, et de leurs rôles dans chacun de ces groupes. Nous parlerons alors d'*adressage logique*, lorsque le récepteur est identifié par d'autres moyens que son adresse physique ou son identifiant.

La plate-forme MadKit [10] met en oeuvre les concepts du modèle AGR en proposant la gestion de cette organisation au niveau de l'infrastructure. Il est ainsi possible d'obtenir tous les agents d'un groupe, ou d'effectuer une diffusion à tous les agents ayant un rôle dans un groupe. La gestion des groupes et des rôles, ainsi que la transmission des messages sont gérées par la plate-forme. La connaissance sociale est composée des caractéristiques Agent/Groupe/Rôle et c'est par ce biais que la sélection des récepteurs est effectuée. La gestion de cette connaissance étant déléguée à la plate-forme, tous les agents d'un même groupe ont accès à la même connaissance. La plate-forme est également responsable de l'inscription lorsque l'agent entre et sort du système.

Du point de vue de la robustesse, l'adressage par le biais des rôles permet de contacter un rôle indépendamment des agents réels, ce qui assure un suivi dynamique en cas d'ajout, disparition ou remplacement d'un agent. De la même façon que pour les agents intermédiaires, le mécanisme de mise à jour des informations est centralisé. L'avantage est donc un coût réduit par rapport à une gestion individuelle, puisque seule la plate-forme doit être informée des modifications. Enfin, la plate-forme vérifie l'adéquation des messages envoyés avec la structure organisationnelle : un agent ne peut communiquer qu'avec les agents du même groupe.

Dans le cadre des communications multi-parties, nous voyons cependant deux limites à cette approche :

– L'adéquation de l'organisation par groupes et rôles dépend fortement de l'application. Les agents doivent être capables de raisonner sur les méta-structures que sont les groupes et rôles, et pouvoir formuler leurs besoins en ces termes, de façon cohérente entre émetteurs

10. www.madkit.org

et récepteurs. De plus, le choix de l'adressage logique reste au niveau de l'émetteur, et les informations ne peuvent dépasser celles de la structure. Autrement dit, on ne peut utiliser d'autres critères pour fonder l'adressage que celles d'adresse, de groupe et de rôle.

– La taille des groupes et le nombre des rôles ne peuvent être prévus *a priori*. La centralisation de l'information peut être source de congestion du fait de la gestion par listes des groupes et des rôles.

2.2.3 Les modèles Publish/Subscribe

Le principe des modèles *Publish and Subscribe* est celui du choix des récepteurs par un ou plusieurs serveur(s) connaissant les besoins en information des agents. Une vue d'ensemble de ces modèles est proposée dans [Eugster et al., 2003]. Les besoins des agents sont caractérisés par un ensemble de conditions sur le thème (*topic-based*), le contenu (*content-based*) ou le type (*type-based*) du message. Ce modèle a un grand succès dans les applications Internet par sa capacité à traiter un nombre important de messages pour un nombre important d'inscrits. Son utilisation dans les systèmes multi-agents est limitée, et est en général liée aux espaces partagés que nous étudions en section suivante.

Critères	Publish/Subscribe
Emetteur	Non pris en compte
Récepteur	En fonction du thème, contenu ou type
Contexte	Informations du serveur
Contrôle	Agent : aucun
	SMA : possible

FIGURE 2.7 – Caractéristiques générales du *Publish and Subscribe*

Un résumé des caractéristiques du modèle *Publish and Subscribe* est donné en figure 2.7. Les besoins des émetteurs ne sont pas exprimés au niveau du serveur et ne sont donc pas pris en compte. Le serveur choisit les récepteurs uniquement en fonction des besoins que ceux-ci ont exprimé en s'inscrivant.

Le contrôle à l'exécution est réalisable sans difficulté au niveau des serveurs, tandis qu'au niveau agent il est inutile puisqu'ils peuvent modifier leurs conditions de réception et qu'il ne sont pas contactés directement par les émetteurs.

L'appariement entre messages et récepteurs détermine les contraintes du contexte pouvant être utilisées. Le modèle *publish/subscribe* étant fondé sur la notion d'évènements, seuls des critères sur les messages sont utilisés : les modèles fondés sur les thèmes permettent l'émission et la réception de messages par le biais de canaux, comme le *Channeled Multicast* vu en section 1.2.2. Ceux fondés sur le contenu prennent en compte toute opération logique sur les champs constituant le message. Enfin, les modèles fondés sur le type du message utilisent une notion hiérarchique de la typologie des messages de façon à étendre la notion de canal. Ainsi, quel que

soit le modèle choisi, une des limites est que les contextes de l'émetteur et des récepteurs ne sont pas pris en compte. Concernant le contexte de transmission, il existe quelques travaux prenant en compte des méta-informations telles que la qualité des réseaux pour modifier le routage des messages [Huang et Garcia-Molina, 2001; Zieba et al., 2005]. Cependant, ces méta-informations sont fixées par le concepteur et ne sont donc pas traitées de façon générique.

2.2.4 Les espaces partagés

2.2.4.a Caractéristiques générales

Les espaces partagés sont des localisations de mémoire dans lesquelles des processus (programmes, agents) peuvent écrire et lire des informations. Les espaces de mémoires peuvent être physiques, *i.e.* correspondre à une adresse de stockage réelle, ou logiques, *i.e.* correspondre à un point d'accès à des ressources distribuées. Du point de vue de l'acte de communication, cela signifie un découplage dans le temps et dans l'espace, puisqu'il n'y a pas besoin de synchronisation entre l'émetteur et le lecteur de l'information, ni même besoin qu'ils connaissent leurs adresses respectives.

La transmission d'une information par le biais d'un espace partagé est une interaction indirecte. La figure 2.8 résume les caractéristiques des espaces partagés.

Critères	Espaces partagés
Emetteur	Non pris en compte
Récepteur	Système de motifs
Contexte	Contexte du récepteur, données partagées
Contrôle	Agent : oui
	SMA : possible

FIGURE 2.8 – Caractéristiques générales des espaces partagés

Nous utilisons le terme de "récepteur" à la place de "lecteur" de l'information par cohérence avec la notion de transfert d'information, et de façon à expliciter le lien avec les communications adressées et multi-parties.

Le découplage entraîne, pour l'émetteur de l'information, une perte totale de choix du récepteur : chaque agent est libre de lire ou non les informations mises à disposition. Par miroir, le récepteur possédant la localisation de l'espace partagé peut accéder aux communications selon ses besoins, par un système de motifs que nous explicitons en section 2.2.4.c.

Le récepteur peut utiliser ses connaissances du contexte pour orienter sa recherche d'informations ; de plus, chaque information du contexte présente au sein même de l'espace partagé peut être obtenue par les agents. Cette information peut avoir été collectée par un serveur ou ajoutée par les agents.

La notion de contrôle au niveau agent est inadéquate, puisque celui-ci est proactif vis-à-vis de l'information. Au niveau multi-agents, il est possible de mettre en oeuvre des mécanismes de

contrôle pour réglementer l'accès aux espaces partagés (section 2.2.4.f).

2.2.4.b Les tableaux noirs

Ce modèle est fondé sur la métaphore du tableau noir, à la façon d'un groupe d'experts écrivant sur ce tableau et partageant leurs conclusions pour que chacun des experts puisse y accéder. Une collection d'articles sur les systèmes à base de tableaux noirs peut être trouvée dans [Engelmore et Morgan, 1988].

Les tableaux noirs ont d'abord été utilisés dans le cadre de systèmes de grande taille découpés en modules, éventuellement distribués. Le principe de base est la présence d'un espace commun de mémoire partagée. Le fonctionnement est le suivant :

- Un ou plusieurs tableaux noirs contiennent les informations, généralement regroupées en niveaux afin d'améliorer la lisibilité et de diminuer la complexité de la recherche d'information.
- Un système de contrôle décide d'activer les modules en fonction des informations disponibles sur le tableau.
- Les modules, encore appelés "sources de connaissances", se chargent de traiter l'information et d'inscrire le résultat de leurs inférences sur le tableau.

Ce système est incrémental, puisqu'il permet de se rapprocher de la solution recherchée à l'aide de multiples sources délocalisées. Par ailleurs, la décomposition du système en sous-tâches indépendantes permet de diminuer la complexité de chacune des sources de connaissance.

Les systèmes à base de tableaux noirs ont été un premier pas vers les systèmes multi-agents, car on retrouve ici la nécessité de décomposer les compétences afin d'atteindre un but commun, en partageant des informations par le biais d'un espace connu par tous les agents.

Il est à noter que ce principe est directement utilisable dans le cadre multi-agents par le remplacement des modules par des agents, mais que cela contredit le principe d'autonomie : les agents doivent pouvoir déclencher par eux mêmes des interactions avec l'espace partagé.

Dans le cadre des SMA, le problème n'est plus de coordonner les agents par un contrôleur, mais de faciliter la transmission de l'information, chaque agent ayant un accès à l'espace partagé. Par rapport aux communications adressées, le problème fondamental n'est pas de savoir où est l'information, mais est de savoir comment la discriminer de façon satisfaisante. Si les agents accèdent de façon systématique aux informations déposées dans l'espace partagé, cela implique une complexité équivalente à la diffusion totale, tant au niveau des communications que du traitement des informations.

Une deuxième difficulté des modèles d'espaces partagés est la prise en compte du contexte. L'émetteur ne sait pas quels sont les agents lisant ses messages, de la même façon que le récepteur ne connaît pas nécessairement l'émetteur, ni à plus forte raison son contexte si cette information n'est pas insérée dans les données échangées. Ceci implique que la notion de communication adressée et les mécanismes qui en découlent, comme les protocole multi-agents, ne sont pas applicables.

Dans les sections suivantes, nous étudions comment la recherche d'information est traitée, avec les modèles à base de tableaux noirs à la Linda [Carriero et al., 1986], ainsi que les middlewares [Freeman et al., 1999; Lehman et al., 2001]. Ensuite, nous abordons plus précisément les questions du contexte et du contrôle dans les espaces partagés.

2.2.4.c Les espaces de tuples

Linda [Carriero et al., 1986] a initié les modèles de coordination orientés-données. Ces modèles ne sont pas spécifiques aux systèmes multi-agents mais destinés à faciliter les échanges d'information pour les systèmes distribués. Le modèle Linda propose une mémoire partagée nommée *espace de tuples*, ainsi qu'un mécanisme de récupération des données utilisant leurs signatures. Nous présentons tout d'abord les mécanismes introduits par le modèle Linda, puis le modèle LIME [Picco et Buschini, 2002] étendu à un environnement mobile, et enfin le modèle TuCSoN [Omicini et Zambonelli, 1999] qui permet de programmer le comportement de l'espace.

Linda. Dans le modèle Linda, un tuple est défini comme une suite ordonnée d'éléments typés, par exemple $<$ "$inform$", $0.2, 4, 5 >$. Un motif (*template*) est un tuple particulier dans lequel les champs sont typés, mais peuvent ne pas avoir de valeur définie. Il y a correspondance entre un tuple et un motif lorsqu'ils ont le même nombre d'éléments, et que leurs types et valeurs sont égaux deux à deux. Un motif peut être apparié avec plusieurs tuples. Le tuple précédent correspond par exemple au motif $<$ "$inform$", $float, integer, 5 >$.

Les tuples sont stockés dans un espace de tuples, lequel propose les primitives suivantes pour les manipuler :

- $out(t)$: ajout du tuple t
- $in(m)$: lecture d'un tuple correspondant au motif m et retrait de ce tuple
- $read(m)$: lecture d'un tuple correspondant au motif m, pas de modification de l'espace de tuples

Les primitives *in* et *read* sont bloquantes : si aucun tuple ne correspond, le processus appelant est suspendu jusqu'à ce qu'un tuple correspondant arrive dans l'espace, ce qui permet de synchroniser les processus. Si plusieurs tuples correspondent à une primitive, un seul de ces tuples est choisi au hasard. La communication par espace de tuples est dite générative : un processus génère un tuple dont le cycle de vie est indépendant du processus qui l'a créé.

Dans le cadre des SMA, nous pouvons noter plusieurs difficultés induites par ce modèle de base :

- Les informations sont centralisées dans l'espace de tuples.
- L'agent doit être proactif par rapport à l'information : il ne reçoit pas automatiquement les informations déposées dans l'espace.
- Les primitives sont bloquantes, et risquent donc de restreindre l'autonomie de l'agent. L'exécution de l'agent est suspendue lors des opérations de lecture dans le cas d'un agent mono-processus, où la lecture des messages est effectuée par le processus gérant le cycle

de l'agent.
- Le format de représentation des données ne contient pas *a priori* d'informations sur le contexte d'émission du tuple, sauf à ajouter ces informations dans le tuple message ou dans un autre tuple.
- La discrimination des tuples par le biais de motifs est peu expressive. En effet, les seuls opérateurs de comparaison sont l'égalité et la correspondance de type, sur un seul tuple. Il est impossible de prendre en compte une correspondance sur plusieurs tuples simultanément.

De nombreux modèles de coordination ont proposé des extensions du modèle Linda. En particulier, nous pouvons noter l'ajout de primitives, par exemple la possibilité de lire plusieurs tuples, et la modification de la sémantique des primitives. Nous allons décrire plus en profondeur trois variantes : LIME dont l'objectif est de gérer un environnement mobile, TuCSoN qui permet d'ajouter une couche de programmation pour la coordination, et le middleware JavaSpaces.

Lime. Les principes de base de LIME (*Linda In a Mobile Environment*) sont ceux de Linda, dans un contexte mobile [Picco et al., 1999]. Ainsi, au lieu de communiquer par le biais d'un espace de tuples centralisé, chaque agent possède son propre espace de tuples. Les primitives sont améliorées grâce à la lecture et au retrait en mode non-bloquant. L'originalité de cette approche est que, lorsque plusieurs agents sont connectés, soit en étant sur la même machine hôte soit par une connexion réseau, leurs espaces de tuples fusionnent. La fusion est réalisée de manière transparente, en donnant l'illusion que tous les tuples appartiennent au même espace. Ceci permet l'utilisation de LIME lorsque les agents et/ou les machines sont mobiles, et que leurs interconnexions ne sont pas permanentes. Ceci est réalisé grâce à un paramètre supplémentaire des primitives, qui permet de spécifier quel espace de tuples doit être modifié par l'opération.

Enfin, un aspect particulièrement intéressant est la possibilité de programmer des réactions, qui sont des parties de code exécutées lorsque certains tuples sont insérés dans l'espace de tuples. Cette possibilité est développée dans le modèle TuCSoN.

TuCSoN. Comme le modèle LIME, TuCSoN (*Tuple Centres Spread over Networks*) [Omicini et Zambonelli, 1999] permet de distribuer les tuples et de les partager entre plusieurs espaces de tuples. Les primitives sont identiques à celles de LIME. La principale originalité réside dans le langage ReSpecT (*Reaction Specification Tuples*), qui permet de définir des comportements en réponse à des évènements de communication. Ces évènements correspondent à l'utilisation d'une des primitives du modèle. L'avantage est de pouvoir modifier les comportements et les règles de communication au niveau de chaque espace de tuples local.

Les réactions sont un ensemble d'opérations non-bloquantes produisant des effets sur l'espace de tuples. Elles ont accès aux tuples, mais aussi aux informations sur l'évènement déclencheur, par exemple la primitive qui a déclenché la réaction et le processus l'ayant appelée. Les réactions peuvent se déclencher entre elles, et sont gérées de façon atomiques. Enfin, ReSpecT est expressif, le langage ayant été prouvé comme étant turing-complet [Denti et al., 1998].

TuCSoN répond donc en partie aux critiques de LINDA pour une utilisation agent, grâce à la prise en compte des évènements. Par ailleurs, la programmation des réactions peut permettre de composer les opérations de lecture, pour affiner les recherches d'information. Cependant, manipuler les réactions requiert de programmer l'espace de tuples de façon à répondre aux besoins de chaque agent. Enfin, la problématique de l'expressivité des tuples dans le contexte agent n'est pas abordée par les auteurs de ce modèle.

2.2.4.d Les middlewares

Le principe de JavaSpaces [Freeman et al., 1999] est la création d'espaces communs d'échanges d'objets du langage de programmation JAVA, indépendants du contenu de ces objets, de façon distribuée. La principale différence avec les systèmes précédents correspond au type des données, qui sont ici des objets et non des tuples.

L'infrastructure de communication de JavaSpaces unifie les échanges grâce un système d'espace de transaction commun, les messages étant transférés de la même façon que tout type d'objet, dans un univers distribué et persistant.

Sont adjoints à ces espaces les primitives d'utilisation suivantes :

- *write* : écrire une entrée dans un espace JavaSpaces
- *read* : lire une entrée qui correspond à un template donné
- *take* : lire une entrée qui correspond à un template donné, et la retirer de l'espace.
- *notify* : notifier un certain objet quand des entrées qui correspondent à un template donné sont écrits dans le service JavaSpaces.

Il existe des requêtes *readIfExists* et *takeIfExists*, qui fonctionnent de la même manière que read et write, mais en mode non-bloquant.

Les T-Spaces [Fontoura et al., 2003] permettent l'ajout dynamique de nouvelles opérations, ainsi que la mobilité des agents. Contrairement à la programmation d'espaces de tuples à la façon de TuCSoN, où le langage ReSpecT utilise les primitives pré-existantes, il est ici possible d'ajouter des primitives pour manipuler directement les tuples. Or, une nouvelle primitive peut modifier le modèle de coordination lui-même. Si l'avantage est que les agents peuvent modifier leurs accès aux tuples, il est difficile de mettre en place des règles communes, ce qui implique un mécanisme de notification des opérations existantes et une adaptabilité des agents à ces nouvelles règles.

Plutôt que modifier les primitives ou les composer dans des réactions, une façon de rendre plus flexible l'obtention des informations du middleware (ou de l'espace de tuples) est d'augmenter l'expressivité du motif passé en paramètre de l'acte de lecture. Nous présentons dans la section suivante deux modèles qui permettent de prendre en compte le contexte de cette manière.

2.2.4.e Le contexte dans les espaces de tuples

Dans le modèle de partage de données, nous avons évoqué que le contexte peut être représenté directement sous forme de tuples. Ce modèle de données n'est pas expressif en soi, puisqu'il

s'agit d'une collection de valeurs. Pour pouvoir utiliser le contexte, deux caractéristiques sont nécessaires : l'observabilité (comment obtenir l'information) et la dynamique (comment gérer la variation du contexte).

Ainsi, EgoSpaces [Julien et Roman, 2004] gère le contexte sous forme de données, et introduit la possibilité de poser des contraintes multiples sur les données observées. Ces contraintes concernent les agents, les données et le réseau lui-même, par exemple "les données de type requête de tous les agents dans les gymnases à moins de 200m". Ce modèle d'observabilité est plus expressif que les modèles classiques, mais la principale limite est qu'il ne permet pas d'apparier les valeurs des données entre elles. Les contraintes sont données sous forme de test sur une valeur ou un type, de la même façon que les templates. Il n'est donc pas possible de lire un tuple en fonction de la valeur d'un autre. Du point de vue de la dynamique, les agents ont deux moyens de collecter des données changeantes, soit en demandant périodiquement le résultat d'une requête, soit par notification, auquel cas l'ajout d'un tuple est notifié à tous les agents ayant inscrit un template correspondant.

Dans [Schelfthout et al., 2006], les auteurs proposent d'utiliser la notion de *vues* sur les espaces partagés, qui sont des contraintes de réception. De la même façon qu'EgoSpaces, le récepteur de la donnée choisit la portée de la vue sur le réseau d'ordinateurs. Les vues actives notifient les agents des ajouts dans l'espace de tuples, et les auteurs ajoutent le maintien des informations en cas de disparition des objets. L'avantage de ce modèle est la possibilité de déterminer la façon dont les données sont traitées avant d'être obtenues, par exemple pour obtenir des listes triées d'objets correspondant au template. Par contre, de même que pour TuCSoN, il ne permet pas *a priori* l'appariement entre éléments contenus dans l'espace de tuples, hormis dans le cadre de la programmation des réactions aux évènements.

L'aspect notification de ces modèles peut être vu comme un modèle *publish and subscribe*, où l'infrastructure elle-même prend une part active dans la communication.

2.2.4.f Le contrôle des opérations dans les espaces de tuples

Deux moyens ont été introduits pour assurer le contrôle des accès dans les espaces de tuples, de façon implicite ou explicite : les espaces multiples et l'utilisation de lois.

Espaces multiples Plusieurs raisons ont motivé l'introduction de plusieurs *tuplespaces* au lieu d'un seul. L'une d'entre elles est de répondre au besoin de sécurité des données en les plaçant dans des espaces de tuples distincts, et en limitant le nombre d'agents ayant accès à chacun d'eux. Ceci restreint la visibilité des données, puisqu'une action de lecture depuis un espace de tuple n'a accès qu'aux données qui y figurent.

Avec l'utilisation d'espaces multiples, la sécurité est lié à la limitation du partage de données, ce qui est en contradiction avec l'objectif des espaces de tuples.

Law Governed Linda (LGI) Dans [Minsky et al., 2001], les auteurs observent qu'isoler la communication dans des espaces multiples n'augmente la sûreté que dans la mesure où cela

élimine le partage, alors que les espaces de tuples sont plus utiles quand plusieurs agents partagent l'accès aux espaces. Ainsi, Minsky *et al.* argumentent que, comme la communication dans Linda se fait par contenu, la sécurité doit aussi être assurée par contenu. Ils présentent les lois, un mécanisme qui utilise le contenu des tuples afin de savoir si l'opération peut réussir. De cette façon, la sécurité est satisfaite tout en tirant profit du partage des données.

2.2.5 Les modèles d'environnement

La notion d'environnement explicite a longtemps été associée au paradigme des agents réactifs, mais des travaux récents [Platon et al., 2007a; Valckenaers et al., 2007; Viroli et al., 2007; Weyns et al., 2007] ont montré les avantages de l'exploitation de cette abstraction dans le cadre général des SMA. Ces travaux, et en particulier ceux de Weyns *et al.*, mettent en exergue la délégation de la gestion d'un certain nombre de "responsabilités" des agents vers l'environnement. En particulier, l'environnement peut prendre en charge le support de la perception et de la communication.

La communauté E4MAS (*Environment For Multi-Agent Systems*[11]) a proposé la définition suivante de l'environnement :

> **Environnement**[12]- L'environnement est une abstraction de premier ordre qui :
> – fournit les conditions pour que les agents existent,
> – effectue la médiation des interactions entre agents et l'accès aux ressources.
>
> [Weyns et al., 2007]

Dans cette définition, nous pouvons observer une double notion d'aide à la modélisation du système et d'infrastructure de support. De façon plus précise les responsabilités suivantes ont été identifiées dans [Weyns et al., 2005] comme pouvant être confiées à l'environnement :

- **Structuration**
  L'environnement permet de structurer le SMA aussi bien physiquement que socialement, et prend en charge les communications.
- **Support et gestion du cycle de vie**
  L'environnement est en charge de la maintenance des ressources et agents, ainsi que de leur dynamique.
- **Fourniture de l'observabilité**
  L'environnement fournit les structures de représentation, ressources et "corps" [13] des agents.
- **Fourniture de l'accessibilité**
  L'environnement gère l'accessibilité aux structures et aux ressources.

11. E4MAS a été l'acronyme d'un groupe de travail Agentlink III, ainsi que d'un workshop lié à la conférence AAMAS (International Conference On Autonomous Agents and Multiagent Systems). La communauté poursuit ses travaux dans le cadre d'EEMMAS (Engineering Environment-Mediated Multiagent Systems).

12. The environment is a first-class abstraction that provides the surrounding conditions for agents to exist and that mediates both the interaction among agents and the access to resources.

13. Le corps logiciel, voir par exemple les travaux de Platon *et al.* en section 1.2.3.

– **Régulation du SMA**
  L'environnement contrôle les accès aux structures et aux ressources, et maintient les lois d'accès.
– **Ontologie**
  L'environnement fournit une ontologie commune qui peut être consultée par les agents

L'environnement étant une abstraction utilisée lors de la modélisation du système multi-agents, sa mise en oeuvre est réalisée par le biais d'un ou plusieurs supports de communication vus au cours de ce chapitre. Nous orientons notre étude sur les concepts liés à la communication (figure 2.9).

Critères	Environnement
Emetteur	Communication
Récepteur	Perception
Contexte	Oui
Contrôle	Agent : possible
	SMA : oui

FIGURE 2.9 – Caractéristiques générales des modèles d'environnement

Les environnements permettent de mettre en oeuvre des mécanismes de diffusion et de perception d'informations. Ces mécanismes sont séparés des modules de communication : c'est la dichotomie entre les communications vues dans leur dimension classique, à la façon de la spécification FIPA, et celles issues de la gestion de mémoire partagée. Ainsi, il reste impossible pour un agent d'intercepter une communication, de la même façon qu'un agent émettant une trace ou un signal ne peut pas choisir ses récepteurs.

L'état du système multi-agents étant stocké dans et géré par l'environnement, il est possible de prendre en compte le contexte dans la distribution des messages. De plus, l'environnement transférant les messages, il est possible de contrôler directement les communications.

Même dans le cadre des environnements, le concepteur doit choisir l'un des deux grands types de modèles de communication, directe ou via un espace partagé. Par exemple, l'architecture de référence d'un environnement proposé dans [Weyns et al., 2005] les représente dans des modules distincts (figure 2.10).

Ainsi, les agents peuvent échanger des messages classiques et interagir via un espace de tuples pour un comportement de type stigmergie [Ricci et al., 2007], mais les deux ne sont pas unifiés. Ce manque d'unification empêche la mise en oeuvre des communications multi-parties, alors que les responsabilités déléguées à l'environnement répondent aux critères que nous avons proposés. C'est pourquoi notre proposition sera un nouveau modèle d'environnement de communication, qui doit prendre en compte pour chaque communication et de façon standardisée les besoins des émetteurs et des récepteurs.

Nous discutons alors le concept de perception et sa relation avec l'interaction dans le cadre

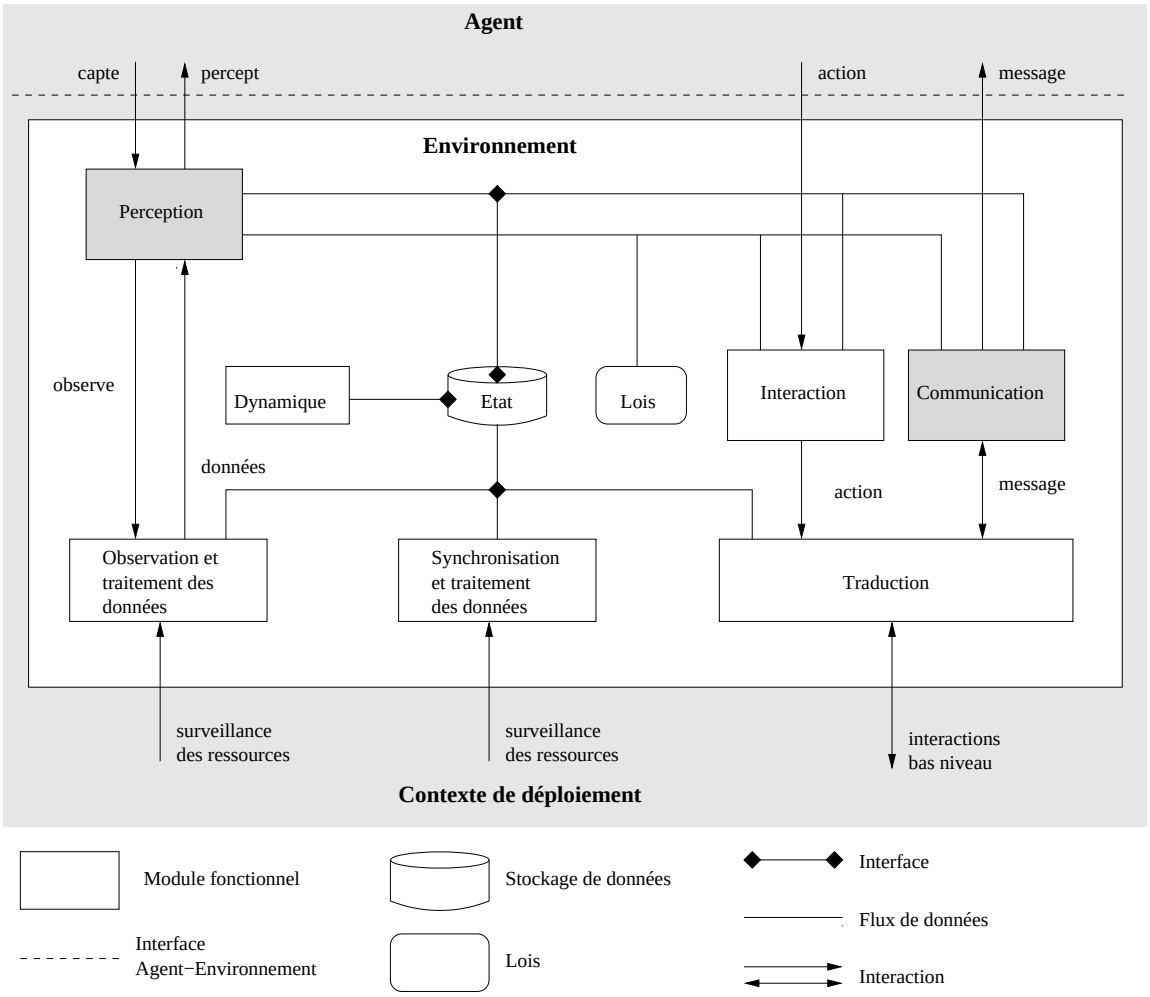


FIGURE 2.10 – Architecture d'environnement, modules *perception* et *communication* [Weyns et al., 2007]

de l'environnement.

2.2.5.a Perception active

Un agent doit pouvoir observer le contexte dans lequel il se trouve, et l'observation des messages fait partie plus globalement de la notion de perception.

Pour cela, nous étudions la notion d'*Active Perception*, introduite dans les systèmes multi-agents par [Weyns et al., 2004]. La perception active provient du domaine de la robotique [Bajcsy, 1988]. Il s'agit de décomposer la perception en plusieurs niveaux sous forme de processus, depuis l'obtention des données jusqu'à leur représentation symbolique. On peut donc découper le processus en plusieurs étapes, illustrées en figure 2.11. L'agent décide d'une action de perception par la mise en oeuvre d'un focus. Cette perception est contrainte au niveau de l'environnement

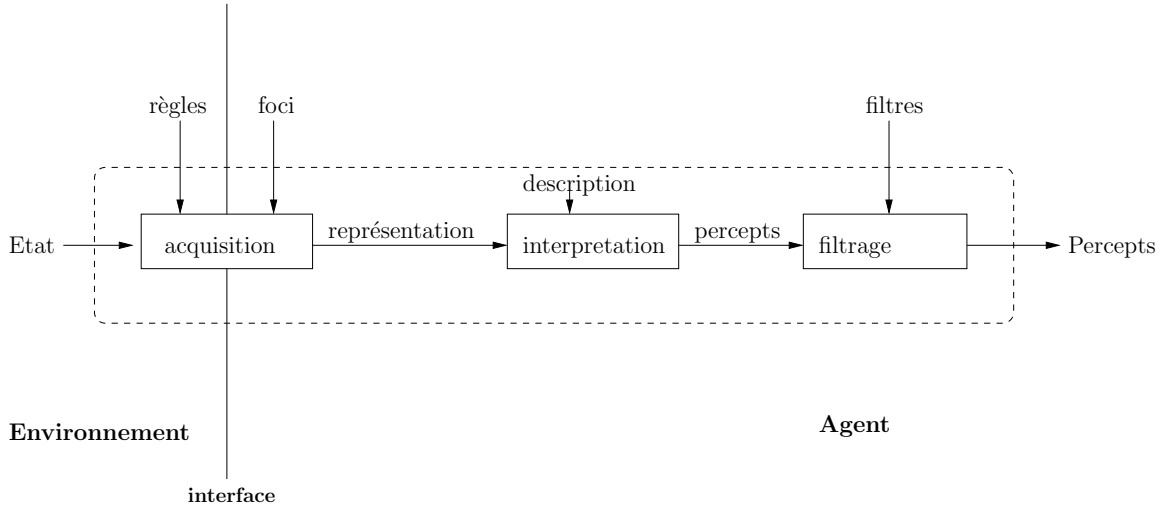


FIGURE 2.11 – Modèle de la perception active

par les capacités des capteurs et les règles de l'environnement [14]. Le résultat de la perception est un ensemble de données brutes. Ces données brutes sont interprétées pour être utilisables dans le raisonnement de l'agent dans le cadre de ses tâches, elles sont ainsi transformées en données symboliques de haut niveau. Enfin, l'agent filtre les informations de façon à sélectionner les données dont il a besoin.

La notion de filtrage est ici internalisée au niveau de l'agent, ce qui lui permet d'effectuer tous les traitements et sélections de façon autonome. Cela peut être vu de façon miroir avec les espaces partagés, dans lequel la sélection des informations est réalisée au niveau de l'infrastructure, avec pour conséquence un mode de représentation et de sélection des données peu expressif, mais aussi une délégation d'une partie de la complexité.

Dans la même logique, dans l'architecture proposée par Weyns *et al.*, un premier filtrage (par *foci*) est réalisé par l'environnement au moment de la décision de perception. Le filtrage détermine les données du contexte qui sont accessibles à l'agent, de façon à respecter les règles de l'environnement (et à limiter la diffusion des informations).

L'avantage de ce processus de filtrage partagé entre l'environnement et l'agent est que le contrôle est assuré, mais la suite du processus maintient une partie du calcul au niveau de l'agent. Pour pallier cette limite, nous proposons d'alléger la complexité des agents en transférant l'ensemble des calculs à l'environnement.

2.2.5.b Stigmergie, phéromones et champs à base de gradient

Les recherches actuelles sur la propagation des informations dans l'environnement utilisent un certain nombre de concepts provenant initialement des environnements pour agents réactifs, à savoir la stigmergie [Ricci et al., 2007], les phéromones [Weyns et al., 2005] et les champs à

14. Ce qui rejoint exactement la notion d'awareness, cf 1.4

base de gradient [Weyns et al., 2006]. Les informations sont des objets d'interactions dynamiques dans le temps et dans l'espace, qui sont ensuite perçus par les agents de façon indirecte. Une revue complète de la littérature peut-être trouvée dans [Weyns et al., 2007].

La stigmergie consiste, dans le cas général, à communiquer indirectement grâce à des signaux observables dans l'environnement. L'exemple le plus connu est celui des phéromones, objets détachés de leurs émetteurs, qui sont situées, et peuvent s'évaporer ou se composer (ce qui permet aux agents de se coordonner pour mener à bien leur tâche, voir par exemple [Drogoul et al., 1995]). Un autre concept a été mis en oeuvre, celui de champs à base de gradient, qui consiste en un signal propagé autour de l'émetteur et dont l'intensité varie suivant la distance à l'émetteur.

Il y a donc deux dynamiques liées aux objets vecteurs de la communication qui sont gérées par l'environnement : la première est liée à l'évolution de l'objet lui-même (par exemple l'évaporation d'une phéromone), et la seconde est liée à la propagation, qui n'est autre que la mise à disposition potentielle de cette information dans un contexte situé.

Parallèlement à la notion de focus pour la perception, les règles de propagation sont déterminées en fonction à la fois des choix de l'émetteur (l'intensité du signal, par exemple) et des lois de l'environnement. Cependant, ce sont à nouveau les récepteurs de l'information qui contrôlent leurs perception, et l'émetteur est passif dans ce choix.

2.2.6 Conclusion

Les communications médiées sont une composante des communications multi-parties. Elles capturent le cas où le choix des informations à recevoir est effectué sur l'initiative du récepteur lui-même. Elles permettent également la mise en place du contrôle *a priori* des communications, grâce au contrôle des accès à l'infrastructure de médiation.

Pour autant, les communications multi-parties ne peuvent pas être mises en place directement. La standardisation FIPA limite la recherche de partenaires à un système d'appariement par compétence (pages jaunes). Dans les modèles organisationnels, c'est la structure de l'organisation qui détermine les informations disponibles pour l'appariement (par exemple les groupes et rôles). Dans les deux cas, la recherche de partenaires n'est pas générique, car elle s'appuie sur des informations spécifiques du modèle.

Pour les modèles d'espaces partagés (espaces de tuples et middlewares) et *Publish/Subscribe*, la principale difficulté se situe dans la prise en compte des besoins de l'émetteur, qui ne peut pas effectuer son choix de récepteurs. De plus, les modèles existants ne regroupent que certaines informations dans l'espace partagé ou le serveur, et ne permettent pas l'appariement de valeurs, ce qui limite l'utilisation du contexte pour le choix des récepteurs.

Le regroupement de toutes les informations sur le système au sein de l'environnement est un moyen de permettre une recherche de récepteurs fondée sur ces informations. Cependant, l'étude des travaux sur l'environnement montre que même lorsque les communications directes et indirectes sont utilisées au sein d'un même système multi-agents, elles sont gérées de façon

séparées.

Pour le support des communications multi-parties, notre proposition sera donc un nouveau modèle d'environnement de communication, qui prend en compte pour chaque communication et de façon unifiée les besoins des émetteurs et des récepteurs, les informations sur l'état du système et le contrôle des communications.

2.3 Conclusion

Nous avons vu dans ce chapitre qu'il n'existe pas de solution complète pour les communications multi-parties. Le problème principal est la prise en compte des besoins des émetteurs et des récepteurs, les différents modèles favorisant soit les premiers (communications directes), soit les seconds (espaces partagés, *Publish/Subscribe*).

Les seuls mécanismes prenant en compte les deux sont le protocole contract-net et la *capability-based coordination*. Ils ont pour objectif de trouver le meilleur interlocuteur. Ils ne sont donc pas adaptés aux communications multi-parties, pour lesquelles les "bons" récepteurs d'un message ne sont pas uniquement les agents qui correspondent à la fois aux critères de l'émetteur et des récepteurs, mais tous ceux qui correspondent à au moins un de ces critères.

Pour résoudre ces problèmes et unifier les modèles d'interaction, il est nécessaire d'unifier le problème de la connexion entre agents (liés à l'interaction directe) et le problème de récupération des informations (liés à l'interaction indirecte). L'étude de l'état de l'art montre qu'il y a actuellement une disjonction entre les familles de modèles de communication directes et indirectes. Le concepteur de système multi-agents ne peut donc pas les utiliser conjointement de façon standardisée.

Pour cela, le modèle doit prendre en compte à la fois les besoins des émetteurs et ceux des récepteurs pour les mêmes communications. En outre, ce modèle doit pouvoir être utilisé dans le cadre d'une infrastructure générique afin de supporter aisément des formes dialogiques complexes comme les communications multi-parties.

Prendre en compte les besoins des récepteurs et ceux des émetteurs peut être réalisé par la diffusion de tous les messages et toutes les informations aux agents, mais ce n'est pas applicable dans le cas ou le nombre d'agents et/ou de messages (liés à la mise à jour, par exemple) est trop important. L'autre solution est la médiation des communications.

Notre approche se fonde sur une médiation par l'environnement, qui reprend les avantages des modèles de communication médiée que nous avons étudié au cours de ce chapitre. En particulier, la médiation est active, c'est à dire que l'agent ne doit pas chercher lui-même l'information. Les émetteurs doivent pouvoir exprimer leurs besoins, car la médiation ne doit pas empêcher les communications classiques. L'environnement contient l'ensemble des informations concernant le contexte, ce qui lui permet de les utiliser dans le choix des récepteurs.

Dans le prochain chapitre, nous présentons une vue d'ensemble de notre modèle, et montrons comment l'environnement permet le support des communications multi-parties. Nous précisons ensuite la formalisation du modèle d'environnement dans le chapitre 4.

Chapitre 3

Présentation générale du modèle d'environnement comme support actif de l'interaction

Sommaire

Nous avons vu dans les chapitres 1 et 2 la nécessité d'un nouveau modèle d'environnement pour obtenir une plus grande flexibilité des interactions. Au niveau multi-agents, nous souhaitons mettre en oeuvre des communications fondées sur les besoins des émetteurs et des récepteurs pour obtenir des communications multi-parties. Au niveau agent, il s'agit de mettre en oeuvre la notion de conscience des autres afin de donner aux agents la possibilité d'influer sur la réception des messages. En particulier, les agents doivent pouvoir recevoir des messages qui ne leur sont pas adressés.

Dans ce chapitre, nous introduisons le principe unifiant les différents types de communications, que nous appelons *coordination fondée sur les propriétés* (PBC [15]). Ensuite, nous donnons une vue générale de notre modèle d'environnement EASI (*Environnement Actif comme Support de l'Interaction* [16]) qui s'appuie sur ce principe, et explicitons son fonctionnement. Dans le chapitre suivant, nous introduisons la formalisation du modèle.

Introduction de l'exemple illustratif : la cité digitale

Afin de montrer l'utilité de la PBC et d'illustrer la modélisation présentée dans les deux chapitres suivants, nous introduisons un exemple. C'est un exemple de support de communauté dans le cadre des *digital cities* (voir par exemple [Benini et al., 2005; Camarinha-Matos et Afsarmanesh, 2002; Hattori et al., 1999; van den Besselaar et Beckers, 2005]). Il s'agit de mettre en commun les informations concernant la ville et de proposer un support de communication entre utilisateurs via un portail commun. Il y a donc une centralisation des informations et échanges au niveau de ce portail.

Les intervenants dans une cité digitale sont de deux sortes : les organisations dont le rôle est de fournir des informations sur les sources disponibles (restaurants, musées...) et les évènements (programmation des cinémas, météo, rencontre sportive...), et les utilisateurs qui recherchent des informations et en diffusent (commentaires sur un évènement, petites annonces).

Les flux d'information sont de trois ordres [Benini et al., 2005], schématisés en figure 3.1 :

1. *Délivrer de l'information au public.* L'information est unidirectionnelle, autrement dit publiée par des fournisseurs de contenu vers un ensemble de lecteurs. Par exemple, il s'agit de publier les décisions municipales ou les évènements et activités proposés par les associations.
2. *Proposer des services.* L'information est distribuée par des fournisseurs, mais il est possible de faire remonter des informations vers les fournisseurs de services selon des modalités fixées. Par exemple, la mise en place de débats participatifs au sujet d'une décision d'urbanisme entre dans ce cadre.
3. *Fournir un environnement de communication.* Les utilisateurs sont tour à tour fournisseurs et consommateurs d'informations. Par exemple, ils peuvent annoncer des évènements qui seront commentés par d'autres utilisateurs.

Le domaine des supports aux communautés est particulièrement pertinent pour notre problématique car il met en relation un grand nombre de sources d'information, qui peuvent apparaître ou disparaître dynamiquement de la communauté, et des utilisateurs aux besoins changeants. D'après [Benini et al., 2005], les portails se limitent à un seul type de service (délivrance d'information, services ou environnement de communication), alors que leur intégration permettrait

15. Property-Based Coordination
16. Environment as Active Support of Interaction

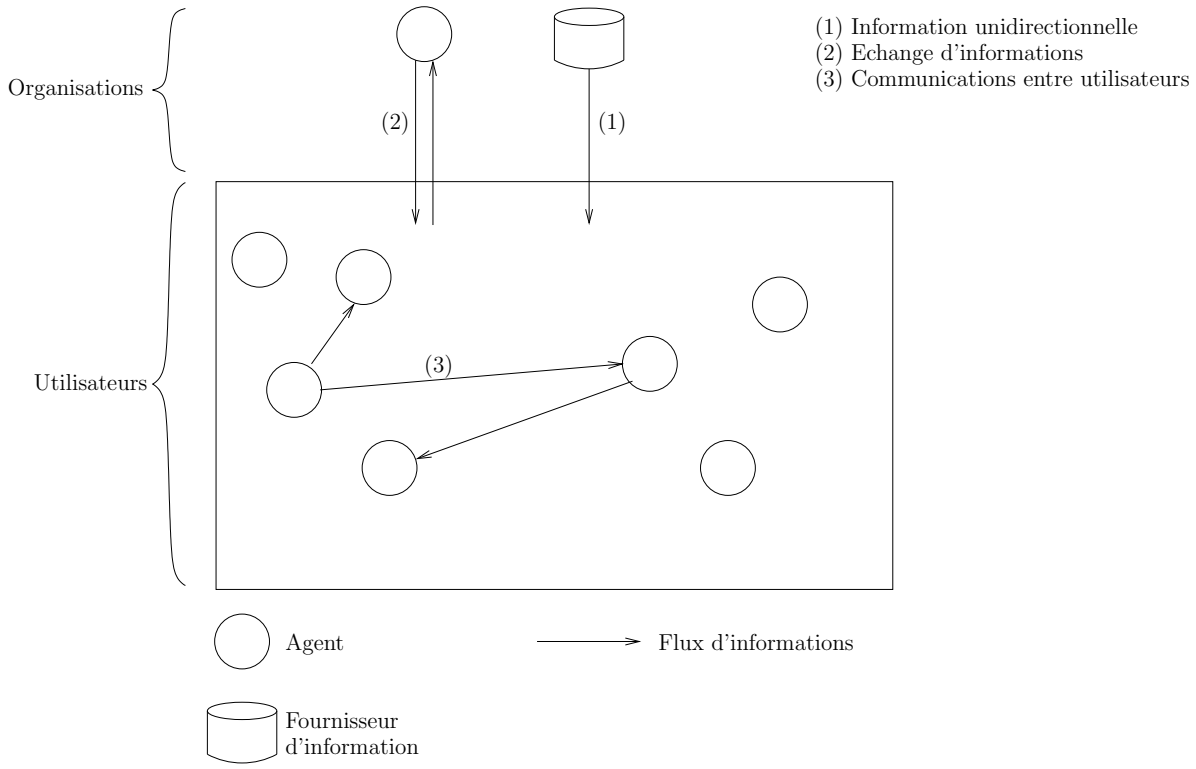


FIGURE 3.1 – Flux d'informations dans les *digital cities*

une plus grande flexibilité et personnalisation. A cette fin, nous souhaitons augmenter les capacités de communications des agents, humains et logiciels, en prenant en compte l'ensemble des informations disponibles. Pour cela, nous mettons en oeuvre les communications multi-parties, qui permettent à la fois aux émetteurs et aux récepteurs d'informations de spécifier leurs besoins.

Dans notre exemple de portail, chaque utilisateur possède un agent le représentant, et la cité digitale centralise les échanges d'information dans un environnement de communication commun. Les communications se déroulent entre utilisateurs et sont médiées par leurs agents représentants. De plus, ceux-ci peuvent mener à bien des tâches pour le compte de leur utilisateurs, et pour cela initier leurs propres communications.

L'une des originalités de notre modèle réside dans la possibilité offerte aux agents d'utiliser de multiples modèles de communication au sein d'un même système multi-agents. Nous montrons dans la suite du chapitre comment, avec un seul modèle de communication, les agents peuvent :

- utiliser des communications directes classiques ;
- utiliser des communications de groupes dont les membres ne sont pas connus individuellement ;
- utiliser des canaux de diffusion ;
- écouter les informations qui les intéressent en fonction du contexte.

Dans ce chapitre, nous décrivons le fonctionnement de l'environnement dans le cadre de

l'exemple de cité digitale. Nous utiliserons également cet exemple pour illustrer la formalisation du modèle d'environnement dans les chapitres 4 et 5.

3.1 Le fondement de notre approche : la coordination fondée sur les propriétés

3.1.1 Les différents types de communication

Comme nous l'avons vu précédemment, la communication est un acte mettant en relation un émetteur, un message et plusieurs récepteurs dans un contexte donné. Nous approfondissons ici les caractéristiques des communications prises en compte par le modèle que nous proposons.

Les communications peuvent être de plusieurs types. Lors d'une communication directe, où le message ne fait que transiter sur le réseau, la communication est dite *éphémère*. Au contraire, pour les communications indirectes (tuples, phéromones...), l'information est stockée pendant un certaine durée dans l'espace partagé. La communication est alors dite *temporelle*. Enfin, une communication dont le message reste disponible pour un temps indéterminé est dite *intemporelle*.

Du point de vue du vocabulaire, le terme de "message" a une forte connotation dans les systèmes multi-agents. Il est généralement utilisé pour désigner des communications éphémères entre agents. Les communications possédant une durée de vie ont de multiples dénominations : traces, phéromones, signaux, objets d'interaction, *etc.* De plus, ces communications sont dites "perçues" par les agents, alors que les messages sont "reçus". Nous rappelons que cela correspond à différents besoins concernant les communications. Pour l'émetteur, il s'agit d'une recherche d'interlocuteurs. Ces interlocuteurs sont soit des destinataires, soit des auditeurs, soit des groupes de réception. Pour le récepteur, il s'agit d'une recherche d'information, auquel cas il est écouteur de la communication.

Dans notre modèle, nous ne faisons pas de distinction *a priori* entre les différents types de communication. Ils sont tous gérés de façon standardisée par l'environnement. Nous utilisons dans la suite de la thèse les termes "message" et "réception" pour toutes les communications, y compris les communications temporelles, de façon à uniformiser leurs présentations. Quel que soit le type de communication, celle-ci peut potentiellement être reçue grâce aux besoins des émetteurs aussi bien que grâce à ceux des récepteurs eux-mêmes.

Le regroupement des messages dans un espace commun permet la mutualisation des communications et la présence de règles communes de transmission des messages. En effet, la gestion des messages étant effectuée par l'environnement, il peut mutualiser les besoins communs à plusieurs agents.

3.1.2 La coordination fondée sur les propriétés

Nous proposons le principe général de *coordination fondée sur les propriétés*. Cette proposition a fait l'objet d'une publication dans [Zargayouna, Saunier, et Balbo, 2006].

Une communication a lieu si un ensemble de conditions sur l'état du système multi-agents est satisfait.

Ces conditions peuvent être choisies à l'initiative de l'émetteur et/ou du récepteur, mais dans tous les cas il existe des règles explicites ou implicites qui permettent l'interaction. Par exemple, une communication directe correspond à une contrainte sur l'identifiant du récepteur à l'initiative de l'émetteur, et une communication par espaces de tuples correspond à un ensemble de contraintes sur cet espace exprimées par des motifs à l'initiative du récepteur. De même, la coordination fondée sur les capacités [Sycara et Wong, 2000] peut être modélisée par une contrainte sur les capacités et préférences des agents.

Pour pouvoir appliquer ces conditions, il faut un accès aux informations du système multi-agents. Nous proposons de représenter chaque composant du système multi-agents par une description observable stockée dans l'environnement, et de permettre aux agents d'utiliser ces descriptions pour gérer leurs interactions. Les composants du système multi-agents sont les agents, les messages ou tout autre objet. Les descriptions des composants, que nous appelons *entités*, sont modélisées uniformément sous forme d'un ensemble de propriétés.

Nous avons choisi de modéliser les conditions de réception sous la forme de *filtres*. Chaque filtre contient trois ensembles de conditions sur les descriptions des entités. Ces trois ensembles de conditions déterminent les entités concernées par le filtre, en d'autres termes *quels agents* (**qui ?**) doivent recevoir *quels messages* (**quoi ?**) dans *quel contexte* (**quand ?**). Lorsque les conditions d'un filtre sont vérifiées pour un agent et un message dans un contexte donné, ce message est reçu par cet agent. A chaque besoin d'un agent correspond un filtre placé dans l'environnement.

Plus formellement, le système multi-agents est donc un couple $SMA =< \Omega, \mathcal{E} >$, avec Ω l'ensemble des composants du système et $\mathcal{E}$ l'environnement. Les composants de Ω sont les agents (dans l'ensemble $\mathcal{A}$), les messages (dans l'ensemble $\mathcal{M}$) et les objets (dans l'ensemble $\mathcal{O}$). L'environnement $\mathcal{E}$ est notamment composé de l'ensemble des descriptions de ces composants ($\mathcal{D}$) et de l'ensemble des filtres ($\mathcal{F}$).

L'état courant du système multi-agents est défini comme étant l'ensemble des descriptions observables $\mathcal{D}$ de ce système. Dans le chapitre 1, nous avons vu que le contexte est composé du contexte de l'émetteur, du contexte de la transmission du message et du contexte du récepteur. Les conditions correspondant aux besoins des agents étant exprimées en fonction de l'état du système multi-agents, les éléments du contexte doivent posséder une description dans l'environnement pour pouvoir être utilisés dans le choix des récepteurs. Pour cela, nous proposons d'utiliser le modèle d'environnement comme un serveur d'information (section 1.4.2, voir par exemple [Fahy et Clarke, 2004]). Nous approfondissons la question de la gestion des descriptions dans les sections 3.2.2 et 3.3.2.

Dans la suite, nous appellerons "connexion" la transmission d'un message à un ou plusieurs agent(s). Nous considérons alors que le problème de connexion consiste à effectuer le choix des récepteurs d'une communication. Ce choix dépend des besoins des agents et est effectué grâce aux filtres.

3.1.3 Les communications multi-parties

Nous avons vu que les "initiatives" des émetteurs et récepteurs sont des conditions sur les messages, agents et/ou tout autre critère dont la description est accessible. Lorsqu'un message est émis, l'environnement se charge de créer les connexions en fonction de ces conditions, en traitant uniformément l'ensemble des initiatives.

Nous avons vu au chapitre 1 que la composition de plusieurs initiatives est le fondement des communications multi-parties. Nous explicitons ici les différences fondamentales entre le modèle que nous proposons et les autres modèles d'interaction, du point de vue fonctionnel :

- Par rapport à la diffusion des messages, le filtrage des messages intéressants est effectué lors de la transmission par l'environnement, au lieu d'*a posteriori* par chacun des agents.
- Par rapport aux interactions par espaces de tuples, les initiatives des émetteurs sont prises en compte, et la transmission des messages est active, autrement dit l'agent n'a pas besoin de faire l'action de chercher les messages, il les reçoit directement.
- Par rapport aux systèmes *Publish/Subscribe*, les initiatives des émetteurs sont prises en compte, et les critères de transmission des messages tiennent compte du contexte qui est stocké par l'environnement.

Ce sont les filtres qui déterminent le modèle d'interaction utilisé, et tous les filtres sont traités de la même façon. Ainsi, les agents peuvent utiliser de façon standardisée n'importe quel modèle en fonction de leurs besoins.

Scénario de communication pour l'exemple de cité digitale

Dans la cité digitale, le scénario de communication est le suivant :

1. Un agent *utilisateur* a besoin d'envoyer un message à tous les agents situés dans un lieu donné $l1$, pour obtenir une information sur ce lieu. L'utilisateur ne souhaite contacter que les agents qui sont disponibles pour ne pas surcharger les autres agents. Par exemple, l'agent a_1 a besoin de savoir si la librairie de l'université Paris-Dauphine est ouverte au temps t.
2. Un agent *utilisateur* souhaite recevoir les messages envoyés aux agents localisés dans un lieu donné $l1$ (par exemple à l'université Paris Dauphine) parce qu'il possède des connaissances liées à ce lieu, bien qu'il n'y soit pas présent au temps t.

La première partie du scénario exprime le besoin de l'émetteur. La réception est conditionnée par deux conditions sur le contexte du récepteur, connus seulement par lui-même : sa localisation et sa disponibilité. La disponibilité est un indicateur booléen dont la valeur est "faux" si l'agent est occupé, et "vrai" sinon. La deuxième partie du scénario exprime le besoin d'un récepteur. La réception est conditionnée par le contexte du message lui-même, c'est à dire sa destination. Ce scénario représente les spécificités des communications multi-parties : pour une même communication, plusieurs besoins sont exprimés, par l'émetteur et par les récepteurs, et ces besoins dépendent fortement des l'état courant du système multi-agents.

Nous illustrons cette première approche à l'aide de la figure 3.2. Dans les deux sections suivantes, nous précisons la dynamique du système du point de vue de l'agent et de l'environnement. L'agent a_1 souhaite envoyer un message m_1 aux agents présents dans le lieu l_1. L'agent a_2 souhaite écouter les messages envoyés aux agents situés en l_1. Seul l'agent a_4 est situé dans le lieu l_1.

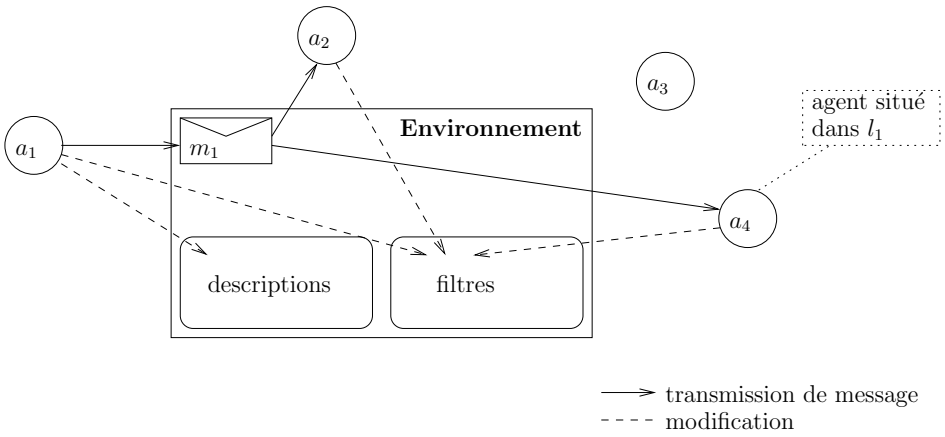


FIGURE 3.2 – Première approche de la modélisation du système multi-agents

Les agents modifient l'état du système $\mathcal{D}$ grâce aux descriptions des entités. Par exemple, les agents ajoutent leurs propres descriptions dans l'environnement. Ils déclarent leurs besoins auprès de l'environnement à l'aide de filtres. Ainsi, les agents a_1 et a_2 ajoutent les filtres relatifs à leurs besoins concernant les messages envoyés aux agents situés dans le lieu l_1. Enfin, les agents envoient leurs messages par le biais de l'environnement, par exemple l'agent a_1 envoie le message m_1. Nous détaillons en section suivante la dynamique du système du point de vue de l'agent.

L'environnement contient les descriptions et les filtres ajoutés par les agents, et se charge de transmettre les messages en fonction de ces deux ensembles. Nous abordons la modélisation de l'environnement dans la section 3.3 de ce chapitre.

3.2 Dynamique du système du point de vue de l'agent

Dans cette section, nous décrivons les interactions de l'agent avec l'environnement.

3.2.1 L'inscription des agents dans l'environnement

Les agents qui souhaitent communiquer à l'aide d'un environnement doivent s'inscrire auprès de cet environnement. Il est possible d'avoir plusieurs environnements de communication en parallèle dans un même système multi-agents, afin de compartimenter des espaces de communication n'ayant pas de rapport entre eux. Par exemple, les informations peuvent être regroupées par thème ou par localisation dans des environnements différents. Dans ce cas, les agents s'inscrivent auprès de chaque environnement qu'ils souhaitent utiliser.

Les différentes entités composant le système multi-agents (agents, messages et objets) possèdent une description, laquelle est stockée au niveau de l'environnement. Lorsqu'un agent "entre" (s'inscrit) dans l'environnement, il doit fournir sa description à l'environnement, par exemple son identifiant, le nom de l'utilisateur, sa localisation et un indicateur de disponibilité. En effet, puisque la distribution des messages est liée à des conditions sur les descriptions, un agent non décrit ne peut pas recevoir de messages.

L'agent peut "sortir" (se désinscrire) de l'environnement. Dans ce cas, sa description est retirée de l'ensemble des descriptions.

3.2.2 Les descriptions des entités par les agents

Dans notre modèle, les agents sont considérés comme des boîtes noires, c'est à dire que l'architecture de l'agent et ses connaissances ne sont pas observables. La description d'un agent correspond à l'idée de partie "publique" (ou "visible") de cet agent, dans le sens où cette description devient connue par l'environnement.

Une fois sa description transmise à l'environnement, un agent peut la modifier à tout moment. Par exemple, si l'agent a_4 est inoccupé lorsqu'il entre dans l'environnement, sa propriété "disponibilité" a pour valeur *vrai*. A chaque fois que l'état de l'agent change (*i.e.* que l'agent devient occupé ou redevient en attente), il modifie la valeur de cette propriété pour qu'elle reflète son état.

Les agents peuvent dynamiquement ajouter des descriptions d'autres entités du système multi-agents. Par exemple, un agent ayant connaissance d'un évènement à venir dans la ville peut ajouter une description de cet évènement.

Les agents peuvent également modifier ou retirer les descriptions qu'ils ont eux-même ajoutées dans l'environnement. Un cas particulier est celui des messages. L'agent transmet le message lui-même (et non sa description) à l'environnement, parce que l'environnement a besoin de stocker le message avant de le transmettre.

3.2.3 Les filtres des agents

Nous avons vu que les besoins des agents sont modélisés par des filtres. Un filtre contient des conditions sur les messages qu'il traite, des conditions sur les agents qui doivent recevoir ces messages, et une description du contexte dans lequel il doit être activé. La connexion est réalisée si l'ensemble des conditions est vérifié. Dans l'exemple de cité digitale, l'agent a_1 souhaite envoyer un message aux agents disponibles se situant dans le lieu $l1$, il ajoute donc un filtre f_{groupe} correspondant à ce besoin. Ce filtre définit un appariement entre la propriété "lieu" des agents et des messages, ainsi qu'un test sur la propriété "disponibilité" de l'agent.

Lorsqu'un agent souhaite envoyer un message, il doit d'abord s'assurer qu'il existe un filtre correspondant à ses besoins dans l'environnement, c'est à dire au message qu'il souhaite envoyer et aux récepteurs qu'il souhaite contacter. Lorsqu'un message transite dans l'environnement, seuls les filtres déjà présents dans cet environnement sont testés. S'il existe déjà un filtre corres-

pondant à ses besoins, l'agent peut transmettre son message. Sinon, il doit d'abord déposer son filtre, lequel sera ensuite utilisé pour transmettre le message.

Un agent souhaitant écouter certains messages doit ajouter un filtre décrivant ces messages. L'ensemble des messages transitant dans l'environnement est ensuite pris en compte par ce filtre. Dans l'exemple cité digitale, l'agent a_2, qui n'est pas situé dans le lieu $l1$ mais souhaite tout de même recevoir les messages destinés aux agents situés dans ce lieu dépose un filtre f_{ecoute} correspondant à ce besoin. Ce filtre définit un test sur la propriété "lieu" des messages, qui doit être égale à l_1, et désigne par son identifiant l'agent a_2 pour recevoir ces messages.

De la même façon que pour les descriptions, les agents peuvent ajouter, modifier et retirer leurs filtres dynamiquement. Si un filtre est utilisé par d'autres agents que celui qui l'a posé, le retrait du filtre a un impact sur ces agents. Par exemple, l'agent a_3 peut envoyer un message aux agent situés dans le lieu l_1 grâce au filtre f_{groupe}, mais si a_1 retire le filtre f_{groupe}, l'agent a_3 devra déposer un nouveau filtre avant de pouvoir envoyer son message. Pour cela, l'agent peut consulter quels sont les filtres stockés dans l'environnement. Nous exposons en section 3.3.2 comment ce problème est également en partie résolu au niveau de l'environnement.

3.3 Dynamique du système du point de vue de l'environnement

Dans la section précédente, nous avons vu de quelles façon les agents peuvent interagir avec l'environnement. Dans cette section, nous décrivons comment les actions des agents sont gérées par l'environnement, ainsi que le fonctionnement de notre modèle d'environnement lui-même.

3.3.1 La gestion des inscriptions des agents par l'environnement

L'environnement gère les entrées et sorties des agents, en lien avec leurs descriptions. Lorsqu'un agent entre dans l'environnement, il est inscrit en tant qu'utilisateur de l'environnement, et sa description est ajoutée. Dans l'exemple, les agents a_1, a_2, a_3 et a_4 sont inscrits dans l'environnement.

Nous avons vu que les agents peuvent ajouter et modifier des descriptions. Lorsqu'un agent sort de l'environnement, l'environnement le désinscrit, et retire l'ensemble des descriptions qu'il a ajouté. L'environnement, s'il perd la communication avec l'agent, désinscrit automatiquement cet agent au bout d'un certain laps de temps.

3.3.2 Le stockage des informations sur l'état du système et des filtres dans l'environnement

L'environnement stocke l'ensemble des descriptions données par les agents. Il gère également l'accès à ces descriptions : un agent ne peut modifier ou retirer que les descriptions qu'il a lui-même ajoutées. Ce choix vise à empêcher des agents malveillants de manipuler des descriptions d'entités que les autres agents ont ajoutées.

Un cas particulier est l'enregistrement des messages. Ceux-ci ne sont pas des descriptions, ils sont donc stockés à part. Par exemple, le message m_1 est enregistré pour le temps où le message doit rester dans l'environnement, tandis que la description du message m_1 est stockée avec les autres descriptions d'entités. Dans notre environnement, la description d'un message est automatiquement générée à partir du message lui-même.

Concernant les filtres, ils peuvent être ajoutés, modifiés et retirés dynamiquement par leur initiateur. De même que pour les descriptions, l'environnement stocke l'ensemble des filtres. Dans le scénario de l'exemple, l'environnement stocke les filtres f_{groupe} et f_{ecoute}, ajoutés respectivement par a_1 et a_2.

Lorsqu'un agent sort de l'environnement, ses filtres sont retirés. Nous avons fait ce choix pour éviter la présence de filtres obsolètes dans l'environnement génèrant des coûts de traitement inutiles. Cependant, un problème peut se poser dans le cas où plusieurs agents utilisent un même filtre pour transmettre leurs messages, mais où l'agent qui a déposé le filtre disparaît. Dans l'exemple, f_{groupe} est un filtre qui peut être utilisé par tous les agents pour envoyer leurs messages.

Dans cette thèse, le problème des filtres communs à plusieurs agents est partiellement géré par l'introduction de filtres "de l'environnement", que nous détaillons dans le chapitre 5.1.1. Le principe est de considérer qu'un certain nombre de filtres n'appartiennent pas à des agents, mais à l'environnement lui-même, parce qu'ils concernent non pas les besoins particuliers d'un agent, mais les besoins d'un ensemble d'agents. Dans ce cas, les filtres ne risquent pas de disparaître du fait d'un agent. Dans l'exemple, le filtre f_{groupe} peut être considéré comme appartenant à l'environnement.

Dans les cas où ces filtres sont effectivement ajoutés par un agent, nous considérons que les autres agents connaissent ces filtres et les remplacent en cas de disparition de l'agent. En effet, les agents peuvent à tout moment consulter les filtres présents dans l'environnement.

3.3.3 La transmission des messages par l'environnement

Le processus de transmission de messages est géré par l'environnement grâce à l'ensemble des filtres. L'environnement contient un algorithme d'appariement qui utilise l'ensemble des descriptions du système pour trouver les entités qui vérifient les conditions des filtres. Lors de l'envoi d'un message, sa description est ajoutée dans l'environnement. Tous les filtres concernant le message sont alors déclenchés, ceux de l'émetteur comme ceux des autres agents.

Dans notre exemple de cité digitale, nous avons vu que le filtre f_{groupe} transmet les messages ayant une propriété "lieu" à tous les agents ayant cette même localisation, et que le filtre f_{ecoute} transmet à l'agent a_2 les messages envoyés aux agents localisés dans le lieu l_1. Lors de l'ajout du message m_1 dans l'environnement, le filtre f_{groupe} de l'agent émetteur a_1 déclenche la transmission du message à l'agent a_4. Le filtre f_{ecoute} (initié par l'agent a_2) déclenche la transmission de ce même message à a_2.

Ce traitement unifié des filtres des émetteurs et des écouteurs de messages permet la prise

en compte de tous les besoins des agents exprimés dans l'environnement. Dans l'exemple, a_4 fait partie d'un groupe de réception du message, et a_2 est un écouteur. De cette façon, la médiation par l'environnement permet la mise en oeuvre des communications multi-parties.

En utilisant des espaces de tuples, des tableaux noirs ou systèmes *Publish and Subscribe*, seuls les besoins des agents récepteurs auraient été pris en compte. Dans le cas de la communication directe, seuls les besoins de l'émetteur sont naturellement pris en compte. La prise en compte des besoins des récepteurs nécessite alors soit des mécanismes de souscription entre tous les agents, soit un réseau d'accointances complet, qui pose des problèmes d'efficacité lorsque les mises à jours des informations sont fréquentes (voir chapitre 2.1).

Chaque modification de l'état du système multi-agents peut entraîner le déclenchement de certains filtres. Lorsque des modifications ont lieu, l'environnement vérifie donc s'il doit transmettre des messages dans le nouveau contexte.

La figure 3.3 récapitule le fonctionnement de l'environnement de communications. Les agents ont une description dans l'environnement par le biais de propriétés. Par soucis de clarté du schéma, nous n'en avons notées que deux dans la figure : "nom" et "lieu". Les agents mettent à jour leurs propriétés, et ils ajoutent et retirent des filtres dans l'environnement. L'agent a_1 a déposé le filtre f_{groupe}, et l'agent a_2 a déposé le filtre f_{ecoute}.

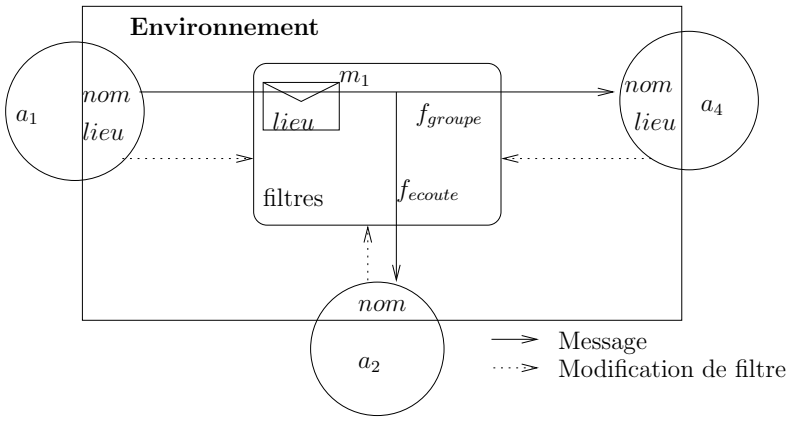


FIGURE 3.3 – Schéma fonctionnel du modèle d'environnement

Le message m_1 est décrit par une propriété "lieu". L'environnement gère les filtres et les descriptions. Lorsque le message m_1 est ajouté dans l'environnement, il est transmis par le biais des filtres des agents f_{groupe} et f_{ecoute}. La médiation des interactions permet de prendre en compte pour ce message les besoins de tous les agents, ici a_1 et a_2.

3.4 Exemple de modélisation : la cité digitale

Dans cette section, nous décrivons le fonctionnement du système multi-agents dans le cadre de l'exemple de cité digitale.

Pour cela, nous avons tout d'abord besoin de connaître les éléments constitutifs du système multi-agents, et de déterminer leurs propriétés. Dans le cadre d'une communauté de type cité digitale, nous avons vu que les agents peuvent représenter des organisations ou des citoyens, et leur description n'est pas identique : une propriété *sexe* n'a pas de sens pour une organisation.

Les propriétés des agents représentant les utilisateurs sont donc les suivantes :

1. *id* (chaîne de caractères) : l'identifiant de l'agent
2. *nom* (chaîne de caractères) : le nom de l'utilisateur
3. *dispo* (booléen) : indique si l'agent est disponible
4. *age* (entier positif) : l'age de l'utilisateur
5. *sexe* (chaîne de caractère) : le sexe de l'utilisateur
6. *lieu* (chaîne de caractère) : l'endroit où est situé actuellement l'utilisateur

Les agents représentant les organisations partagent les propriétés 1 et 2 avec les utilisateurs, auxquelles est ajoutée :

7. *type* (chaîne de caractères) : le type d'organisation (institution, entreprise...)

Notons que si certaines propriétés sont statiques (*id*, *nom*...) ou faiblement dynamiques (*age*), certaines peuvent au contraire être très changeantes (*disponibilité*, *lieu*). Il ne s'agit pas ici de considérer la façon dont ces valeurs sont acquises, celles-ci pouvant se faire par exemple pour la propriété *lieu* via l'utilisateur, via un serveur d'information ou via un système de géolocalisation, mais d'étudier la façon dont ces informations sont ensuite utilisées.

Pour la suite, nous considérons que l'environnement contient quatre agents a_1 à a_4, dont les descriptions sont les suivantes :

id	*nom*	*dispo*	*age*	*sexe*	*lieu*	*type*
a1	Saunier	true	26	M	Vincennes	
a2	Balbo	false	unknown	M	Arcueil	
a3	LAMSADE					laboratoire
a4	Remli	true	27	F	Dauphine	

Une fois les agents décrits, il faut décrire les différents types de messages.

Les messages classiques suivent la syntaxe FIPA-ACL [FIPA, 2002b], et à ce titre proposent tous les paramètres comme étant des propriétés visibles, hormis le contenu du message : *performative*, *sender*, *receiver*, *reply-to*, *language*, *encoding*, *ontology*, *protocol*, *conversation-id*, *reply-with*, *in-reply-to* et *reply-by*. Parfois sera ajouté un *type* de message, permettant de classifier le type d'information contenue dans le message. Toutes ces propriétés sont des chaînes de caractères.

Un second type de messages est utilisé dans le système multi-agents. Ce sont les messages d'information émis par les fournisseurs pour annoncer les évènements. Les messages d'information sont des communications temporelles, qui persistent dans l'environnement pendant un certain temps, tandis que les messages classiques sont éphémères.

Les messages d'information liés à des évènements sont décrits par les propriétés suivantes :

1. *titre* (chaîne de caractères) : l'intitulé de l'évènement
2. *type* (chaîne de caractères) : le type d'évènement (sportif, concert, météo...)
3. *lieu* (chaîne de caractères) : l'adresse physique de l'évènement
4. *source* (chaîne de caractères) : l'émetteur de l'information
5. *date_debut* (date) : date du début de l'évènement
6. *date_fin* (date) : date de la fin de l'évènement

Enfin, les agents *organisation* peuvent donner des informations sur des lieux. Il ne s'agit alors pas de messages, mais de descriptions concernant le contexte.

Les propriétés des messages d'information liés à des lieux sont :

1. *nom* (chaîne de caractères)
2. *type* (chaîne de caractères) : le type de lieu associé (parc, musée, restaurant...)
3. *adresse* (chaîne de caractères) : l'adresse physique du lieu
4. *source* (chaîne de caractères) : l'émetteur de l'information
5. *id_agent* (chaîne de caractères) : l'identifiant de l'(éventuel) agent lié au lieu
6. *contenu* (chaîne de caractères) : le contenu de l'information

Pour illustrer l'utilisation des filtres, nous décrivons ici un certain nombre de besoins en communication des agents et les filtres correspondant.

Le premier exemple est celui des besoins de communications directes, lorsque l'émetteur connaît l'identité de son interlocuteur. Les messages concernés sont ceux dont la propriété *receiver* a une valeur égale à la propriété *id* de l'agent concerné. Dans le cas des interactions dyadiques, le filtre f_{direct} effectue un appariement entre ces deux propriétés. Le récepteur de la communication est de type *destinataire*, car il est prévu et connu par l'émetteur qui a initié la connexion.

Un second besoin est la communication de groupe, lorsque l'émetteur choisit un groupe dont il connaît les critères, mais dont il ne connaît pas nécessairement les membres. Par exemple, un agent *utilisateur* veut envoyer une requête d'information à tous les agents disponibles qui se situent dans un lieu donné. Une solution est de décrire une propriété *lieu* dans la description du message, auquel cas le filtre f_{groupe} effectue un appariement entre la propriété *lieu* de l'agent et la propriété *lieu* du message, auquel s'ajoute le test sur la propriété *dispo* de l'agent qui doit être égale à *true*.

Avec ce filtre, tous les messages ayant la propriété *lieu* seront transmis aux agents disponibles situés dans le lieu indiqué. Soit un message m_1 ayant pour description "$a3$" comme valeur de la propriété *receiver* et "*Dauphine*" comme valeur de la propriété *lieu*. Il est transmis à tous les agents disponibles de l'université Paris-Dauphine grâce au filtre f_{groupe}, dans l'exemple seulement a_4, et à l'agent a_3 grâce au filtre f_{direct}.

Ces deux premiers exemples montrent comment exprimer les besoins de l'émetteur. Les communications multi-parties doivent également prendre en compte les besoins des récepteurs. Un

exemple de récupération des données en fonction des besoins du récepteur est lié à la diffusion des évènements. Les agents *organisation* déposent dans l'environnement des annonces d'évènement, lesquelles sont récupérées par les agents intéressés.

Soit un agent a_x souhaitant recevoir toutes les annonces de ballet ayant lieu à l'opéra Bastille, cet agent dépose un filtre f_{ballet}. Les conditions sur le message portent sur la propriété *type*, qui doit être égale à "ballet", et sur la propriété *lieu*, qui doit être égale à "opera bastille". La condition sur les agents qui doivent recevoir ces messages porte sur la propriété *id* de l'agent a_x. L'agent est un *écouteur*, puisque c'est à son initiative qu'il accède aux informations.

Les filtres peuvent prendre en compte le contexte de transmission, c'est à dire la description d'autres entités. Par exemple, l'agent a_x souhaite ne recevoir les évènements de type concert que si ceux-ci ont lieu dans un restaurant. Pour cela, le filtre $f_{concert}$ utilise les informations liées aux lieux. Il effectue un appariement entre la propriété *lieu* du message et la propriété *nom* des descriptions liées à des lieux, et vérifie que la propriété *type* du message est égale à "concert" et que la propriété *type* de la description est égale à "restaurant".

Grâce à ce filtre, l'agent reçoit tous les évènements déjà décrits dans l'environnement et dont les descriptions sont correctes pour le filtre. De plus, tant qu'il laisse son filtre dans l'environnement, il recevra également tous les nouveaux évènements ajoutés par les autres agents dont les propriétés sont correctes pour le filtre.

Nous donnons ci-dessous une description d'évènement et une description de lieu dont les valeurs des propriétés sont correctes pour $f_{concert}$.

titre	type	lieu	source	date_debut	date_fin	contenu
"John Quartz' quartet"	"concert"	"La trompette bleue"	a10	18/08/2007 19 :30	18/08/2007 23 :00	"Concert Jazz"

nom	type	adresse	source	id_agent	contenu
"La trompette bleue"	"restaurant"	"7, Bld Saint Pierre"	a10	a10	"Concerts tous les soirs sauf mardi"

L'agent *organisation* a_{10}, qui est lié au restaurant "La trompette bleue", a ajouté une description de ce lieu dans l'environnement. Il ajoute également les différents concerts organisés sous la forme de messages d'information sur les évènements, en l'occurrence un concert de Jazz. L'agent a_x ayant posé le filtre $f_{concert}$ reçoit cet évènement car le lieu où se déroule le concert est un restaurant.

Le dernier cas est une interaction de type écoute flottante. C'est le cas lorsque le message est initialement adressé à d'autres agents, par exemple lorsqu'un agent a_x souhaite écouter les requêtes émises dans un lieu donné. Le filtre f_{ecoute} s'écrit alors de la même façon que le filtre f_{groupe}, hormis l'appariement sur les propriétés "lieu", puisque cette fois le récepteur est l'agent ayant déposé le filtre. Le filtre f_{ecoute} est donc fondé sur l'identifiant de a_x pour le choix du récepteur, et sur la propriété *lieu* des messages pour le choix de ceux-ci.

3.5 Conclusion

Dans ce chapitre, nous avons introduit les principes de notre modèle d'environnement EASI (Environnement Actif comme Support de l'Interaction). Il s'agit d'un environnement de communication actif utilisant la notion de *coordination fondée sur les propriétés*.

Le modèle repose sur une description des entités du système multi-agents au sein de l'environnement. Chaque entité est décrite grâce à un ensemble de propriétés, et l'ensemble des descriptions des entités représente l'état courant du système. Les agents sont en charge de la mise à jour des descriptions.

Nous avons proposé la mise en place de filtres pour représenter les besoins en communication des agents. Ces filtres sont des conditions sur l'état du système. Un filtre met en relation les messages avec leurs récepteurs dans certains contextes grâce aux descriptions.

Chacun des agents ajoute et retire des filtres dans l'environnement afin d'exprimer aussi bien ses besoins en émission que ses besoins en réception. L'une des originalité de notre modèle est que, grâce à la modélisation unifiée des entités et des besoins, chaque communication est multi-partie. L'environnement prend en compte de manière standardisée tous les besoins des agents pour la sélection des récepteurs.

Nous avons illustré le fonctionnement de l'environnement sur un exemple de cité digitale, ainsi que la façon dont notre modèle permet la mise en oeuvre de scénarios de communication multi-parties. Dans le chapitre suivant, nous détaillons la formalisation de notre modèle d'environnement.

Chapitre 4

Formalisation du modèle d'environnement actif comme support de l'interaction

Sommaire

Le modèle EASI que nous proposons dans cette thèse s'appuie sur le modèle d'environnement actif initié dans [Balbo, 2000], qui permet l'interception des communications par les agents. Nous proposons ici d'améliorer le modèle d'interaction par une formalisation de l'environnement et par un processus de sélection des récepteurs dans le cadre des communications multi-parties. L'originalité d'EASI est de modéliser l'ensemble des éléments du système multi-agents de façon

unifiée, et de les utiliser lors du choix des récepteurs. La gestion des communications est réalisée par l'environnement.

Nous avons vu dans les chapitres précédents que le problème de connexion consiste à rechercher les "bons" récepteurs. Ce problème de connexion est résolu par l'environnement en fonction de la description qu'il a des composants de l'interaction : les agents, les messages et le contexte. Nous modélisons les besoins des agents par des *filtres*. Un filtre constitue la réification d'une connexion, il décrit de manière unifiée les conditions de réception des messages.

4.1 La formalisation issue de l'Analyse de Données Symboliques

L'idée fondatrice de notre approche est que l'établissement des connexions dépend de la description du système multi-agents. L'environnement de communication, auquel est délégué la recherche des récepteurs, doit contenir deux ensembles de données, la description des entités du système multi-agents et un ensemble de conditions (les filtres). Dans cette section, nous détaillons les différents critères auxquels le modèle de données doit répondre, puis nous introduisons la formalisation utilisée pour le modèle.

4.1.1 Problématique du modèle

Le modèle formel d'environnement doit répondre à un certain nombre de critères : (i) l'efficacité de la recherche des récepteurs, (ii) l'expressivité du format de représentation des données, et (iii) la facilité d'exploitation du modèle.

Comme nous l'avons vu, notre modèle vise à regrouper dans l'environnement toutes les informations sur les composants du système multi-agents qui sont nécessaires à l'interaction (critère i). L'environnement peut rapidement contenir un nombre important de descriptions et de filtres. Or, les communications doivent être délivrées rapidement aux récepteurs, il faut donc être capable de décrire et d'organiser des classes de récepteurs de grande taille dans le but de les retrouver de façon efficace.

Le critère (ii) est celui de la représentation des données, qui doit être expressive et permettre de gérer de façon unifiée à la fois les descriptions des entités du système multi-agents et celles des filtres.

Le dernier critère (iii) concerne l'exploitation du modèle : la modélisation doit être indépendante du modèle d'implémentation, et elle doit être opérationnelle, c'est à dire pouvoir être mise en oeuvre rapidement dans le cadre d'applications réelles, en s'appuyant si possible sur des technologies existantes.

La recherche des récepteurs implique un processus de discrimination entre d'une part les descriptions respectant les conditions des filtres, et d'autre part les autres descriptions. En d'autres termes, il s'agit d'une classification des descriptions des composants du système multi-agents selon les filtres présents dans l'environnement. C'est pourquoi nous avons choisi la formalisation proposée par l'Analyse de Données Symboliques (ADS) [Billard et Diday, 2003, 2007; Diday

et Noirhomme-Fraiture, 2008] pour notre modèle d'environnement EASI. L'ADS propose un modèle d'analyse de grands ensembles de données dont la formalisation, que nous détaillons ci-dessous, répond à nos critères.

En s'appuyant sur les concepts logiques d'intension et d'extension, elle permet de décrire et de manipuler des classes de données. L'intension est une description logique d'un groupe d'individus [17], par exemple les hommes qui ont plus de 30 ans. L'extension de cette intension est l'ensemble des individus qui satisfont la description. Ces concepts sont détaillés dans la section suivante. La formalisation est expressive car elle permet l'utilisation de tout type de données, qu'elles soient qualitatives, quantitatives ou complexes, c'est à dire formées de plusieurs données simples. Enfin, la formalisation ADS est à la fois indépendante de l'implémentation et exploitable, car les définitions de classes peuvent être traduites en requêtes SQL et en formules de logique du premier ordre [18].

4.1.2 Introduction à la formalisation issue de l'analyse de données symboliques

Nous introduisons maintenant les définitions de base de l'Analyse de Données Symboliques, que nous adaptons au cadre des systèmes multi-agents.

Le monde réel est composé de n individus $\omega_i \in \Omega = \{\omega_1, ..., \omega_n\}$, qui sont chacun caractérisés par r variables et une application p donnant la valeur de ces variables p_j, avec $j = 1, ..., r$. Chacune des variables p_j possède un domaine de description D_j. Par exemple, l'individu ω_1 possède six variables : *id*, *nom*, *dispo*, *age*, *sexe* et *lieu*. Leurs valeurs sont $p_1(\omega_1) =$ "$a1$", $p_2(\omega_1) =$ "$Saunier$", $p_3(\omega_1) = true$, $p_4(\omega_1) = 26$, $p_5(\omega_1) =$ "M", $p_6(\omega_1) =$ "$Vincennes$". Nous utilisons indifféremment la numérotation de la propriété ou son nom, par exemple $p_3(\omega)$ est équivalent à $dispo(\omega)$. Les domaines de description sont $D_1 = D_2$ deux chaînes de caractères non-vides, $D_3 = \{true, false\}$, $D_4 = [0..120]$, $D_5 = \{$"M", "F"$\}$ et $D_6 = L$ avec L l'ensemble des localisations valides.

Soit $p_j(\omega_i)$ la valeur particulière prise par la variable p_j pour le $i^{ème}$ individu, d_{ω_i} est la description de l'individu ω_i par l'ensemble des valeurs de ses variables dans le monde modélisé. Dans l'exemple précité, d_{ω_1} est égal à ("$a1$", "$Saunier$", $true$, 26, "M", "$Vincennes$"). L'ensemble des descriptions d'individus est noté $\mathcal{D} = \{d_{\omega_1}, ..., d_{\omega_n}\}$

En pratique, les données symboliques correspondant à l'ensemble des individus sont représentées par une matrice $n \times r$, dont les colonnes sont les variables et les lignes les individus. Dans la figure 4.1, nous illustrons les premières cases du tableau avec un exemple. Lorsqu'une variable n'est pas définie pour un individu donné, cette case n'est pas remplie.

Les concepts du monde réel $C_k,\ k = 1, .., N$ sont des abstractions qui décrivent un ou plusieurs individus. Si chaque individu est un concept, alors $n = N$. Un concept est une description en intension, son extension notée $E(C_k)$ contient tous les individus satisfaisant cette description.

17. Le terme "individu" désigne tout type d'entité identifiable, par exemple humains ou objets.
18. Ce qui permet notamment d'utiliser des systèmes experts pour gérer l'environnement (section 6.13)

	p_1	p_2	p_3	p_4	p_5	p_6	p_7	...	p_r
ω_1	"a1"	"Saunier"	true	26	"M"	"Vincennes"		...	$p_r(\omega_1)$
ω_2	"a2"	"Balbo"	false	unknown	"M"	"Arcueil"		...	$p_r(\omega_2)$
ω_3	"a3"	"Lamsade"					"laboratoire"	...	$p_r(\omega_3)$
ω_4	"a4"	"Remli"	true	27	"F"	"Dauphine"		...	$p_r(\omega_4)$
ω_5	"a5"	"Boria"	false	23	"M"	"Dauphine"		...	$p_r(\omega_5)$
...	...	...	...	...	...	...	...	...	...
ω_n	$p_1(\omega_n)$	$p_2(\omega_n)$	$p_3(\omega_n)$	$p_4(\omega_n)$	$p_5(\omega_n)$	$p_6(\omega_n)$	$p_7(\omega_n)$	...	$p_r(\omega_n)$

FIGURE 4.1 – Exemple d'un tableau de données pour l'analyse de données symboliques

Par exemple, un concept peut regrouper tous les individus dont la localisation est l'université Paris Dauphine, tandis que l'extension de ce concept sera composée de l'ensemble des individus dont la valeur de la variable *lieu* est "*Dauphine*".

Un objet symbolique est une description en intension d'une *classe* d'individus. Son extension est l'ensemble des individus (la classe) dont la description satisfait les relations avec des valeurs de description.

Une assertion est un objet symbolique particulier. Soit $v = (v_1, ..., v_r)$ la description requise d'un individu ω, avec v_j une valeur appartenant à l'ensemble de description D_j, l'assertion a la forme générale suivante :

$as = [p_{j_1} R_{j_1} v_{j_1}] \wedge ... \wedge [p_{j_v} R_{j_v} v_{j_v}]$ avec $1 \leq j_1, ..., j_v \leq r$.

R_j est l'opérateur de comparaison entre la variable p_j et la valeur z_j. $[p_j R_j v_j]$ est donc un test de la valeur de p_j. Les indices j_1 à j_v sont inclus dans les indices $1, ..., r$ des variables, ce qui signifie que toutes les variables ne sont pas nécessairement testées. Par exemple, l'assertion $as_1 = [p_4 < 30] \wedge [p_6 = \text{"}Dauphine\text{"}]$ décrit les individus de moins de 30 ans situés dans l'université Paris-Dauphine. Les tests sont effectués sur les variables p_4 (la variable *age*) et p_6 (la variable *lieu*). Avec les individus de la figure 4.1, l'extension $E(as_1)$ est une classe d'individus qui contient les individus ω_4 et ω_5.

Une assertion correspond à des conditions sur les valeurs des variables décrivant un individu. L'assertion $as(\omega)$ est une fonction de vérité de Ω vers $\{true, false\}$. Lorsqu'on évalue une assertion as pour un individu particulier $\omega \in \Omega$, elle prend la valeur $as(\omega) = true$ si l'assertion est vérifiée, et $as(\omega) = false$ sinon.

Une assertion est un ensemble de conditions décrivant un individu. Il est possible de réunir les conditions sur plusieurs individus au sein d'un même objet symbolique qui est alors appelé une *horde*. Par exemple, la horde $h = [type(u) = \text{"}laboratoire\text{"}] \wedge [age(v) > 25]$ décrit deux individus. Son extension est l'ensemble des couples d'individus (u, v) respectant les conditions. Dans la figure 4.1, il s'agit des deux couples (ω_3, ω_1) et (ω_3, ω_4). L'individu ω_3 est le seul individu satisfaisant la première condition $[type(u) = \text{"}laboratoire\text{"}]$, et les individus ω_1 et ω_4 satisfont la seconde condition $[age(v) > 25]$.

Les hordes peuvent contenir des individus dont les descriptions ne sont pas homogènes, dans

le sens où ils ne sont pas décrits par les même variables. Par exemple, les individus ω_1 et ω_3 ont seulement deux variables en commun, puisque l'individu ω_1 est décrit par $p_1, ..., p_6$ alors que l'individu ω_3 est décrit par p_1, p_2 et p_7.

La figure 4.2, inspirée de [Billard et Diday, 2003], récapitule les notions présentées précédemment. Le monde réel est composé d'individus et de concepts. Les concepts sont des descriptions en intention d'un ou plusieurs individus. L'extension d'un concept $E(C_k)$ (relation (1)) contient des individus $\omega \in \Omega$, qui forment une classe. À chaque individu ω du monde réel correspond une description $d_\omega \in \mathcal{D}$ dans le monde modélisé, obtenue par la fonction p (relation (2)). Enfin, un objet symbolique est la formalisation d'un concept C_k (relation (5)). Les objets symboliques de $\mathcal{S}$ sont des descriptions en intension qui utilisent des descriptions d'individus d_ω (relation (3)), et l'extension $E(as)$ de ces objets symboliques contient tous les individus qui satisfont cette description (relation (4)).

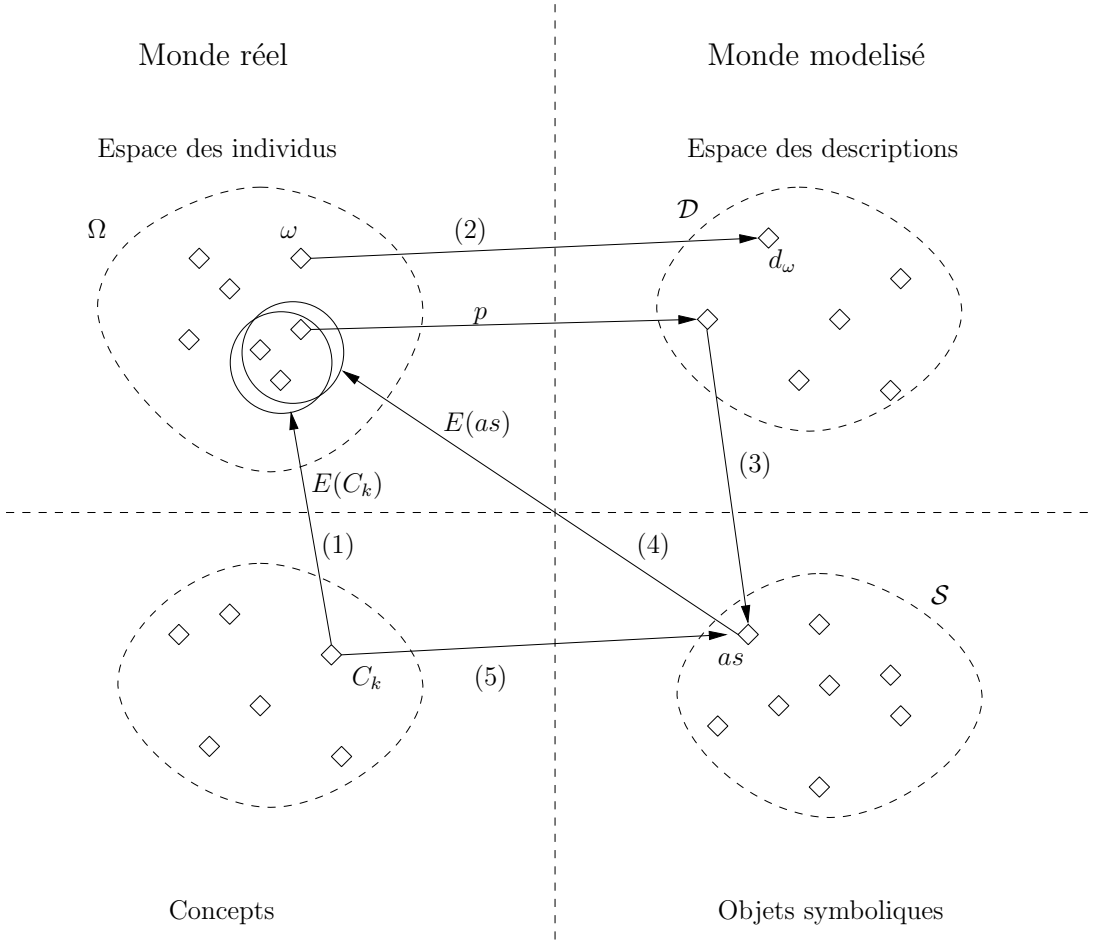


FIGURE 4.2 – Schéma général de la modélisation par l'analyse de données symboliques

Si la formalisation as d'un concept C_k est parfaite, leurs extensions sont identiques ($E(as) = E(C_k)$). Une problématique de l'Analyse de Données Symboliques est de vérifier que le processus

de modélisation est correct, par la comparaison entre ces deux extensions. Deux types d'erreur peuvent apparaître :

– l'extension $E(as)$ contient des individus qui n'appartiennent pas à $E(C_k)$,

– l'extension $E(as)$ ne contient pas certains individus qui appartiennent à $E(C_k)$

Cette problématique est importante lorsque les individus du monde réel sont complexes, et que les variables utilisées pour les décrire ne sont pas évidentes à choisir. Par contre, un système multi-agents contient des composants qui ont déjà été modélisés par leurs concepteurs. En conséquence, dans cette thèse, nous posons comme hypothèse que le processus de modélisation des descriptions et des objets symboliques est correct.

Pour notre formalisation, nous utilisons les notions d'individu, de description et d'assertion. Lorsque certaines notations sont adaptées au modèle EASI, nous rappelons alors les notations de l'ADS auxquelles elles se rattachent.

4.2 Le modèle formel d'environnement

Nous avons donné une vue générale du rôle de l'environnement dans le chapitre précédent : avec EASI, la sélection des récepteurs est effectuée en fonction de règles de transmission des messages que sont les filtres. En rapport avec l'analyse de données symboliques, les filtres sont composés d'objets symboliques, par exemple des descriptions de classes d'agents récepteurs et des descriptions de classes de messages. L'extension de l'objet symbolique décrivant les messages est la classe des messages qui sont transmis par le filtre. L'extension de l'objet symbolique décrivant les récepteurs est la classe d'individus qui reçoivent ces messages.

Nous avons également défini au chapitre précédent le système multi-agents comme un couple $SMA =< \Omega, \mathcal{E} >$, avec Ω l'ensemble des composants du système et $\mathcal{E}$ l'environnement. Ω est l'ensemble des individus (aussi appelés entités) du monde réel, ce sont donc les agents, messages et objets qui constituent le SMA. L'environnement contient le monde modélisé, autrement dit les descriptions des individus et les filtres des agents.

Définition 1 – *L'environnement $\mathcal{E}$ est un triplet $\langle \mathcal{D}, \mathcal{F}, IP \rangle$, avec :*

– *$\mathcal{D}$ l'ensemble des descriptions des individus*

– *$\mathcal{F}$ l'ensemble des filtres*

– *IP l'intervalle des priorités, avec $IP \subset \mathbb{N}$*

L'environnement comprend un ensemble de n descriptions d'entités $\mathcal{D} = \{d_{\omega_1}, d_{\omega_2}, ..., d_{\omega_n}\}$. Ce sont les descriptions de l'ensemble des entités $\Omega = \{\omega_1, \omega_2, ..., \omega_n\}$ qui composent le système multi-agents.

L'environnement contient également un ensemble de k filtres, $\mathcal{F} = \{f_1, f_2, ..., f_k\}$. Un filtre f est une conjonction de conditions sur les propriétés des entités liées à un problème de connexion particulier. Pour pouvoir ordonner les filtres, ceux-ci ont une priorité dans l'intervalle IP, qui est fixé au niveau de l'environnement. Les filtres sont définis en section 4.2.2.

4.2.1 Les entités

Nous avons vu que les agents fournissent à l'environnement les descriptions des entités $\Omega = \{\omega_1, \omega_2, ..., \omega_n\}$. Tout composant du système multi-agents dont le concepteur souhaite utiliser la description est une entité.

Nous distinguons trois types d'entités : les agents, les messages et les objets. Les agents sont les entités qui peuvent interagir proactivement avec l'environnement. Les messages sont les entités qui peuvent être transmises par l'environnement de communication aux agents. Dans l'exemple de cité digitale, il s'agit des messages et des informations sur les évènements. Les objets sont toutes les entités qui ne sont ni des agents ni des messages, mais qui ont tout de même une description dans l'environnement. Il peut s'agir de ressources, ou de données du contexte. Dans l'exemple, les informations sur les lieux sont des objets. Nous notons par la suite $\mathcal{A}$ l'ensemble des agents, $\mathcal{M}$ l'ensemble des messages et $\mathcal{O}$ l'ensemble des objets, avec $\Omega = \mathcal{A} \cup \mathcal{M} \cup \mathcal{O}$.

Chaque entité ω est un composant du système multi-agents, et à chaque entité correspond une description. Une description est composée d'un ensemble de propriétés p_j, appelées aussi variables dans le cadre de l'ADS.

Définition 2 *Propriété*
Soit p_j une propriété et D_j le domaine de description de la propriété p_j
$p_j : \Omega \mapsto D_j \cup \{unknown, null\}$

$$p_j(\omega) = \begin{cases} valeur \in D_j & si\ la\ valeur\ est\ définie \\ unknown & si\ la\ propriété\ n'est\ pas\ renseignée \\ null & si\ la\ propriété\ n'est\ pas\ définie \end{cases}$$

Une propriété p_j est une fonction qui associe une valeur à une entité $\omega \in \Omega$. Le domaine de description D_j de la variable peut être quantitatif, qualitatif, un intervalle ou un ensemble fini de données. Nous définissons P comme l'ensemble des propriétés utilisées pour décrire les entités, avec $P = \{p_j | \forall j \in \{1, ..., r\}\}$.

La description d_ω d'une entité ω est composée de l'ensemble des noms et des valeurs et de ses propriétés.

Définition 3 *Description d'une entité*
Soit ω une entité et $p_1, ..., p_r$ ses propriétés, la description de l'entité ω est l'ensemble de couples
$\forall p_j \in P,\ d_\omega = \{< p_j, p_j(\omega) > | p_j(\omega) \neq null\}$

A la différence de l'ADS, il s'agit d'un ensemble de couples <propriété, valeur>. La valeur d'une propriété peut être modifiée dynamiquement, excepté si elle est égale à *null*, ce qui exprime l'absence de cette propriété. Notons que l'utilisation de noms de propriétés et de valeurs nécessite une ontologie commune de la part des agents.

Une illustration de l'environnement est donnée en figure 4.3. Les entités possèdent une description dans l'environnement, par exemple l'agent 1 (entité ω_1) a pour description six propriétés : *id*, *nom*, *dispo*, *age*, *sexe* et *lieu*. Si l'on interroge la valeur de la propriété *id* de l'agent 1, on obtient "$a1$". Si l'on interroge la valeur de la propriété *type*, on obtient *null*. De même seront décrites les autres entités ω_2, ω_3 et ω_6 qui représentent deux agents et un message. Le deuxième élément de l'environnement est l'ensemble des filtres, que nous décrivons dans la section suivante. Le dernier élément représenté dans l'environnement est fonctionnel, il concerne la dynamique de transmission des messages. Un algorithme de mise en correspondance est utilisé de façon à faire correspondre les filtres avec les descriptions des entités lors de la transmission des messages.

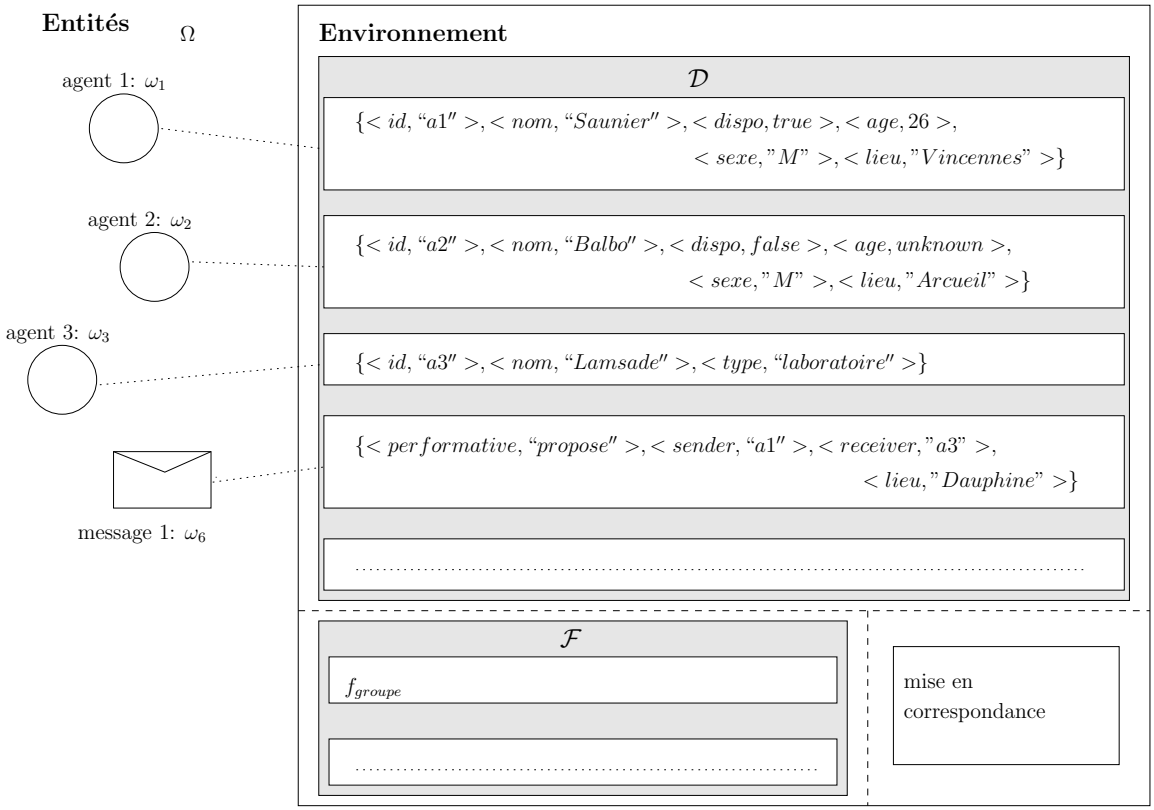


FIGURE 4.3 – Modèle d'environnement : un exemple

Exemple de la cité digitale. Nous avons vu dans le chapitre précédent que les propriétés des agents *utilisateurs* sont *id* pour l'identifiant de l'agent, *nom*,*dispo* (qui indique si l'agent est disponible), *age*,*sexe* et *lieu* (qui indique l'endroit où est localisé l'utilisateur). Les agents *organisation* sont décrits par les propriétés *id*, *nom* et *type*.

Un exemple de description d'un agent *utilisateur* ω_1 est ainsi :
$d_{\omega_1} = \{< id, \text{“}a1\text{”} >, < nom, \text{“}saunier\text{”} >, < dispo, true >, < age, 26 >, < sexe, \text{“}M\text{”} >, <$

$lieu$, "$Vincennes$" $>\}$

4.2.2 Les filtres

Pour une connexion, l'émetteur recherche de façon précise les récepteurs, dans un contexte de transmission particulier.

Un filtre est un objet symbolique. C'est un ensemble d'assertions décrivant les entités qui sont liées par un besoin de communication. Un filtre contient une assertion décrivant les agents récepteurs, une assertion décrivant les messages qu'il doit transmettre, et un ensemble d'assertions décrivant le contexte dans lequel le message doit être transmis.

Cette description en intension peut être rapprochée de la notion de horde en ADS, puisqu'elle concerne des entités décrites par des propriétés différentes, en l'occurrence les agents, les messages et éventuellement d'autres descriptions du contexte.

Définition 4 (Filtre) – *Un filtre $f \in \mathcal{F}$, noté $f(a, m, C)$ avec $a \in \mathcal{A}$, $m \in \mathcal{M}$ et $C \subset \Omega$ est un tuple $\langle f_a, f_m, f_C, name, priority, initiator \rangle$ où :*

- *f_a est la description en intension de l'agent récepteur telle que :*
  $a \in \mathcal{A},\ f_a(a) = \wedge_{p_j \in P_{f_a}} [p_j(a) R^a_{p_j} v^a_{p_j}]$.
- *f_m est la description en intension du message telle que :*
  $m \in \mathcal{M},\ f_m(m) = \wedge_{p_j \in P_{f_m}} [p_j(m) R^m_{p_j} v^m_{p_j}]$.
- *f_C est la description en intension du contexte telle que :*
  $C \subset \Omega, f_C(C) = \wedge_{\forall \omega \in C}\ as_\omega(\omega)$, *avec* $as_\omega(\omega) = \wedge_{p_j \in P_{as_\omega}} [p_j(\omega)\ R^\omega_{p_j}\ v^\omega_{p_j}]$.
- *$name$ est le nom du filtre.*
- *$priority \in IP$ est la priorité du filtre.*
- *$initiator$ est l'initiateur du filtre.*

La description des récepteurs f_a est une assertion fondée sur les propriétés P_{f_a}, avec P_{f_a} l'ensemble des propriétés sur lesquelles portent les tests. En d'autres termes, un agent doit posséder ces propriétés pour être un récepteur potentiel du message. Pour chaque propriété p_j appartenant à P_{f_a}, $R^a_{p_j}$ est un opérateur de comparaison et $v^a_{p_j}$ est une valeur ou une variable.

Par exemple, soit l'assertion $f_a(a) = [age(a) > 18] \wedge [lieu(a) =$ "$Dauphine$"$]$, l'ensemble des propriétés testées P_{f_a} est constitué de $\{age, lieu\}$. R^a_{age} est l'opérateur $>$, et R^a_{lieu} est l'opérateur $=$. Les valeurs v^a_{age} et v^a_{lieu} sont respectivement 18 et "$Dauphine$".

De la même façon que pour la description des agents, la description du message à recevoir est donnée par l'assertion f_m. Le contexte de l'interaction, *i.e.* les autres entités sur lesquelles portent des conditions, est donné par la horde f_C, qui est une conjonction d'assertions as_ω. De ce point de vue, le contexte de la communication est un sous-ensemble de Ω, et donc une partie de l'état $\mathcal{D}$ du système multi-agents.

Une des originalités d'EASI est de permettre l'utilisation de variables dans les assertions. Ces variables permettent de réaliser un appariement entre les valeurs des propriétés de plusieurs entités. Les variables sont préfixées par "?". Par exemple, le filtre fictif $f_{appariement}$ s'écrit

$\langle [p_1(a) =?x],\ [p_2(m) =?x], \emptyset, \text{“}appariement\text{”}, 0, \text{“}a1\text{”}\rangle$. Il permet un appariement de la valeur de la propriété p_1 de l'agent avec la valeur de la propriété p_2 du message grâce à la variable $?x$. Ainsi, un message dont la propriété p_2 est différente de *null* sera reçu par tous les agents dont la valeur de p_1 est égale à la valeur de p_2 du message.

Par défaut, nous considérons qu'une assertion vide est satisfaite par toutes les descriptions. Si la description en intension des agents f_a ou des messages f_m est égale à $\emptyset$, alors ce sont respectivement tous les agents ou tous les messages qui sont concernés par ce filtre.

L'initiateur *initiator* d'un filtre est l'agent qui l'ajoute dans l'environnement. Lorsque l'initiateur du filtre le dépose dans l'environnement, il lui donne un nom *name* et une priorité *priority* dans l'intervalle IP. L'intervalle IP est défini par le concepteur du système multi-agents. La priorité est un nombre, qui définit un ordre sur les filtres dans l'environnement en fonction des besoins de l'initiateur du filtre. Par la suite, la valeur par défaut de la priorité est la valeur médiane de l'intervalle IP.

Exemple cité digitale. Pour illustrer l'utilisation des filtres, nous définissons ici les filtres correspondant à l'exemple de la cité digitale.

Le premier besoin f_{direct} est celui des communications adressées, lorsque l'émetteur connaît l'identité de son interlocuteur. Les messages concernés sont ceux dont la propriété *receiver* a une valeur $?x$, et le récepteur est l'agent dont la propriété *id* est de même valeur $?x$. Le filtre effectue donc un appariement entre ces deux propriétés :
$f_{direct}(a, m, C) = \langle [id(a) =?x], [receiver(m) =?x], \emptyset, \text{“}direct\text{”}, 0, \text{“}a1\text{”}\rangle$

L'assertion sur le contexte est $\emptyset$, ce qui signifie que le filtre est valide dans tous les contextes. Par défaut, nous donnons la valeur de la priorité des filtres à 0.

Le second besoin en communication f_{groupe} est celui de l'agent a_1, qui souhaite envoyer son message à tous les agents disponibles dans un lieu $l1$ donné. Il faut alors caractériser les trois éléments du filtre : les récepteurs concernés, les messages concernés, et le contexte.

Une solution est de donner une propriété *lieu* dans la description du message, qui contient le lieu de destination du message. La syntaxe du filtre dont a_1 est l'initiateur est la suivante :
$f_{groupe}(a, m, C) = \langle [dispo(a) = true] \wedge [lieu(a) =?x], [lieu(m) =?x], \emptyset, \text{“}groupe\text{”}, 0, \text{“}a1\text{”}\rangle$

Avec ce filtre, tous les messages ayant la propriété *lieu* sont transmis aux agents disponibles situés dans le lieu indiqué, grâce à l'appariement avec la propriété *lieu* de l'agent. L'agent a_1 devra alors décrire ses messages par une propriété $< lieu, l1 >$ pour obtenir le comportement désiré.

Par exemple, un message m_1 ayant pour description $< performative, \text{“}propose\text{”} >, < sender, \text{“}a1\text{”} >, < receiver, \text{“}a3\text{”} >, < lieu, \text{“}Dauphine\text{”} >$ sera transmis à l'agent a_3 grâce au filtre f_{direct}, et à tous les agents disponibles de l'université Paris-Dauphine grâce au filtre f_{groupe}, en l'occurrence seulement a_4.

Nous avons vu que les agents *organisation* déposent dans l'environnement des annonces d'évènement. Ces annonces sont des messages persistants qui sont récupérés par les agents intéressés. Le premier filtre f_{ballet} de récupération de données est :

$f_{ballet}(a, m, C) = \langle [id(a) = a_x], [type(m) = \text{“}ballet\text{”}] \wedge [lieu(m) = \text{“}opera\ bastille\text{”}], \emptyset, \text{“}ballet\text{”}, 0, \text{“}ax\text{”} \rangle$

Un agent a_x souhaite recevoir toutes les annonces de ballet ayant lieu à l'opéra Bastille. Cet agent dépose alors un filtre f_{ballet} pour lequel il est le seul récepteur, qui teste les valeurs des propriétés *type* et *lieu* des messages liés aux évènements.

Le second filtre de récupération de données $f_{concert}$ prend en compte le contexte de transmission, c'est à dire la description d'autres entités. Il s'agit, pour un agent a_x, de recevoir les évènements de type concert uniquement si ceux-ci ont lieu dans un restaurant :
$f_{concert}(a, m, \{c\}) = \langle [id(a) = a_x], [type(m) = \text{“}concert\text{”}] \wedge [lieu(m) = ?x], [nom(c) = ?x] \wedge [type(c) = \text{“}restaurant\text{”}], \text{“}concert\text{”}, 0, \text{“}ax\text{”} \rangle$

Cette fois, il est nécessaire d'effectuer un appariement sur une entité du contexte, matérialisée dans le filtre par $\{c\}$.

Le dernier besoin en communication f_{ecoute} de l'exemple de la cité digitale est celui d'un agent a_2 ne se situant pas en $l1$ mais souhaitant écouter les messages qui sont envoyés aux agents qui y sont situés. Sa propriété *lieu* étant différente de $l1$, le filtre f_{groupe} ne lui transmet pas ces messages. Nous avons vu que les messages transmis aux agents situés en $l1$ possèdent une propriété *lieu*. Le filtre de l'agent a_2 utilise cette propriété :
$f_{ecoute}(a, m, C) = \langle [id(a) = \text{“}a2\text{”}], [lieu(m) = l1], \emptyset, \text{“}ecoute\text{”}, 0, \text{“}a2\text{”} \rangle$

L'agent avec la propriété *id* de valeur “$a2$” percevra tous les messages dont la valeur de la propriété *lieu* est $l1$.

4.3 Regrouper et caractériser les descriptions des entités

A chaque fois qu'un message est ajouté dans l'environnement, celui-ci doit trouver tous les agents récepteurs. Or, le nombre de descriptions d'entités et de filtres peut être très important, et la recherche des récepteurs devient alors difficile.

L'Analyse de Données Symboliques a pour but de trouver des classes d'entités similaires. En règle générale, les données ne sont classifiées que ponctuellement, et les temps de traitements ne sont pas primordiaux. Dans le cadre des systèmes multi-agents, notre environnement doit effectuer le transfert des messages rapidement. Le stockage des données brutes sous la forme d'une matrice des descriptions est une solution coûteuse : pour chaque message, la recherche des récepteurs nécessite au moins un parcours complet de la matrice de taille $n \times r$, avec n le nombre d'entités et r le nombre de propriétés, pour chaque filtre de $\mathcal{F}$. La possibilité de réaliser des appariements augmente encore la complexité, puisque ce sont plusieurs parcours de la matrice qu'il faut réaliser pour chaque filtre définissant des appariements entre descriptions d'entités.

Pour faciliter la recherche des récepteurs, nous cherchons donc à limiter le nombre de tests à réaliser, en regroupant et caractérisant *a priori* les descriptions des entités.

Nous introduisons dans cette section les catégories d'entités. Ensuite, nous utilisons la définition des filtres pour caractériser les ensembles d'informations qui sont liés à chaque filtre du point

de vue de la communication. Ces ensembles sont ensuite utilisés dans nos algorithmes en section 4.4 pour améliorer l'efficacité des appariements.

4.3.1 Les catégories d'entités

Nous avons vu que tous les composants du système multi-agents sont au même niveau d'abstraction, c'est à dire sont des entités. Nous ne différencions pas *a priori* les agents des messages ou des autres objets. De façon à trouver un premier niveau d'information concernant les classes des entités, nous construisons des catégories d'entités. Pour cela, nous utilisons l'information structurelle d'existence des propriétés observables : si une valeur est définie pour cette propriété, alors nous disons que cette propriété *existe*.

Nous définissons l'ensemble P_ω des propriétés effectivement décrites d'une entité ω tel que : $\omega \in \Omega, P_\omega = \{p_j \in P | p_j(\omega) \neq null\}$

P_ω est l'ensemble des propriétés p_j de l'entité ω dont la valeur est différente de *null*. Par exemple, soit a un agent *utilisateur* de la cité digitale, $P_a = \{id, nom, dispo, age, sexe, lieu\}$ est l'ensemble de ses propriétés.

Nous appelons *Pdescription* de ω l'ensemble de propriétés P_ω. De façon générale, une Pdescription P_i est un sous-ensemble de propriétés de P.

Une *catégorie* est un sous-ensemble d'entités décrites par un même sous-ensemble de propriétés. Formellement, une catégorie Cat est donc une application de l'ensemble des propriétés P vers l'ensemble des parties de Ω telle que :
$P \mapsto \mathcal{P}(\Omega),\ Cat(P_i) = \{\omega \in \Omega | \forall p_j \in P_i, p_j(\omega) \neq null\}$

Une catégorie $Cat(P_i)$ est un sous-ensemble de Ω, elle contient toutes les entités ω telles que leur Pdescription contient P_i.

Dans l'exemple de la cité digitale, la catégorie de P_a est l'ensemble des agents *utilisateur*, puisque tous les agents *utilisateur* possèdent les propriétés $\{id, nom, dispo, age, sexe, lieu\}$ dans leurs descriptions, et seulement les agents *utilisateur* puisque les agents *organisation* ne sont pas décrits par les propriétés $\{dispo, age, sexe, lieu\}$.

La catégorie d'une Pdescription P_i contient toutes les entités possédant au moins les propriétés P_i, ainsi que les éventuelles entités ω_j telle que $P_i \subset P_{w_j}$. Dans l'exemple précédent, $Cat(P_a)$ contient tous les agents *utilisateur*. Si nous définissons une Pdescription $P_\omega = \{id, nom\}$, celle-ci contiendra non seulement les agents *utilisateur*, mais également les agents *organisation* puisque les agents des deux catégories possèdent ces propriétés.

Pour récapituler, un ensemble d'entités peut être défini de deux façons, soit par une Pdescription soit par une assertion. Par exemple, si l'assertion d'un filtre f_a décrivant l'agent est $[id =$ "$a3$"$]$, la Pdescription de f_a est $P_{f_a} = \{id\}$. La catégorie d'une Pdescription contient toutes les entités possédant au moins les propriétés de la Pdescription P_{f_a}. L'extension d'une assertion contient toutes les entités possédant au moins toutes les propriétés testées dans l'assertion P_{f_a} et dont chacune des propriétés satisfait les tests. Par exemple, $E(f_a)$ contient uniquement l'agent ayant sa propriété id égale à "$a3$", tandis que $Cat(P_{f_a})$ contient toutes les entités possédant la

propriété *id*.

Opérations sur les ensembles de propriétés

Nous avons vu qu'un ensemble de propriétés est la description en intension d'une catégorie d'entités, et qu'un filtre doit inclure plusieurs descriptions en intention d'entités : la classe d'agent devant recevoir le message, la classe de messages concernée, et les entités du contexte. La définition d'un filtre donne la Pdescription des catégories d'entités concernées avec P_{f_a}, P_{f_m} et P_{f_C}.

Par exemple, si une connexion est liée à la valeur de la position d'un agent, alors la Pdescription en intension du filtre correspondant P_{f_a} contient la propriété *lieu*. Lorsque la Pdescription du filtre est restreinte à cette seule propriété, alors toutes les entités ayant cette propriété dans leur description, par exemple certains messages, feront également partie de l'extension du filtre dans Ω. La recherche des agents concernés par un filtre ne doit donc être effectuée que parmi l'ensemble $\mathcal{A}$ des agents, soit $\{\omega|\omega \in Cat(P_{f_a}) \cap \mathcal{A}\}$. De la même façon, la recherche des messages concernés par un filtre est effectuée parmi l'ensemble des messages $\mathcal{M}$, soit $\{\omega|\omega \in Cat(P_{f_m}) \cap \mathcal{M}\}$.

A chaque filtre f correspond un tuple $\langle Cat(P_{f_a}) \cap \mathcal{A}, Cat(P_{f_m}) \cap \mathcal{M},\ \langle \forall \omega \in C,\ Cat(P_{as_\omega})\rangle\rangle$. Les deux premiers éléments du tuple sont les agents et messages potentiellement concernés par f. Le troisième élément du tuple est lui-même un tuple, car les conditions sur le contexte d'un filtre forment une horde. Ce tuple contient les contextes potentiels $\langle \forall \omega \in C,\ Cat(P_{as_\omega})\rangle$ de f, autrement dit toutes les combinaisons d'entités dont la description correspond aux propriétés testées dans les assertions as_ω.

Pour simplifier la notation, nous notons dans la suite $Cat(P_{f_a})$ à la place de $Cat(P_{f_a}) \cap \mathcal{A}$, et de la même façon $Cat(P_{f_m})$ à la place de $Cat(P_{f_m}) \cap \mathcal{M}$.

Le tuple $\langle Cat(P_{f_a}), Cat(P_{f_m}),\ \langle \forall \omega \in C,\ Cat(P_{as_\omega})\rangle\rangle$ permet de regrouper *a priori* les entités dont la Pdescription est valide pour le filtre. Les entités pouvant être écartées des tests sont celles qui ne possèdent pas les propriétés P_{f_a}, P_{f_m} ou P_{as_ω}, respectivement pour les agents, messages ou entités du contexte.

Dans l'exemple de la cité digitale, pour le filtre f_{groupe}, les entités concernées par le filtre sont :
$\langle a \in Cat(\{dispo, lieu\}), m \in cat(\{lieu\}), \langle C \in \mathcal{P}(\Omega)\rangle\rangle$

Nous illustrons ces ensembles d'entités grâce à la figure 4.4. L'objectif est de trouver les agents appartenant à $E(f_a)$ qui reçoivent effectivement les messages $E(f_m)$ grâce au filtre f_{groupe}. Deux ensembles d'entités sont représentés : les agents $\mathcal{A}$ et les messages $\mathcal{M}$. Les agents sont soit des agents *utilisateur*, qui possèdent les propriétés *lieu* et *dispo*, soit des agents *organisation*, qui ne possèdent aucune de ces deux propriétés. Certains messages possèdent la propriété *lieu*. Les autres entités ne sont pas représentées car le filtre f_{groupe} ne contient pas de conditions sur le contexte.

Le filtre possède deux assertions, f_a pour les agents, et f_m pour les messages. Si on re-

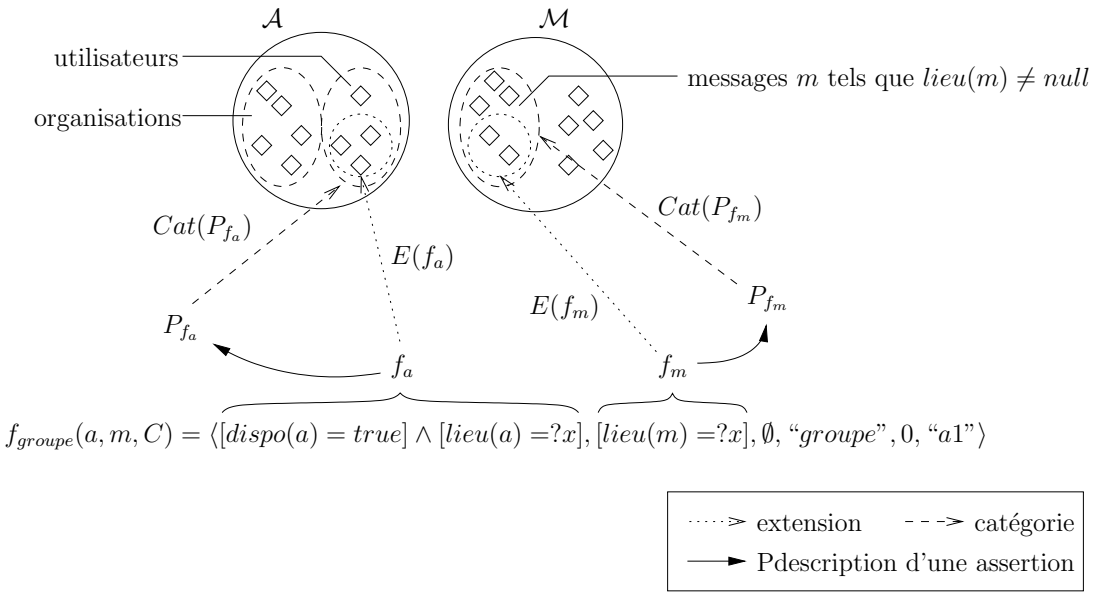


FIGURE 4.4 – Entités liées au filtre f_{groupe}

cherche directement les agents appartenant à $E(f_a)$ et les messages appartenant à $E(f_m)$, il est nécessaire de parcourir les ensembles $\mathcal{A}$ et $\mathcal{M}$. Or, à chacune des assertions f_a et f_m correspond une Pdescription, respectivement P_{f_a} et P_{f_m}. Les ensembles d'entités $Cat(P_{f_a})$ et $Cat(P_{f_m})$ permettent de regrouper les agents et messages possédant les propriétés testées dans le filtre. Les agents n'appartenant pas à $Cat(P_{f_a})$ ne peuvent pas recevoir de message via ce filtre, et les messages n'appartenant pas à $Cat(P_{f_m})$ ne peuvent pas être transmis par ce filtre. Dans le filtre f_{groupe}, $Cat(P_{f_a})$ contient les agents *utilisateur*, et $Cat(P_{f_m})$ contient les messages possédant la propriété *lieu*.

De cette façon, les catégories d'entités peuvent être classifiées *a priori*, grâce à la syntaxe du filtre. Au moment de la transmission des messages, au lieu de rechercher les récepteurs $E(f_a)$ dans $\mathcal{A}$, nous pouvons restreindre cette recherche à $Cat(P_{f_a})$, et de la même façon pour la recherche des messages f_m. Les algorithmes utilisant les catégories pour classifier les entités sont détaillés dans la section 4.4.

Les opérations sur les Pdescriptions sont les opérations ensemblistes. Nous étudions l'impact de ces opérations sur les catégories des Pdescriptions.

La catégorie résultant de l'union des propriétés de deux Pdescriptions P_1 et P_2 est $\omega \in Cat(P_1 \cup P_2) \Leftrightarrow P_\omega \supset (P_1 \cup P_2)$. Une entité appartient à la catégorie de l'union si elle possède au moins les deux ensembles de propriétés d'origine.

Par définition, la Pdescription d'une entité contient toutes les propriétés de cette entité. La catégorie de l'union de deux Pdescriptions dont l'intersection est vide ne contient donc aucune des entités des ensembles initiaux.

Dans l'exemple, nous rappelons que la catégorie des agents *utilisateur* a pour Pdescription

$\{id, nom, age, dispo, lieu\}$, et la catégorie des agents *organisation* $\{id, nom, type\}$.

L'union des Pdescriptions $P_{\cup agents}$ des agents *utilisateur* et *organisation* est donc :

$$P_{\cup agents} = \{id, nom, dispo, age, sexe, lieu\} \cup \{id, nom, type\} \Leftrightarrow$$
$$P_{\cup agents} = \{id, nom, dispo, type, age, sexe, lieu\}$$

La catégorie de $P_{\cup agents}$ ne contient paradoxalement aucun agent. Si l'opération d'union est utile pour obtenir l'ensemble des propriétés utilisées dans le système par certaines classes d'entités, elle ne permet pas de regrouper les entités.

La catégorie résultant de l'intersection des propriétés de deux Pdescriptions P_1 et P_2 est $\omega \in Cat(P_1 \cap P_2) \Leftrightarrow P_\omega \supset (P_1 \cap P_2)$. Une entité appartient à la catégorie de l'intersection si elle possède au moins l'intersection des deux ensembles de propriétés d'origine. La catégorie de l'intersection de deux Pdescriptions contient donc les entités des deux ensembles de départ, ainsi que toute autre entité répondant à cette nouvelle Pdescription.

Dans l'exemple, l'intersection $P_{\cap agents}$ des deux Pdescriptions précédentes est

$$P_{\cap agents} = \{id, nom, dispo, age, sexe, lieu\} \cap \{id, nom, type\} \Leftrightarrow$$
$$P_{\cap agents} = \{id, nom\}$$

La catégorie de $P_{\cap agents}$ contient cette fois tous les agents. L'intersection donne donc l'ensemble des propriétés grâces auxquelles il est possible de contacter à la fois les agents *utilisateur* et les agents *organisation*.

4.3.2 Caractériser les informations du modèle

Dans la section précédente, nous avons catégorisé les entités potentiellement valides pour les filtres. Nous utilisons maintenant les Pdescriptions pour étudier les ensembles d'entités reliant les filtres, les agents et les messages. Dans cette section, nous étudions plus précisément ces ensembles d'informations et nous développons leur signification pour la gestion des besoins des agents. Dans la section suivante, nous voyons leur utilisation dans le processus de sélection des récepteurs.

4.3.2.a Informations relatives aux filtres

Nous définissons le domaine de réception d'un filtre en fonction d'un agent et d'un message comme les catégories des Pdescriptions des assertions f_a et f_m qui composent le filtre f. Le domaine de réception contient donc respectivement les agents et les messages potentiellement concernés par le filtre.

Définition 5 (Domaine de réception d'un filtre) *Pour un filtre f :*

- $Cat(P_{f_a}) = \{a \subset \mathcal{A} | P_a \supset P_{f_a}\}$
- $Cat(P_{f_m}) = \{m \subset \mathcal{M} | P_m \supset P_{f_m}\}$

Le domaine de réception $Cat(P_{f_a})$ est l'ensemble des agents a possédant au moins les propriétés P_{f_a} requises dans la définition du filtre f. De la même façon, le domaine de réception $Cat(P_{f_m})$ est l'ensemble des messages m possédant au moins les propriétés P_{f_m}.

Ces ensembles contiennent les descriptions des entités qu'un filtre regroupe pour produire une interaction, ils peuvent être utilisés par les agents pour composer leur domaine d'interaction. Lorsque $Cat(P_{f_a}) = \emptyset$ ou $Cat(P_{f_m}) = \emptyset$, c'est à dire qu'ils sont vides, il n'y a à cet instant t aucun agent (respectivement message) dans l'environnement qui possède cette Pdescription. Cela signifie soit que le filtre est mal conçu, soit qu'il n'est pas utile à l'instant t : il est périmé, ou les entités que le filtre est destiné à traiter n'ont pas encore de description dans l'environnement.

Dans l'exemple de la cité digitale, pour les deux filtres f_{groupe} et f_{ecoute} [19], $Cat(P_{f_m})$ est l'ensemble des messages ayant la propriété *lieu*. Pour le filtre f_{groupe}, $Cat(P_{f_a})$ est l'ensemble des agents situés dans un lieu et ayant une disponibilité, tandis que pour le filtre f_{ecoute}, $Cat(P_{f_a})$ est l'ensemble des agents car tous les agents possèdent la propriété *id*. Si $E(P_{f_m}) = \emptyset$ à un instant t, alors il n'y a pas de messages concernés par les filtres f_{groupe} et f_{ecoute}.

4.3.2.b Informations relatives aux agents

Pour améliorer la pré-classification des entités, nous définissons les ensembles d'entités qui peuvent être mis en relation avec les agents grâce aux filtres. Grâce aux domaines de réception, nous regroupons les filtres $reception_a$ qui sont liés à une catégorie d'agents. Puis, grâce à cet ensemble de filtres, nous trouvons les messages contenus dans $recept_a$ que ces agents peuvent recevoir en fonction des filtres présents dans l'environnement. L'objectif est de pouvoir trouver rapidement les messages qu'un agent doit recevoir.

Les canaux de réception d'un agent sont l'ensemble des filtres qui le concernent.

Définition 6 (Canaux de réception d'un agent)
$\forall a \in \mathcal{A},\ reception_a = \{f \in \mathcal{F} | a \in Cat(P_{f_a})\}$

Les canaux $reception_a$ sont l'ensemble des filtres $f \in \mathcal{F}$ tels que l'agent a appartient à leur domaine de réception $Cat(P_{f_a})$. Ce sont tous les filtres grâce auxquels la catégorie de l'agent peut recevoir des messages, puisque cette définition utilise les Pdescriptions.

Pour un agent a, $reception_a$ est composé des filtres ajoutés par l'agent lui-même, mais aussi des filtres ajoutés par d'autres agents mais dont il est récepteur potentiel. Pour les filtres qu'il a ajouté lui-même, $Cat(P_{f_m})$ correspond à ses besoins en information en tant que récepteur. Dans le cadre des communications multi-parties, son rôle vis-à-vis du message est alors celui d'écouteur : ces filtres représentent son attention aux messages transmis par l'environnement. Pour les autres filtres, $Cat(P_{f_a})$ et $Cat(P_{f_m})$ représentent les besoins des émetteurs utilisant le filtre. Le rôle de l'agent a est alors celui de destinataire ou d'auditeur, selon son implication dans l'interaction.

19. Rappel :$f_{groupe}(a, m, C) = \langle [dispo(a) = true] \wedge [lieu(a) = ?x], [lieu(m) = ?x], \emptyset, \text{“}groupe\text{”}, 0, \text{“}a1\text{”} \rangle$
$f_{ecoute}(a, m, C) = \langle [id(a) = \text{“}a2\text{”}], [lieu(m) = l1], \emptyset, \text{“}ecoute\text{”}, 0 \rangle$

Nous considérons $reception_a$ comme l'interface qui permet de communiquer avec cet agent : tous les messages que l'agent reçoit transitent par l'un de ces filtres. Cette interface évolue au cours du temps avec l'ajout et le retrait des filtres en fonction des besoins des agents initiateurs de ces filtres.

Dans l'exemple de la cité digitale, si seul f_{groupe} est présent dans l'environnement, tous les agents ayant une propriété *lieu*, autrement dit les agents *utilisateur*, ont $reception_a = \{f_1\}$. Les agents *organisation* ne sont concernés par aucun filtre, donc pour eux $reception_a = \emptyset$.

Les canaux de réception $reception_a$ permettent de connaître tous les filtres par lesquels l'agent a peut recevoir des messages. Le *recept* d'un agent correspond à l'ensemble des messages qu'un agent peut recevoir grâce à ses canaux de réception.

Définition 7 (recept d'un Agent) $recept_a = \{m \in \mathcal{M} | \exists f \in reception_a, m \in Cat(P_{f_m})\}$

Le recept $recept_a$ est l'ensemble des messages $m \in \mathcal{M}$ tels qu'il existe au moins un filtre dans $reception_a$ dont le domaine de réception $Cat(P_{f_m})$ contient m . En d'autres termes, il s'agit de tous les messages concernés par les filtres de $reception_a$.

Si $recept_a$ est vide, alors a est "sourd". Cela peut être un choix, par exemple s'il a une tâche à réaliser et n'a pas besoin de recevoir de messages à ce moment, et si aucun autre agent ne souhaite interagir avec lui. Ce choix est temporaire, puisque l'ensemble est continuellement mis à jour. De plus, il ne peut être considéré comme isolé du point de vue interactionnel puisqu'il peut toujours émettre des messages.

Pour les agents *utilisateur*, $recept_a$ contient tous les messages possédant une propriété *lieu*, tandis que pour les agents *organisation*, $recept_a$ ne contient pas de messages.

4.3.2.c Informations relatives aux messages

De la même façon que pour les agents, le modèle EASI permet d'analyser les entités qui sont liées à un message grâce aux filtres. Nous définissons donc l'ensemble des filtres $channel_m$ qui sont liés à une catégorie de messages. Grâce à cet ensemble, nous pouvons obtenir tous les agents $receiver_m$ susceptibles de recevoir les messages, et tous les contextes $context_m$ dans lesquels la réception peut avoir lieu. Lorsqu'un message est transmis à l'environnement, nous pourrons restreindre la recherche des récepteurs à l'ensemble $receiver_m$, et la recherche d'un contexte valide à l'ensemble $context_m$.

Les canaux d'un message sont l'ensemble des filtres par lesquels il peut être transmis.

Définition 8 (Canaux d'un message) $\forall m \in \mathcal{M}, channel_m = \{f \in \mathcal{F} | m \in Cat(P_{f_m})\}$

Les canaux $channel_m$ sont l'ensemble des filtres $f \in \mathcal{F}$ tels que le message m appartient à leur domaine de réception $Cat(P_{f_m})$.

Si $channel_m$ est vide, alors aucun agent ne recevra ce message, car aucun filtre ne correspond à sa description. Dans ce cas, l'émetteur doit soit ajouter un nouveau filtre qui utilise les propriétés du message, soit modifier le message en conséquence pour obtenir l'adéquation entre propriétés requises et propriétés du message.

Dans l'exemple, avec f_{groupe} et f_{ecoute}, les canaux $channel_m$ de tous les messages possédant la propriété $lieu \in P_m$ sont $\{f_{groupe}, f_{ecoute}\}$. Par contre, les messages ne contenant pas la propriété lieu ($lieu \notin P_m$) ont un $channel_m$ égal à $\emptyset$, puisqu'il n'y a pas de filtre(s) pour les transmettre.

Le domaine de réception d'un message correspond à l'ensemble de ses récepteurs potentiels.

Définition 9 (Domaine de réception d'un message)
$receiver_m = \{a \in \mathcal{A} | \exists f \in channel_m, a \in Cat(P_{f_a})\}$

Le domaine $receiver_m$ est l'ensemble des agents $a \in \mathcal{A}$ tels que l'agent a appartient au domaine de réception $Cat(P_{f_a})$ d'au moins un filtre de $channel_m$.

Si $channel_m$ n'est pas vide, mais que $receiver_m$ l'est, cela signifie que la description du récepteur n'est pas correcte à cet instant t.

Dans l'exemple, pour les messages possédant la propriété *lieu*, $receiver_m$ contient tous les agents possédant les propriétés *lieu* et *dispo* (grâce à f_{groupe}) et tous les agents possédant la propriété *id* (grâce à f_{ecoute}). Cet exemple montre que si les ensembles que nous avons défini par les Pdescriptions permettent effectivement de réduire le nombre d'entités avec lesquelles les filtres doivent être testés, la réduction n'a lieu que si les propriétés ne sont pas partagées par toutes les entités. En l'occurrence, la propriété *id* est partagée par tous les agents, donc un filtre ne testant que la propriété *id* doit être testé avec toutes les entités.

Les contextes de réception d'un message sont l'ensemble des contextes dans lesquels le message peut être transmis.

Définition 10 (Contextes de réception d'un message)
$context_m = \{C \subset \Omega | \exists f \in Channel_m,\ \forall \omega \in C,\ \omega \in Cat(P_{as_\omega})\}$

Nous avons vu que le contexte est un ensemble d'entités. Les ensembles C tels que $\forall \omega \in C,\ \omega \in Cat(P_{as_\omega})$ sont les contextes dont chacune des entités ω possède les propriétés requises P_{as_ω} pour le filtre f. Nous les appelons les contextes *potentiels* de f.

$context_m$ est alors l'ensemble des contextes $C \subset \Omega$ tels que C est un contexte potentiel d'au moins un filtre de $channel_m$. Autrement dit, ce sont tous les ensembles d'entités possédant les Pdescriptions requises pour permettre à un filtre de transmettre le message m.

Si $context_m$ est vide, alors cela signifie que le message ne peut pas être transmis faute de contexte adéquat. Dans l'exemple de cité digitale, seul le filtre $f_{concert}$ [20] définit un contexte. Pour un message annonçant un évènement de type concert, $context_m$ contient toutes les descriptions de lieux possédant les propriétés $\{nom, type\}$.

4.3.2.d Informations relatives aux contextes

Nous avons défini les entités liées respectivement aux agents et aux messages, le dernier élément pour compléter la pré-classification concerne donc le contexte. Il s'agit de trouver les filtres $fContext_C$ pouvant se déclencher dans un contexte donné.

20. Rappel : $f_{concert}(a, m, \{c\}) = \langle [id(a) = a_x], [type(m) = \text{“}concert\text{”}] \wedge [lieu(m) = ?x], [nom(c) = ?x] \wedge [type(c) = \text{“}restaurant\text{”}], \text{“}concert\text{”}, 0 \rangle$

Définition 11 (Canaux d'un contexte) $fContext_C = \{f \in \mathcal{F} | \forall \omega \in C,\ \omega \in Cat(P_{as_\omega})\}$

Les canaux d'un contexte $fContext_C$ sont l'ensemble des filtres $f \in \mathcal{F}$ tels que C est un contexte potentiel de f. Si $fContext_C$ est vide, alors aucun message ne peut être transmis dans ce contexte.

Nous avons vu que si une assertion du filtre est vide, alors l'extension de cette assertion contient toutes les entités. En conséquence, tous les contextes de Ω sont des contextes potentiels des filtres ne comportant pas d'assertion f_C.

Dans l'exemple de cité digitale, pour une description de lieu ω_7, $fContext_C$ est composé de $f_{concert}$, mais aussi de tous les autres filtres.

4.4 Gestion dynamique des connexions

Une des difficultés du problème de connexion est de trouver un algorithme permettant de gérer les interactions quelle que soit la dynamique du système multi-agents. Le critère principal d'évaluation de la dynamique est la fréquence de mise à jour des propriétés.

L'algorithme d'appariement est fondé sur la relation de validité suivante :
$Soit\ a \in \mathcal{A}, m \in \mathcal{M}, C \subset \Omega$
$V : \mathcal{A} \times \mathcal{M} \times P(\Omega) \times \mathcal{F} \rightarrow \{true, false\}$
$V(a, m, C, f) = f_a(a) \wedge f_m(m) \wedge f_C(C)$

Un filtre $f \in \mathcal{F}$ est dit *valide* pour un agent a, un message m et un contexte C si la valeur de $V(a, m, C, f)$ est *true*. La valeur de $V(a, m, C, f)$ dépend de la valeur de vérité des assertions qui composent le filtre pour les entités testées.

A chaque fois qu'une connexion est réalisée, un destinataire reçoit un message. Autrement dit, lorsque $V(a, m, C, f)$ est vraie, l'environnement transmet le message m à l'agent a, ainsi qu'éventuellement un ensemble d'informations sur le contexte de la validation. Cet ensemble, aussi appelé ensemble de réception, est noté K_f, avec $C' \subset C,\ K_f = \{m, C', name\}$. L'ensemble des informations reçues par l'agent en même temps que le message m est composé du nom du filtre $name$, et d'un sous-ensemble C' du contexte, qui est une partie des descriptions des entités de C. Un avantage d'EASI est ainsi que le récepteur peut connaître le contexte dans lequel il reçoit un message, c'est à dire les descriptions ayant déclenché la réception.

Trois évènements peuvent déclencher la transmission d'un message : l'ajout de ce message, la modification d'une description et l'ajout d'un filtre. Nous proposons dans un premier temps l'algorithme général déclenché par l'ajout d'un message, et deux améliorations de cet algorithme. Ensuite, nous voyons les deux algorithmes liés à la modification d'une description et à l'ajout d'un filtre, qui dérivent des mêmes principes.

4.4.1 Algorithme d'appariement initial

L'algorithme 1 est une première solution au problème de gestion de la dynamique. Nous l'exposons de façon à montrer les principes de base utilisés par la suite. Pour chaque message

ajouté dans l'environnement, l'algorithme associe les agents qui sont liés à ce message par des filtres, en fonction du contexte. La réception effective est dénotée par la primitive $receive(a, K_f)$, qui signifie la réception par l'agent a de l'ensemble de réception K_f.

Nous ne faisons aucune hypothèse sur l'architecture des agents. En considérant donc uniquement un ensemble K_a, qui dénote les connaissances privées de l'agent a, la primitive sera représentée algorithmiquement par :
$receive(a, K_f) \Leftrightarrow K_a \leftarrow K_a \cup K_f$

Les informations contenues dans K_f sont ajoutées à l'ensemble des connaissances de l'agent K_a.

Cet algorithme effectue un test systématique de chacun des filtres avec toutes les descriptions dans l'environnement. Il est exhaustif parce que pour chaque message (ligne 1), on vérifie pour chaque agent (ligne 2) et pour chaque contexte (ligne 3) s'il y a un filtre (ligne 4) qui est valide (ligne 5).

Algorithm 1 L'algorithme d'appariement initial

```
1- Pour chaque (m ∈ M)
2-   Pour chaque (a ∈ A)
3-     Pour chaque (C ∈ Ω))
4-       Pour chaque (f ∈ F) faire
5-         Si (V(a, m, C, f)) Alors
6-           receive(a, K_f)
7-         Fin si
8-       Fin pour
9-     Fin pour
10-  Fin pour
11- Fin pour
```

Dans notre scénario exemple, le filtre de l'agent a_1 correspond à une communication de groupe, et celui de l'agent a_2 correspond à l'écoute flottante. Lorsqu'un agent ajoute dans l'environnement un message possédant une propriété *lieu*, l'algorithme est déclenché. Celui-ci teste pour tous les agents la validité de tous les filtres. Lorsque les filtres f_{groupe} et f_{ecoute} sont validés, les agents concernés reçoivent le message. En suivant le même processus, à chaque fois qu'une description est modifiée ou qu'un filtre est ajouté, l'algorithme est à nouveau déclenché.

Ce premier algorithme naïf pose des problèmes d'efficacité. Le calcul systématique de la relation de validation est fondé sur les ensembles $\mathcal{A}$, $\mathcal{M}$, Ω et $\mathcal{F}$, qui peuvent rapidement être grands.

Le calcul le plus complexe est celui de l'ensemble des permutations de $|C|$ éléments de Ω, le nombre d'éléments du contexte. Si Ω contient n entités, il faut calculer $n^{|C|}$ éléments. La complexité de l'algorithme est donc $|A| \times n^{|C|} \times |\mathcal{F}|$ pour chaque message.

Cet algorithme devient très rapidement impossible à calculer en un temps raisonnable, il est

donc nécessaire de réduire les ensembles de recherche. Pour cela, nous utilisons dans les sections suivantes les catégories d'entités et les ensembles définis précédemment.

4.4.2 L'algorithme fondé sur l'organisation statique des descriptions

Pour réduire le nombre de tests, il faut trouver les ensembles les plus petits liés à chaque connexion, et donc à chaque filtre. Une première solution est donc de ne tester la validité des filtres que pour les entités possédant les propriétés requises. Nous avons vu en section 4.3.1 que pour chaque filtre, ces entités sont identifiées par le tuple :
$\langle Cat(P_{f_a}), Cat(P_{f_m}),\ \langle \forall \omega \in C,\ Cat(P_{as_\omega})\rangle\rangle$

Chacun des éléments du tuple est calculable pour une description du système multi-agents donnée, *i.e.* un ensemble de descriptions donné.

Nous avons vu qu'un message peut être reçu par plusieurs agents grâce au même filtre. De même, un même message peut être perçu grâce à plusieurs filtres. Par conséquent, la difficulté est de trouver pour un message m tous les récepteurs potentiels, en fonction des filtres liés à m. Pour cela, nous utilisons les ensembles définis en section 4.3.2 et résumés dans la figure 4.5.

Nom	Définition
$channel_m$	$\{f \in F \mid m \in Cat(P_{f_m})\}$
$receiver_m$	$\{a \in \mathcal{A} \mid \exists f \in channel_m, a \in Cat(P_{f_a})\}$
$context_m$	$\{C \subset \Omega \mid \exists f \in Channel_m,\ \forall \omega \in C,\ \omega \in Cat(P_{as_\omega})\}$
$reception_a$	$\{f \in F \mid a \in Cat(P_{f_a})\}$
$fContext_C$	$\{f \in \mathcal{F} \mid \forall \omega \in C,\ \omega \in Cat(P_{as_\omega})\}$

FIGURE 4.5 – Rappel des définitions des ensembles pour l'appariement statique.

Les canaux d'un message $channel_m$ sont l'ensemble des filtres f liés au message m tels que m appartient à la catégorie $Cat(P_{f_a})$. Pour chaque filtre f dans $channel_m$, on peut calculer l'ensemble des récepteurs potentiels $receiver_m$ et l'ensemble des contextes potentiels $context_m$. $receiver_m$ est l'ensemble des agents appartenant aux catégories $Cat(P_{f_a})$ des filtres appartenant à $channel_m$. $context_m$ est l'ensemble des contextes dont les entités appartiennent aux catégories $Cat(P_{as_\omega})$ des filtres appartenant à $channel_m$. En suivant le même principe, $reception_a$ est l'ensemble des filtres liés à un agent a, c'est à dire que l'agent appartient à la catégorie $Cat(P_{f_a})$ de chacun de ces filtres. $fContext_C$ est l'ensemble des filtres f pour lesquels C est un contexte potentiel de f.

L'algorithme 2 présente la même structure que l'algorithme 1 mais chacun des ensembles initiaux a été réduit aux entités et filtres potentiels. Par exemple, l'ensemble des agents $\mathcal{A}$ est remplacé par $receiver_m$, autrement dit le sous-ensemble des agents possédant les propriétés requises. Pour un message m, un agent $a \in receiver_m$ et un contexte $C \in context_m$, l'ensemble minimal des filtres pouvant effectuer la connexion est $reception_a \cap channel_m \cap fContext_C$.

Cet algorithme limite la recherche d'appariement à l'espace des entités qui ont été classifiées

Algorithm 2 L'algorithme d'appariement statique

1- Pour chaque ($m \in \mathcal{M}$))
2- Pour chaque ($a \in receiver_m$)
3- Pour chaque ($C \in context_m$)
4- Pour chaque ($f \in (reception_a \cap channel_m \cap fContext_C)$)
5- Si ($V(a, m, C, f)$) Alors
6- $receive(a, K_f)$
7- Fin si
8- Fin pour
9- Fin pour
10- Fin pour
11- Fin pour

en fonction de leur Pdescription, ce qui améliore la résolution de la connexion. La complexité de cet algorithme dépend de la taille des ensembles $receiver_m$, $context_m$ et $reception_a \cap channel_m \cap fContext_C$), mais chacun de ces ensembles est respectivement inférieur à $|\mathcal{A}|$, $n^{|C|}$ et $|\mathcal{F}|$.

La valeur des propriétés n'étant pas prise en compte, ce niveau de description n'est pas sensible à la fréquence de mise à jour du système multi-agents : la modification des propriétés des entités ne modifie pas les ensembles calculés. Ceux-ci ne doivent donc être calculés qu'une seule fois.

4.4.3 L'algorithme fondé sur l'organisation dynamique des descriptions

Lorsque les propriétés ont un taux de mise à jour raisonnable, il est possible d'anticiper qu'un sous-ensemble de récepteurs potentiels, au sens possédant les propriétés requises, ne satisfait pas certaines conditions en terme de valeurs. L'algorithme précédant évalue tout de même ces entités. Par exemple, le filtre f_{groupe} contient l'assertion $[dispo(a) = true]$. L'algorithme naïf teste tous les agents, et l'algorithme fondé sur l'organisation statique teste seulement les agents possédant la propriété *dispo*. L'objectif ici est d'utiliser *a priori* les valeurs contenues dans les descriptions, pour réduire les tests à l'ensemble des agents dont la propriété *dispo* est $true$.

Nous proposons donc un nouvel algorithme qui, tout en suivant le même déroulement que le précédent, est fondé non plus sur les Pdescriptions mais sur les extensions des descriptions des entités, *i.e.* $E(f_a)$, $E(f_m)$ et $E(f_C)$ dans Ω. Ces extensions sont rarement calculables entièrement *a priori*, car les appariements réalisés dans les filtres peuvent mettre en correspondance des propriétés de plusieurs ensembles d'entités.

Nous proposons donc de modifier l'algorithme 2 en ôtant des ensembles d'appariement les entités dont la valeur des propriétés ne satisfait pas les conditions des filtres. Sur l'ensemble $channel_m$, la sélection sera donc faite par :
$channel_m^v = \{f \in channel_m | \ \forall p_j \in P_{f_m}[p_j(m) R_i^{f_m} v_i^{f_m}] \neq false\}$

Il s'agit de l'ensemble des filtres f tels qu'aucun des tests de l'assertion f_m de la forme $[p_j(m)R_i^{f_m}v_i^{f_m}]$ ne soit faux. Dans le filtre f_{groupe}, il s'agit de tous les messages possédant la propriété *lieu*.

Le résultat du test $[lieu(m) =?x]$ n'est pas calculable *a priori*, c'est pourquoi la définition de $channel_m^v$ vérifie que les tests ne sont pas faux, au lieu de vérifier s'ils sont vrais.

Par continuité, nous définissons les ensembles restreints de récepteurs potentiels :
$\forall m \in \mathcal{M}, receiver_m^v = \{a \in receiver_m |\ \forall f \in channel_m^v,\ \forall p_j \in P_{f_a}, [p_j(a)R_{p_j}^{f_a}v_{p_j}^{f_a}] \neq false\}$

$receiver_m^v$ contient tous les agents de $receiver_m$ dont les tests sur les propriétés P_{f_a} des filtres de $channel_m^v$ ne sont pas faux. Dans l'exemple, si $channel_m^v = \{f_{groupe}\}$, alors $receiver_m^v$ contient les agents possédant les propriétés *dispo* et *lieu*, et dont la propriété *dispo* est égale à *true*.

Ce calcul peut être fait pour tous les ensembles de l'algorithme précédent, que nous noterons alors avec l'indice v (pour *valide*). L'algorithme 3 utilise ces ensembles restreints.

Algorithm 3 L'algorithme d'appariement dynamique

1- Pour chaque $(m \in \mathcal{M})$
2- Pour chaque $(a \in receiver_m^v)$
3- Pour chaque $(C \in context_m^v)$
4- Pour chaque $(f \in (reception_a^v \cap channel_m^v \cap fContext_C^v)$
5- Si $(V(a, m, C, f))$ Alors
6- $receive(a, K_f)$
7- Fin Si
8- Fin pour
9- Fin pour
10- Fin pour
11- Fin pour

Du point de vue de la complexité de cet algorithme, chaque ensemble le composant est de taille inférieure ou égale à son équivalent dans l'algorithme d'appariement statique (algorithme 2). Il en résulte que le processus d'appariement est plus rapide grâce à un parcours d'ensembles plus petits, par contre le coût de maintenance des ensembles est plus élevé. En effet, lorsqu'un agent met à jour les propriétés d'une entité, la composition des ensembles construits peut être modifiée. Cette entité peut ne plus être potentiellement valide pour un filtre, ou au contraire le devenir.

Les deux algorithmes fondés sur l'organisation des descriptions ont été implémentés et testés. Les résultats et l'interprétation des tests sont présentés en section 7.2.

4.4.4 Algorithme de gestion des modifications de descriptions

Nous avons vu en détail les algorithmes dédiés à la gestion de l'ajout d'un message dans l'environnement. Nous présentons dans cette section l'algorithme de gestion des modifications de descriptions. Nous présentons cet algorithme et le suivant avec les ensembles "statiques" du système multi-agents, mais il est possible de transposer ces algorithmes avec les ensembles "dynamiques".

Lorsque la description d'une entité ω est modifiée, certains filtres peuvent devenir valides. Le premier pas de l'algorithme 4 concerne le type de description : si la description modifiée est celle d'un message, alors l'algorithme général (2) est déclenché.

Si l'entité n'est pas un message mais que c'est un agent (ligne 3), alors on recherche pour cet agent tous les messages appartenant à son *recept*, autrement dit les messages qu'il peut potentiellement recevoir (ligne 4). Le bloc continue (lignes 5 à 13) par la recherche des contextes et filtres valides pour cette description de l'agent.

La fin de l'algorithme (lignes 14 à 24) prend en compte le cas où la description de l'entité apparaît dans un contexte d'un ou plusieurs filtres. Ce cas est possible quel que soit le type d'entité, agent, message ou objet. Dans l'exemple de la cité digitale, l'entité du contexte testée dans le filtre f_{groupe} est un agent.

Les filtres pouvant devenir valides sont ceux dont au moins une des trois catégories $\langle a \in Cat(P_{f_a}), m \in Cat(P_{f_m}), C \in \langle \forall \omega \in C,\ Cat(P_{as_\omega})\rangle\rangle$ (section 4.3.1) est modifiée par le changement de description ω.

Une fois que l'ensemble de messages pouvant potentiellement être transmis est calculé, l'algorithme reprend le fonctionnement général pour chacun d'eux.

4.4.5 Algorithme de gestion des ajouts de filtres

Le dernier algorithme que nous proposons est celui déclenché lors de l'ajout d'un filtre par un agent.

Cet algorithme est évident, puisque nous avons caractérisé les différents ensembles composant les filtres. Les catégories liées au filtre donnent les entités dont les propriétés sont compatibles $\langle a \in Cat(P_{f_a}), m \in Cat(P_{f_m}), C \in \langle \forall \omega \in C,\ Cat(P_{as_\omega})\rangle\rangle$ (section 4.3.1).

L'algorithme 5 recherche parmi ces entités pour quels messages le nouveau filtre est valide.

4.5 Conclusion

Dans ce chapitre, nous avons introduit le modèle EASI et sa formalisation. Il s'agit d'un environnement de communication actif qui repose sur la notion de coordination fondée sur les propriétés.

Le modèle EASI s'appuie sur une description des éléments du système multi-agents au sein de l'environnement. Cette description peut être utilisée par chacun des agents pour exprimer ses besoins en communication. Ces besoins, modélisés sous forme de filtres, sont aussi bien des

Algorithm 4 L'algorithme de gestion des modifications de descriptions

1- Si $\omega \in \mathcal{M}$ Alors
2- Algorithme 2
3- Sinon Si $\omega \in receiver_m$ Alors
4- Pour chaque $m \in recept_\omega$
5- Pour chaque $(C \in context_m)$
6- Pour chaque $(f \in (reception_\omega \cap channel_m \cap fContext_C))$
7- Si $(V(\omega, m, C, f))$ Alors
8- $receive(\omega, K_f)$
9- Fin si
10- Fin pour
11- Fin pour
12- Fin pour
13- Fin Si

14- Pour chaque (m tel que $\omega \in C, C \in context_m$)
15- Pour chaque $(a \in receiver_m)$
16- Pour chaque $(C \in context_m$ tel que $\omega \in C)$
17- Pour chaque $(f \in (reception_a \cap channel_m \cap fContext_C))$
18- Si $(V(a, m, C, f))$ Alors
19- $receive(a, K_f)$
20- Fin si
21- Fin pour
22- Fin pour
23- Fin pour
24- Fin pour

besoins en émission, que des besoins en réception. Chaque communication est multi-parties car elle prend en compte tous les besoins des agents pour la sélection des récepteurs.

Nous avons choisi la formalisation issue de l'analyse des données symboliques parce qu'elle offre trois avantages. Elle permet de manipuler des données de tous types (quantitatives, qualitatives, intervalles ou ensembles finis), et donc d'exprimer et décrire les entités du système multi-agents de façon expressive. La description des filtres et entités est unifiée, et permet de classer les descriptions efficacement pour la sélection des récepteurs. Enfin, dans un soucis de faciliter l'exploitation du modèle, la formalisation des filtres est exprimable avec des formules de logique du premier ordre.

Contrairement à la majorité des middlewares et espaces de tuples existants (voir par exemple [Julien et Roman, 2004; Omicini et Zambonelli, 1999; Schelfthout et al., 2006]), l'utilisation de variables permet un appariement entre les valeurs des descriptions des entités. Ceci permet

Algorithm 5 L'algorithme de gestion des ajouts de filtres

Soit $Permut(E)$ l'ensemble des permutations de E
1- Pour chaque $(a \in Cat(P_{f_a}))$
2- Pour chaque $(m \in Cat(P_{f_m}))$
3- Pour chaque $(C \in Permut(\forall \omega \in C,\ Cat(P_{as_\omega}))$
4- Si $(V(a, m, C, f))$ Alors
5- $receive(a, K_f)$
6- Fin si
7- Fin pour
8- Fin pour
9- Fin pour
10- Fin pour

de prendre en compte le contexte de façon flexible et dynamique, mais le coût de traitement de l'appariement est important. Pour améliorer l'efficacité de la recherche des récepteurs, nous avons proposé de pré-classifier les entités en catégories selon leurs propriétés. Nous avons ensuite défini les ensembles d'entités pouvant être mises en relation par les filtres.

La transmission d'un message peut être causée par son propre ajout dans l'environnement, par la modification d'une description, ou par l'ajout d'un filtre. Nous avons proposé les algorithmes de gestion de la transmission des messages fondés sur les ensembles précédents.

Dans le chapitre suivant, nous complétons ce modèle pour la régulation des communications.

Chapitre 5

Faciliter et réguler l'interaction : l'environnement comme régulateur de l'interaction

Sommaire

Le modèle décrit jusqu'ici permet aux agents d'agir sur leurs canaux de réception en fonction de leurs besoins. C'est suffisant dans le cadre d'agents coopératifs. Un exemple de règle, dans le cadre des modèles d'agents situés, est que deux agents ne peuvent pas communiquer s'ils sont trop éloignés. Cependant, la conformité du système multi-agents à ces règles est confiée au concepteur des agents.

Dans le cadre de systèmes ouverts et hétérogènes, c'est à dire dont les agents ne peuvent être vérifiés et peuvent provenir de plusieurs concepteurs, cette conformité implicite n'est plus suffisante. Le modèle EASI permet de choisir les récepteurs des messages de façon flexible, mais

ne vérifie pas l'adéquation des filtres à des règles du système multi-agents. L'environnement médiant les communications, il est cependant possible de contrôler lors de la transmission que le choix des récepteurs est compatible avec ces règles fixées soit au niveau du système multi-agents, soit au niveau des agents eux-mêmes :

- Contrôle multi-agents : les communications prenant place dans le système multi-agents doivent pouvoir être restreintes, autrement dit tous les agents n'ont pas nécessairement le droit ou la possibilité de recevoir tous les messages. La règle indiquant que si deux agents sont trop éloignés, ils ne peuvent alors pas recevoir leurs messages respectifs, peut être une règle du système multi-agents.
- Contrôle agent : un récepteur donné est choisi soit par le biais d'un filtre qu'il a lui-même déposé, ou par le biais d'un autre filtre. Dans ce second cas, il doit pouvoir refuser les messages qui ne l'intéressent pas. Par exemple, un agent peut souhaiter ne pas recevoir de messages lorsqu'il est occupé.

Dans l'exemple de cité digitale, chaque agent peut écouter n'importe quel message par le biais de filtres. Ainsi, les messages ne peuvent pas être "privés", dans le sens où les émetteurs ne peuvent pas empêcher les autres agents d'accéder à leurs messages. Un certain nombre de situations non souhaitables peuvent être mises en oeuvre par les agents, par exemple un *utilisateur* qui intercepte les communications de service entre agents *organisation*.

Dans ce chapitre, nous étendons le modèle EASI pour gérer la régulation du système multi-agents : l'Environnement Actif comme Régulateur de l'Interaction (EARI [21]). Ces travaux ont donné lieu à publication dans [Saunier et al., 2007] et [Saunier et Balbo, 2008]. Dans la première section, nous abordons la gestion des accès à l'environnement. Nous différencions deux types de filtres, ceux qui représentent les besoins d'un agent et ceux qui sont communs à l'ensemble des agents. Cette différenciation nous permet de distinguer la problématique multi-agents, *i.e.* les règles communes, de la problématique agent, *i.e.* les besoins particuliers de chaque agent. Ensuite, nous caractérisons de façon plus précise notre approche du contrôle des accès aux descriptions et aux filtres par les agents.

Dans la deuxième section, nous introduisons les *filtres négatifs* qui, à l'inverse des filtres étudiés au chapitre précédent, impliquent la non-réception des messages lorsqu'ils sont valides. Ainsi, les filtres peuvent correspondre soit à une règle pour recevoir certains message, soit à une règle pour ne pas les recevoir : le contrôle est effectué par le biais des filtres négatifs. Cependant, l'introduction des filtres négatifs peut entraîner une incohérence dans les règles, dans le cas où deux filtres concernant le même message et le même agent mais aux effets opposés sont valides. Nous voyons dans la dernière section les différents cas de conflits pouvant apparaître : conflits entre les règles de l'environnement et les besoins des agents, conflits entre filtres de plusieurs agents. Nous exposons alors différentes politiques permettant de gérer ces conflits.

21. Environment as Active Regulator of Interaction

5.1 La gestion des accès à l'environnement

5.1.1 L'initiateur du filtre

Nous avons vu que les agents retirent et ajoutent des filtres en fonction de leurs besoins en interaction. Or, les agents peuvent avoir des besoins en commun. Par exemple, tous les agents peuvent avoir besoin des communications adressées. Il est possible d'abstraire ces besoins communs en tant que services d'interaction offerts par l'environnement lui-même. Par exemple, la voix est portée par l'air, qui a des propriétés physiques. Dans le cadre logiciel, cela peut être considéré comme une règle de comportement de l'environnement. Par exemple, le filtre de communication adressée fondé sur l'identifiant de l'agent et sur le récepteur souhaité du message est un filtre qui concerne tous les agents. C'est un service de l'environnement : tous les messages sont traités de la même façon pour tous les agents.

Dans le modèle EASI, nous proposons de partitionner les filtres en deux catégories, suivant leur initiateur : un agent, ou l'environnement. Nous considérons que les filtres dédiés à la gestion du système multi-agents appartiennent à l'environnement. Fonctionnellement, ils seront fixés par le concepteur du système multi-agents, gérés par un ou plusieurs agents systèmes, ou gérés par un mécanisme interne à l'environnement. En conséquences, les filtres de l'ensemble $\mathcal{F}$ sont : $\mathcal{F} = \mathcal{F}_E \cup \mathcal{F}_A$ avec :

- $\mathcal{F}_E = \{f \in \mathcal{F} | initiator = environnement\}$ l'ensemble des filtres ajoutés par ou pour l'environnement,

- $\mathcal{F}_A = \{f \in \mathcal{F} | initiator \in \mathcal{A}\}$ l'ensemble des filtres des agents.

$\mathcal{F}_E$ contient les filtres qui sont les règles de l'environnement, elles définissent la politique d'interaction du système multi-agents, par exemple une transmission de message standard pour certains types de messages. De cette façon, l'environnement peut ajouter de nouveaux canaux de réception aux agents, par lesquels ils recevront des messages qu'ils n'auraient pas reçu sinon. Par exemple, grâce au filtre gérant les interactions directes, tous les agents reçoivent nécessairement les messages qui leur sont adressés sans avoir à poser eux-mêmes le filtre de réception correspondant.

Dans le cadre de l'étude des communications multi-parties, cela subdivise le rôle d'auditeur en deux cas, celui où la réception est due à l'initiative de l'émetteur et celui ou la réception est due à une initiative -en fait, une règle- de l'environnement.

Les avantages de cette approche fondée sur un environnement régulateur sont donc :

- Un contrôle indépendant des agents : les règles sont gérées au sein de l'environnement de transmission, et non de façon externe (par la surveillance, par exemple). Ainsi, l'environnement régule les communications par un contrôle en temps réel des actions, au lieu d'agir *a posteriori* par des sanctions.
- Une simplification accrue des agents : avec EASI, les tâches de transmission et de résolution du problème de connexion sont déléguées à l'environnement. L'ajout de règles standard de transmission des messages dont l'agent n'a pas à s'occuper participe à cette simplification.

5.1.2 Le contrôle des accès aux descriptions et aux filtres

Nous avons vu dans le chapitre 3 que les agents peuvent modifier et supprimer les descriptions qu'ils ont ajouté dans l'environnement, ainsi que les filtres dont ils sont les initiateurs. Nous précisons ici le cadre de ce choix, et de quelle manière l'introduction de l'environnement comme initiateur de filtres s'y insère.

Il existe trois principales familles de modèles pour le contrôle d'accès [Osborn, 1997; Samarati et di Vimercati, 2001] : le contrôle d'accès obligatoire, le contrôle d'accès discrétionnaire et le contrôle d'accès à base de rôles. Il s'agit de contrôler les accès d'un *sujet* à un *objet*.

Le contrôle d'accès obligatoire est utilisé lorsque la sécurité des informations impose que les décisions de protection ne soient pas prises par le propriétaire des objets. Les décisions de protection sont imposées par le système lui-même, en fonction de la nature de l'information (objet) et du sujet cherchant à y accéder. Le contrôle d'accès discrétionnaire est au contraire fondé sur le propriétaire de l'information. Celui-ci décide quels sont les droits d'accès à "ses" objets, en fonction de l'identité ou du groupe du sujet. Enfin, dans les systèmes de contrôle d'accès à base de rôles, un ou plusieurs rôles sont attachés à chaque sujet. A chaque rôle est associé un ensemble de permissions d'accès aux objets.

Le contrôle d'accès obligatoire nécessite que le système soit capable de classifier les informations ; cette classification ne tient pas compte des besoins de leurs propriétaires. Nous considérons au contraire que les agents sont prioritaires pour accéder à leurs propres informations.

Dans le cadre d'EASI, tous les agents ont le droit d'ajouter de nouvelles descriptions ou filtres, dont ils sont propriétaires. Pour les modifications et retraits, notre approche est fondée sur le contrôle d'accès discrétionnaire : l'agent est responsable des objets dont il est propriétaire, et ses objets ne peuvent pas être modifiés par les autres agents. L'environnement joue deux rôles relatifs aux accès. En tant que système, il assure le contrôle des accès des différents agents. En tant que sujet -nous avons vu qu'il peut être initiateur de filtres- il possède les même droits que les autres agents.

Il est possible de raffiner la gestion des accès dans les systèmes multi-agents grâce au contrôle d'accès à base de rôles, qui généralise le contrôle d'accès discrétionnaire, voir par exemple [Cremonini et al., 2000]. Dans cette thèse, nous avons choisi de focaliser notre étude sur les communications et non sur l'accès aux descriptions ; nous discutons en détail la gestion des filtres dans la section 5.3 tandis que l'approfondissement du contrôle des accès aux descriptions du modèle EASI est envisagé dans les perspectives.

5.2 Contrôle des communications

5.2.1 L'environnement actif comme régulateur de l'interaction

Nous avons vu que les règles de l'environnement peuvent édicter une obligation de recevoir une information en fonction d'un certain nombre de conditions. Si ces règles peuvent s'exprimer de façon positive, elles peuvent aussi le faire de manière négative, au niveau de l'environnement,

d'une part, et au niveau de l'agent, d'autre part. L'environnement définit des règles de portée des messages. Au niveau de l'agent, il s'agit de la décision d'action, décision qui peut être d'écouter, c'est à dire de porter une initiative, ou au contraire de limiter son attention aux stimuli.

Pour exprimer cela, nous définissons un nouveau modèle d'environnement, l'*Environnement Actif comme Régulateur de l'Interaction* (EARI), qui étend le modèle EASI.

Définition 12 – *L'environnement $\mathcal{E}$ est un quadruplet $\langle \mathcal{D}, \mathcal{F}, IP, \mathcal{R} \rangle$, avec :*
- *$\mathcal{D}$ l'ensemble des descriptions des entités*
- *$\mathcal{F}$ l'ensemble des filtres positifs et négatifs*
- *IP l'intervalle des priorités, avec $IP \subset \mathbb{N}$*
- *$\mathcal{R}$ une application de $\mathcal{F}$ vers $I(IP)$ l'ensemble des intervalles de IP, donnant les politiques de priorités liées aux filtres*

Dans le modèle EARI, nous proposons deux améliorations au modèle EASI : un nouveau type de filtres, et l'ajout de politiques de priorités. Dans la section suivante, nous introduisons les filtres *négatifs*, dont l'objectif est de contrôler les communications, puis nous détaillons leurs conséquences sur le fonctionnement du modèle EARI. Dans la section 5.3, nous définissons l'application $\mathcal{R}$ qui permet la mise en oeuvre de politiques de contrôle des communications.

5.2.2 Filtres négatifs

Pour exprimer les besoins de contrôle des communications, nous introduisons les filtres négatifs, que nous notons f^-. La différence avec les filtres du modèle EASI, que nous appellerons par opposition filtres positifs, se situe au niveau du traitement effectué : lorsqu'un filtre négatif est valide, le message concerné ne peut pas être perçu par l'agent.

L'ensemble des filtres négatifs est noté $\mathcal{F}^- \subset \mathcal{F}$, par symétrie l'ensemble des filtres positifs est noté $\mathcal{F}^+ \subset \mathcal{F}$, avec $\mathcal{F} = \mathcal{F}^- \cup \mathcal{F}^+$ et $\mathcal{F}^- \cap \mathcal{F}^+ = \emptyset$. Un filtre est soit positif, soit négatif.

Par exemple, dans le cadre de l'exemple de la cité digitale, nous souhaitons assurer la possibilité de transmettre des messages privés, dans le sens où ceux-ci ne peuvent pas être écoutés par d'autres agents.

Le support des messages privés nécessite deux filtres, (i) pour assurer la transmission effective du message à son destinataire, et (ii) pour assurer qu'il ne sera pas écouté par d'autres agents. Les messages privés ont une propriété observable nommée *private*, qui est booléenne. Une solution est de s'appuyer sur le filtre f^+_{direct} (i), qui assure la transmission du message au destinataire. Nous rappelons sa syntaxe :
$f^+_{direct}(a, m, C) = \langle [id(a) = ?x], [receiver(m) = ?x], \emptyset, \text{“}direct\text{”}, 0, environnement \rangle$

Il suffit alors d'ajouter un filtre pour gérer la problématique (ii), par exemple le suivant :
$f^-_{private}(a, m, C) = \langle [id(a) = ?x], [receiver(m) \neq ?x] \wedge [private(m) = true], \emptyset, \text{“}private\text{”}, 0, environnement \rangle$

Le filtre $f^-_{private}$ concerne uniquement les messages privés grâce à la condition $[private(m) = true]$. Il est valide pour tous les agents qui ne sont pas désignés par la propriété *receiver* du message.

5.2.3 Algorithme générique de gestion des filtres négatifs

Nous avons choisi d'assurer la gestion des filtres négatifs par une modification de l'algorithme de traitement des messages, que nous présentons ici.

Algorithm 6 Algorithme d'appariement (filtres négatifs).

$Filters(a, m, C)$: liste des éléments f de $reception_a \cap channel_m \cap fContext_C$, ordonnés par priorité décroissante
$agent - suivant$: booléen

1- Pour chaque ($m \in \mathcal{M}$)
2- Pour chaque ($a \in receiver_m$)
3- $agent - suivant \leftarrow false$
4- Pour chaque ($C \in context_m$) et tant que $agent - suivant = false$
5- Pour f dans $Filters(a, m, C)$ et tant que $agent - suivant = false$
6- Si ($V(a, m, C, f)$) Alors
7- Si ($f \in \mathcal{F}^-$) Alors
8- $agent - suivant \leftarrow true$
9- Sinon
10- $receive(a, K_f)$
11- Fin si
12- Fin si
13- Fin pour
14- Fin pour
15- Fin pour
16- End For

Les filtres positifs et négatifs ont un effet contradictoire. A un instant t, s'il existe $f^- \in \mathcal{F}^-$ et $f^+ \in \mathcal{F}^+$ tels que $V(a, m, C, f^-) \wedge V(a, m, C, f^+)$, alors l'agent a doit à la fois recevoir le message m et ne pas le recevoir. La décision est effectuée grâce aux priorités des filtres. Ceux-ci sont traités par ordre de priorité décroissant : lorsqu'un filtre négatif est valide pour un couple agent/message, l'algorithme d'appariement 6 passe à l'agent suivant sans traiter les autres filtres, et donc sans permettre la réception par cet agent. Du point de vue de sa complexité, cet algorithme est équivalent à celui présenté en section 4.4.2.

Nous étudions les effets contradictoires que peuvent avoir deux filtres valides. Nous rappelons que $E(f_a)$ est pour le filtre f l'extension de l'assertion f_a concernant les agents. Soient deux filtres quelconques f et g, si le filtre f est valide, alors pour tout filtre g tel que $priority(g) < priority(f)$, le résultat de l'extension calculée par l'algorithme pour ce filtre g est $E(g_a) \setminus E(f_a)$ (l'ensemble $E(g_a)$ privé de $E(f_a)$).

En conséquence, l'effet contradictoire d'un filtre f sur un autre filtre g avec $priority(f) >$

$priority(g)$ dépend du rapport entre leurs extensions, les ensembles $E(f_a)$ et $E(g_a)$:

- $E(f_a) \cap E(g_a) = \emptyset$: si l'intersection des extensions de f_a et de g_a est vide, alors f n'a aucun effet sur g.
- $E(f_a) \supseteq E(g_a)$: si l'extension de g_a est incluse dans l'extension de f_a, alors on dit que f *couvre* g. g ne s'exprime donc jamais.
- $E(f_a) \cap E(g_a) \neq \emptyset$: si l'intersection des extensions de f_a et de g_a est non vide, alors on dit que f *couvre partiellement* g. Seuls les agents appartenant à l'intersection des extensions $E(f_a) \cap E(g_a)$ sont affectés par f.

Il est important de noter que les extensions $E(f_a)$ et $E(g_a)$ sont dynamiques, et qu'en conséquence les effets d'un filtre sur un autre filtre peuvent évoluer au cours du temps.

Exemple cité digitale Nous reprenons les deux filtres f^+_{direct} et $f^-_{private}$. Lors du traitement du filtre par l'algorithme, le filtre f^+_{direct} assure la transmission de tous les messages, privés ou non, au destinataire désigné par la propriété *receiver*. Le filtre $f^-_{private}$ concerne tous les agents qui ne sont pas désignés par la propriété *receiver* du message.

Puisque $f^-_{private}$ est un filtre négatif, les agents qui ne sont pas désignés comme destinataire dans le message ($[receiver(m) \quad \neq ?x]$) ne reçoivent pas le message. De plus, l'algorithme ne traitera pas d'autres filtres positifs ou négatifs pour cet agent. Ainsi, suite à un filtre négatif valide, les agents concernés ne peuvent plus recevoir le message.

5.2.4 Résolution des conflits

Si un filtre positif et un filtre négatif pour un même couple message/agent existent simultanément dans l'environnement, c'est l'ordre des priorités qui détermine la réception ou la non-réception.

Un conflit émerge si deux filtres, l'un positif et l'autre négatif, ont tous deux la même priorité et sont valides pour un même agent, par exemple lorsque deux agents posent des filtres contradictoires. Formellement, cela signifie :
$\exists f^- \in \mathcal{F}^-,\ \exists f^+ \in \mathcal{F}^+,\ V(a,m,C,f^-) \wedge V(a,m,C,f^+)$ et $priority(f^-) = priority(f^+)$

La résolution du conflit ayant lieu lors de l'exécution de l'algorithme, elle dépend de l'ordre de traitement des filtres. Le critère premier du tri est la priorité, le critère secondaire pour des filtres de même priorité sera le type de filtre à traiter en premier. Le choix doit prendre en compte deux critères :

- *Restrictivité* : l'interaction est-elle favorisée, auquel cas les filtres positifs sont prioritaires, ou doit-elle être restreinte, auquel cas les filtres négatifs sont prioritaires ? C'est avant tout un critère qualitatif, dépendant de l'application et du comportement souhaité.
- *Efficacité* : un filtre négatif induit un coût moindre qu'un filtre positif, du point de vue de l'algorithme (passage à l'agent suivant) et de la transmission elle-même. Le gain est réalisé pour chaque agent dont un filtre négatif est valide $|\{a \in A | \exists f^-,\ V(a,m,C,f^-)\}|$. Avec l'ensemble des permutations de $|C|$ éléments de Ω, le gain est au pire 0 si le filtre

négatif est le dernier traité. Au mieux, le gain est de l'ordre de $n^{|C|} \times |Filters(a, m, C)|$, qui représentent les pas (4) et (5) de l'algorithme, avec $n^{|C|}$ le nombre de contextes et $|Filters(a, m, C)|$ le nombre de filtres.

5.3 Politiques de priorité

Si un filtre positif et un filtre négatif pour un même couple message/agent existent simultanément dans l'environnement, c'est l'ordre des priorités qui détermine la réception ou la non-réception. Par extension, le choix des priorités relatives affecte le comportement général du système multi-agents.

Dans l'exemple, la seule présence de filtres négatifs ne suffit pas à assurer le contrôle. En effet, puisque le traitement des filtres est effectué par ordre de priorité décroissant, un filtre négatif n'a d'effet que si aucun filtre positif de priorité supérieure ne s'est déclenché.

Par exemple, prenons le cas où un agent a_x dépose le filtre suivant :
$f^+_{interception}(a, m, C) = \langle [id(a) = \text{“}ax\text{”}], [private(m) = true], \emptyset, \text{“}interception\text{”}, 1, a_x \rangle$

L'agent a_x souhaite recevoir tous les messages privés de l'environnement, qu'ils lui soient destinés ou non. Sa priorité est de 1, et donc supérieure à celle de $f^-_{private}$. Dans l'ordre de déroulement de l'algorithme, le filtre $f^+_{interception}$ est testé avant le filtre $f^-_{private}$, l'agent a_x reçoit donc bien les messages privés, contrairement à l'objectif initial.

Grâce à l'algorithme et au tri des filtres en fonction de leur priorité, l'environnement les traite toujours dans le même ordre. Ainsi, la décision relative à la réception d'un message par un agent n'est pas ambiguë. Mais cet exemple montre qu'il peut être nécessaire de contrôler également les priorités des filtres.

Nous étudions alors les choix de priorité, et leur impact sur le déroulement des communications.

5.3.1 Définition d'une politique de priorités

Nous appelons par la suite la gestion des priorités une *politique*. Plus précisément, une politique détermine quelles priorités peuvent être données aux filtres, et donc leur ordre de traitement par l'algorithme. Les politiques sont définies au niveau de l'environnement par l'application $\mathcal{R}$.

Définition 13 *$\mathcal{R}$ est une application de $\mathcal{F}$ vers $I(IP)$, avec $I(IP)$ l'ensemble des intervalles de IP, telle que $\mathcal{R}(f) = [min, max]$*

L'application $\mathcal{R}$ associe à chaque filtre f un intervalle de priorités $[min, max]$ dans l'ensemble des priorités autorisées dans la définition de l'environnement IP. L'intervalle $[min, max]$ définit les priorités autorisées pour le filtre f.

Nous avons vu qu'un objectif du modèle EARI est d'admettre la présence d'agents hétérogènes dans le système tout en contrôlant les communications. Pour cela, les politiques sont décidées par le gestionnaire de l'environnement. Il s'agit en fait du concepteur au niveau du système

multi-agents, qui met en place les règles de l'environnement, par opposition aux concepteurs des agents.

Les politiques sont vérifiées à l'exécution par l'environnement, mais les priorités sont choisies par les agents. Les politiques en vigueur sont connues par tous les agents participant à l'environnement. Lorsqu'un agent ajoute un filtre dans l'environnement, celui-ci vérifie sa validité au regard des politiques grâce à l'application $\mathcal{R}$. L'environnement refuse les filtres qui ne sont pas valides.

Au regard des politiques de contrôle d'accès que nous avons évoquées dans la section 5.1.2, nos politiques de priorités se rapprochent du contrôle d'accès à base de rôles. Les sujets sont les agents qui déposent les filtres, et les objets sont les intervalles de priorité. Les intervalles autorisés associés à chaque filtre par l'application $\mathcal{R}$ ne sont pas choisis par les propriétaires, mais par le concepteur. De plus, ces intervalles peuvent dépendre non seulement de l'identité du propriétaire de ce filtre, mais aussi de ses propriétés.

Nous détaillons dans les sections suivantes plusieurs politiques de priorité. Les politiques relatives aux règles de l'environnement sont fondées sur l'initiateur du filtre, et les politiques relatives aux filtres des agents sont fondées sur les propriétés de l'agent et du filtre lui-même.

5.3.2 Politiques relatives aux règles de l'environnement

Le premier type de politique à étudier est celui du rapport entre les filtres de l'environnement et les filtres des agents. Nous distinguons alors deux types de politique au niveau multi-agents (non exclusifs entre eux) : prédominance de l'environnement et environnement facilitateur.

5.3.2.a Prédominance de l'environnement

La première politique est de faire primer les filtres de l'environnement sur ceux des agents.

Politique 1 (Prédominance de l'environnement) $\mathcal{R} : \mathcal{F} \mapsto I(IP)$

$$\mathcal{R}(f) = \left\{ \begin{array}{ll} [min_e, max_e] & si\ initiator = environnement \\ [min_a, max_a] & si\ initiator \in \mathcal{A} \end{array} \right. \Bigg| \ avec\ max_a < min_e$$

Autrement dit, tous les filtres de l'environnement ont une priorité plus haute que tous les filtres des agents (figure 5.1). D'après l'algorithme 6, cela signifie que, quels que soient les filtres des agents, ils ne sont traités qu'après les filtres de l'environnement pour chaque message.

Fonctionnellement, l'environnement autorise des intervalles de priorité différents pour les filtres des agents $[min_a, max_a]$ et pour les filtres de l'environnement $[min_e, max_e]$. Avec la politique 1, la borne max_a est strictement inférieure à la borne min_e, pour éviter le risque de conflit entre des filtres des agents et des filtres de l'environnement de même priorité. Si un agent tente d'ajouter un filtre $f \in \mathcal{F}_A$ mais que $priority(f) \notin [min_a, max_a]$, le filtre est refusé.

Avec cette politique, il y a donc trois cas de figure pour un couple message/agent :

1. Dans le cas où il n'y a pas de filtres de l'environnement, les filtres des agents déterminent la réception. L'algorithme effectue la recherche des agents dans $\mathcal{A}$.

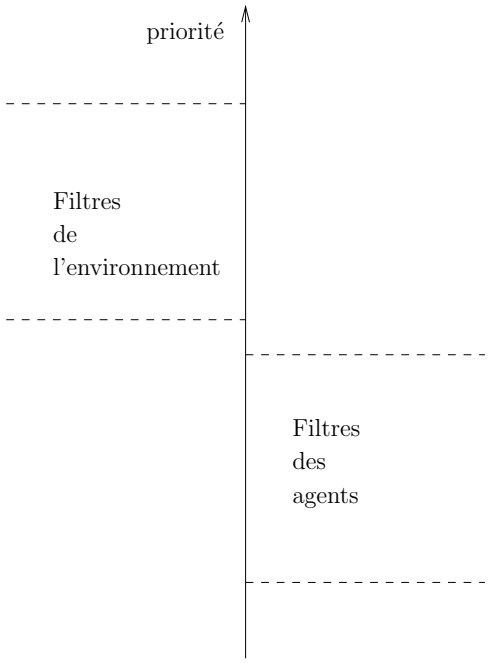


FIGURE 5.1 – Politique de priorité des filtres : la prédominance de l'environnement

2. Dans le cas où il y a un ou plusieurs filtres positifs de l'environnement, les filtres des agents sont pris en compte. L'algorithme effectue la recherche des agents dans $\mathcal{A}$.
3. Dans le cas où il y a un ou plusieurs filtres négatifs de l'environnement, les filtres des agents ne sont pas pris en compte. L'algorithme effectue la recherche des agents dans $\mathcal{A} \setminus \cup_{f \in \mathcal{F}^-} E(f_a)$, autrement dit l'ensemble des agents $\mathcal{A}$ qui n'appartiennent pas à l'extension $E(f_a)$ d'un filtre négatif de l'environnement valide de priorité supérieure.

Cette politique permet donc à l'environnement de réguler totalement le système multi-agents. Ses filtres sont pris en compte dans tous les cas, et en cas de filtres négatifs ceux-ci annulent les filtres des agents.

Du point de vue de l'agent, cela signifie qu'un agent peut recevoir un message malgré un filtre négatif qu'il aurait posé (cas 2), mais aussi qu'il peut ne pas le recevoir malgré un filtre positif (cas 3).

Les filtres de l'environnement peuvent être ajoutés ou retirés dynamiquement, tout comme ceux des agents. C'est pourquoi un agent peut choisir de laisser dans l'environnement un filtre couvert à un instant t par un filtre de priorité supérieure. Un filtre n'est inutile que s'il est couvert par d'autres filtres sur toute la période où il se trouve dans l'environnement.

Exemple cité digitale Dans l'exemple exposé en introduction, le problème venait de la capacité de l'agent à poser des filtres dont la priorité est supérieure à ceux de l'environnement.

Soit IP le domaine de définition des priorités avec $IP = [-128, 127]$, la prédominance de l'environnement sera déterminée par l'application $\mathcal{R}$ suivante :

$$\mathcal{R}:\ \mathcal{F} \mapsto I(IP),\quad \mathcal{R}(f) = \begin{cases} [1,127] & \text{si } initiator = environnement \\ [-128,0] & \text{si } initiator \in \mathcal{A} \end{cases}$$

Dans ce cas, le filtre d'interaction directe f^+_{direct} a nécessairement une priorité positive ($priority(f^+_{direct}) > 0$) puisqu'il appartient à l'environnement, ainsi que le filtre négatif $f^-_{private}$ d'interdiction d'écoute des messages privés ($priority(f^-_{private}) > 0$).

Lorsque l'agent a_x décide de poser son filtre d'interception $f^+_{interception}$, il doit choisir sa priorité :

- Soit il décide de lui adjoindre une priorité positive ($priority(f^+_{interception}) > 0$), auquel cas le filtre est refusé par l'environnement.
- Soit il décide de lui adjoindre une priorité négative ou nulle ($priority(f^+_{interception}) \leq 0$), auquel cas l'algorithme le traite systématiquement après f^+_{direct} et $f^-_{private}$. Or, le filtre négatif $f^-_{private}$ est valide pour tout agent n'étant pas destinataire d'un message privé.

  Preuve : $\forall m \in \mathcal{M},\ E(f^-_{private}) = \{a \in A | id(a) \neq receiver(m)\}$
  $\Rightarrow a_x \notin E(f^-_{private}) \Leftrightarrow id(a_x) = receiver(m)$

  Lorsqu'un filtre négatif est valide, l'algorithme ne traite pas les filtres de plus faible priorité. En conséquence, l'algorithme ne traite le filtre $f^+_{interception}$ que dans les cas où $f^-_{private}$ n'est pas valide, c'est à dire quand a_x est récepteur désigné des messages. Puisque f^+_{direct} est également valide dans ces cas, le filtre $f^+_{interception}$ est inutile.

Grâce à cette politique de priorité, les messages privés ne peuvent être interceptés par aucun agent, puisque leurs filtres auront une priorité inférieure à ceux de l'environnement.

5.3.2.b Environnement facilitateur

La seconde politique est de permettre aux filtres des agents de l'emporter sur ceux de l'environnement.

Politique 2 (Environnement facilitateur) $\mathcal{R}:\ \mathcal{F} \mapsto I(IP)$

$$\mathcal{R}(f) = \left\{ \begin{array}{ll} [min_e, max_e] & si\ initiator = environnement \\ [min_a, max_a] & si\ initiator \in \mathcal{A} \end{array} \right| \ avec\ max_a > max_e$$

Ainsi, les filtres des agents peuvent avoir une priorité supérieure à ceux de l'environnement (figure 5.2). De la même façon que précédemment, cela signifie qu'un filtre (positif ou négatif) d'un agent peut annuler, pour un couple message/agent l'effet des filtres de l'environnement.

Nous parlons alors d'environnement facilitateur, car les règles de l'environnement peuvent être enfreintes au moyen de filtres ayant une priorité adéquate. Cependant, l'existence même de règles de l'environnement indique la présence normative de règles de transmission des messages, qui représentent le comportement souhaité du système multi-agents. Ces règles par défaut peuvent ensuite être adaptées suivant les besoins des agents, mais elle sont là pour faciliter les interactions du système.

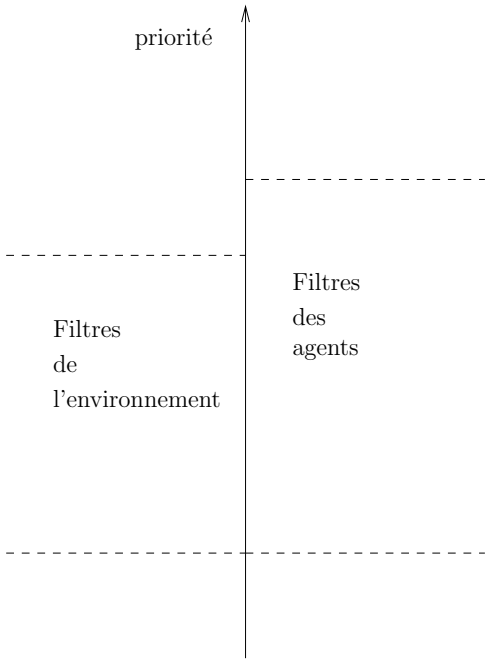


FIGURE 5.2 – Politique de priorité des filtres : l'environnement facilitateur

Exemple cité digitale Soit IP le domaine de définition des priorités avec $IP = [-128, 127]$, l'environnement facilitateur sera déterminé par l'application :

$$\mathcal{R} : \mathcal{F} \mapsto I(IP), \quad \mathcal{R}(f) = \begin{cases} [-128, 0] & \text{si } initiator = environnement \\ [-128, 127] & \text{si } initiator \in \mathcal{A} \end{cases}$$

Le premier exemple pour faciliter les communications est la mise en place de communications de groupe utilisées pour contacter les agents *organisation* en fonction de leur type. La condition sur les messages sera une propriété $receiver - type$, qui sera appariée à la propriété *type* des agents :
$f^{+}_{type}(a, m, C) = \langle [type(a) = ?x], [receiver - type(m) = ?x], \emptyset, \text{“}type\text{”}, 0, environnement \rangle$

Avec ce filtre, tous les agents *organisation* recevront les messages à destination de leur type d'organisation. C'est un comportement par défaut, que les agents peuvent décider de refuser de suivre.

Ainsi, un agent *organisation* a_x ne souhaitant pas recevoir de message lorsqu'il est occupé ajoutera le filtre :
$f^{-}_{indispo}(a, m, C) = \langle [id(a) = id(a_x)] \wedge [dispo(a) = false], \emptyset, \emptyset, \text{“}indispo\text{”}, priority, a_x \rangle$

L'agent a_x a alors deux choix :

– Soit il décide de lui adjoindre une priorité positive ($priority(f^{-}_{indispo}) > 0$), auquel cas il ne suit pas le filtre f^{+}_{type} de distribution des messages de groupe de l'environnement. En effet, lorsque sa propriété *dispo* est égale à $false$, il ne recevra jamais de message par les filtres de priorité inférieure :

$$\forall m \in \mathcal{M},\ dispo(a_x) = false \Rightarrow\ E(f^-_{indispo}) = \{a_x\}$$

L'algorithme ne traitera donc aucun des filtres de l'environnement pour cet agent.

– Soit il décide de lui adjoindre une priorité négative ($priority(f^-_{indispo}) < 0$), auquel cas il suit le filtre de l'environnement. Les messages de groupe lui sont donc bien retransmis, mais les autres filtres de priorité inférieure sont bloqués.

Comparaison avec la politique de prédominance de l'environnement. La différence fondamentale entre cette politique et la politique de prédominance de l'environnement est que dans le premier cas, l'agent reste maître de ses réceptions, alors que dans le second, l'autonomie de l'agent est restreinte par les règles de l'environnement.

Pour illustrer cette différence, si on applique une politique de prédominance de l'environnement dans l'exemple, on obtient alors les conséquences suivantes :

– Soit il décide de lui adjoindre une priorité positive ($priority(f^-_{indispo}) > 0$), auquel cas le filtre est refusé par l'environnement.

– Soit il décide de lui adjoindre une priorité négative ($priority(f^-_{indispo}) < 0$), auquel cas la situation est identique à la politique d'environnement facilitateur : les messages de groupe lui sont retransmis, ainsi que tout les messages transmis via les filtres de l'environnement.

5.3.2.c Mise en oeuvre conjointe de politiques

Les politiques de prédominance de l'environnement et d'environnement facilitateur ne sont pas mutuellement exclusives au sein d'un système multi-agents. Il est possible de mettre en oeuvre une conjonction de ces politiques (figure 5.3). Un environnement peut posséder à la fois des règles impératives, qui seront inviolables, et des règles de facilitation, pour supporter les interactions sans les imposer. Dans ce cas, l'application $\mathcal{R}$ a la forme suivante :

$$\mathcal{R} :\ \mathcal{F} \mapsto I(IP),\quad \mathcal{R}(f) = \left\{ \begin{array}{ll} [min_e, max_e] & \text{si } initiator = environnement \\ [min_a, max_a] & \text{si } initiator \in \mathcal{A} \end{array} \right. \Bigg| \text{ avec } max_a > min_e$$

Les priorités des filtres des agents sont inférieures à celles des filtres impératifs de l'environnement (comprises dans l'intervalle $]max_a, max_e]$), mais peuvent être supérieures à celles des filtres facilitateurs (comprises dans l'intervalle $[min_e, max_a[$).

Dans l'exemple cité digitale, les filtres concernant les messages privés sont des filtres impératifs, afin d'assurer aux agents que ces messages ne seront pas écoutés. Par contre, le filtre de communication de groupe est un filtre facilitateur, dont l'effet peut être modifié par les filtres des agents.

Pour terminer cette section sur les politiques liées aux filtres de l'environnement, il est possible de placer sur un axe les politiques en fonction des entités qu'elles avantagent :

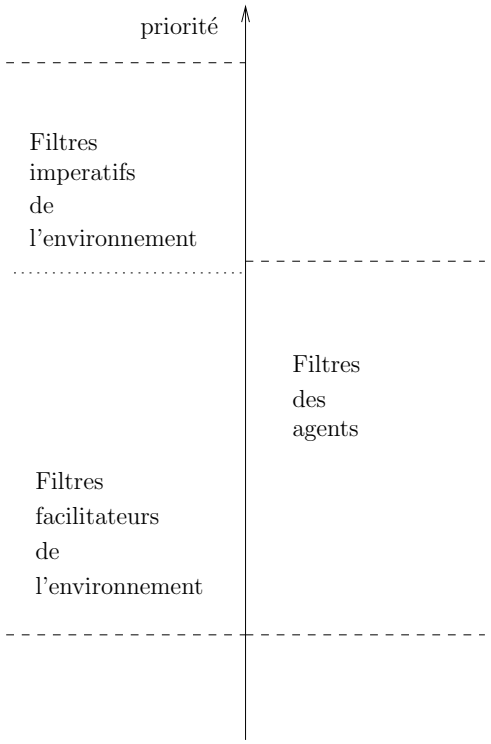
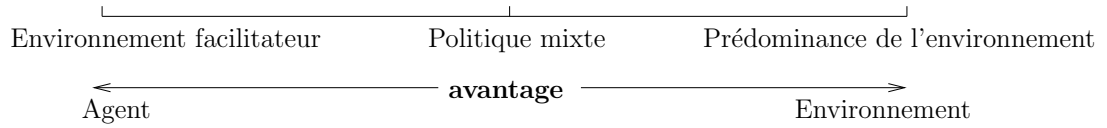


FIGURE 5.3 – Politique de priorité des filtres : exemple d'utilisation jointe de politiques



La politique mixte est équilibrée, puisqu'elle permet de mettre en oeuvre à la fois des règles inviolables de l'environnement et des services. L'environnement facilitateur favorise les agents puisque ceux-ci peuvent dans tous les cas ignorer les filtres de l'environnement. A l'inverse, la prédominance de l'environnement défavorise les agents dans le sens où quelques soient leurs filtres, ils ne peuvent en aucun cas moduler les règles de l'environnement en fonction de leurs besoins.

5.3.3 Politiques relatives aux filtres des agents

Nous avons vu l'impact des politiques de priorités entre filtres de l'environnement et filtres des agents. Nous étudions maintenant le cas de conflit entre les filtres des agents. Ces conflits peuvent émerger pour trois raisons : une mauvaise conception des filtres, des besoins et intérêts divergents, ou un comportement malveillant. Une mauvaise conception des filtres peut entraîner l'inclusion involontaire d'agents dans l'extension de f_a. Dans ce cas, le comportement du filtre s'applique à ces agents non-voulus, éventuellement en contradiction avec leurs objectifs. Des besoins divergents peuvent amener un agent à ajouter des filtres pour d'autres agents, alors que

ceux-ci ne souhaitent pas être concernés par ces filtres. Un comportement malveillant est l'ajout volontaire d'un filtre dans l'objectif de nuire aux autres agents.

Un seul filtre négatif d'un agent, dont la description couvrirait tous les agents et tous les messages, peut suffire pour désactiver tous les autres filtres, dans le sens où ceux-ci ne seraient jamais traités par l'algorithme. Par miroir, un filtre positif ayant les mêmes caractéristiques qu'une diffusion généralisée surchargerait les agents en coût de traitement des messages.

Dans l'exemple de la cité digitale, soient deux agents a_1 et a_2. a_2 décide d'écouter les évènements de type concert si ceux-ci ont lieu dans un restaurant :
$f^+_{concert}(a, m, \{c\}) = \langle [id(a) = \text{“}a2\text{”}], [type(m) = \text{“}concert\text{”}] \wedge [lieu(m) = ?x], [nom(c) = ?x] \wedge [type(c) = \text{“}restaurant\text{”}], \text{“}concert\text{”}, 0, a_2 \rangle$

a_1 est un agent malveillant qui décide d'ajouter un filtre de façon à ce que a_2 ne reçoive plus aucun message. Un filtre parvenant à cet objectif est le suivant :
$f^-_{sourd}(a, m, C) = \langle [id(a) = \text{“}a2\text{”}], \emptyset, \emptyset, \text{“}sourd\text{”}, 0, a_1 \rangle$

Avec les politiques 1 et/ou 2, tous les agents peuvent déposer des filtres de même priorité. Les filtres pris en compte dépendent de la priorité entre filtres positifs et négatifs. Dans le cas où les filtres négatifs sont privilégiés, l'agent a_2 ne percevra aucun message, le filtre de a_1 étant traité avant ses propres filtres positifs, par exemple $f^+_{concert}$. Dans le cas où les filtres positifs sont privilégiés, l'agent a_2 ne sera pas vulnérable aux filtres négatifs des autres agents. Par contre, si un filtre positif possédant la même syntaxe que f^-_{sourd} est ajouté, alors a_2 recevra tous les messages, y compris ceux qu'ils ne souhaitent pas recevoir.

Nous proposons une solution à ce risque, qui est patent dans le cadre d'un système multi-agents dont tous les agents ne sont pas homogènes et contrôlés par le concepteur. Nous différencions les filtres posés par les agents, et en particulier introduisons un type de filtre particulier, les filtres *personnels*. Les filtres personnels sont les filtres dont l'initiateur est le seul récepteur potentiel.

Définition 14 (Filtre personnel) *f est un filtre personnel de l'agent a_x si et seulement si $initiator_f = a_x$ et $E(f_a) = \{a_x\}$*

Autrement dit, quand un agent pose un filtre pour lui même, c'est un filtre personnel. Les filtres personnels déclenchent ou bloquent la réception pour leur seul initiateur. Ceci sera déterminé par exemple grâce à son identifiant : si f_a est $[id(a) = a_x]$, alors $E(f_a)$ contient seulement a_x.

Nous notons $\mathcal{F}_{AP} \subset \mathcal{F}_A$ l'ensemble des filtres personnels. De cette façon, nous distinguons les filtres qui n'impliquent que leur propriétaire de ceux qui impliquent –exclusivement ou non – d'autres agents. Il reste alors à déterminer les politiques de l'environnement relatives aux filtres des agents. Celles-ci sont indépendantes des politiques relatives aux filtres de l'environnement.

5.3.3.a Prédominance personnelle

La première politique est celle de la prédominance personnelle, qui consiste à favoriser les filtres personnels des agents.

Politique 3 (Prédominance personnelle) $\mathcal{R} : \mathcal{F} \mapsto I(IP)$

$$\mathcal{R}(f) = \left\{ \begin{array}{ll} [min_e, max_e] & si\ initiator = environnement \\ [min_{ap}, max_{ap}] & si\ initiator \in \mathcal{A}\ et\ f \in \mathcal{F}_{AP} \\ [min_a, max_a] & si\ initiator \in \mathcal{A}\ et\ f \notin \mathcal{F}_{AP} \end{array} \right| \ avec\ max_{ap} > max_a$$

Les filtres personnels des agents ont une priorité plus haute que les autres filtres des agents (figure 5.4). L'avantage de cette politique est d'assurer aux agents la prévalence de leurs filtres, et d'ainsi limiter pour eux les effets de filtres mal conçus ou malveillants posés par d'autres agents. Au niveau du système multi-agents, cette politique permet d'empêcher un blocage du comportement normal du modèle EASI.

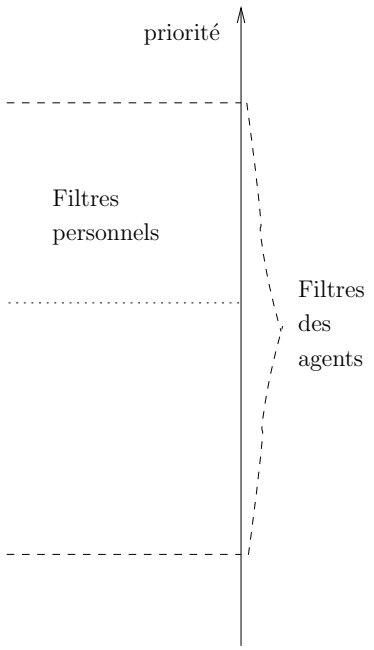


FIGURE 5.4 – Politique de priorité des filtres : la prédominance personnelle

Exemple cité digitale Nous reprenons l'exemple avec $f^+_{concert}$ et f^-_{sourd}. Soit IP le domaine de définition des priorités avec $IP = [-128, 127]$, la prédominance personnelle est déterminée par l'application :

$$\mathcal{R} : \mathcal{F} \mapsto I(IP),\ \ \mathcal{R}(f) = \left\{ \begin{array}{ll} [-128, 127] & \text{si } initiator = environnement \\ [-128, 127] & \text{si } initiator \in \mathcal{A} \text{ et } f \in \mathcal{F}_{AP} \\ [-128, 0] & \text{si } initiator \in \mathcal{A} \text{ et } f \notin \mathcal{F}_{AP} \end{array} \right.$$

L'agent a_2 déposant son filtre $f^+_{concert}$ peut lui donner une priorité dédiée aux filtres personnels (dans $[1, 127]$), auquel cas $priority(f^+_{concert}) > priority(f^-_{sourd})$. Pour chaque nouveau message,

$f^{+}_{concert}$ est traité avant f^{-}_{sourd}. Celui-ci n'influence donc pas la réception des messages pour l'agent a_2.

Grâce aux filtres personnels, les agents maîtrisent leurs communications. Ainsi, les autres agents ne peuvent pas les empêcher de recevoir les messages qui les intéressent, ni les "noyer" sous un flot de messages indésirables.

5.3.3.b Politique différenciée

La politique différenciée consiste à traiter de façon différente les agents, en fonction de critères tels que leur catégorie (figure 5.5). Nous donnons deux politiques de différenciation des priorités.

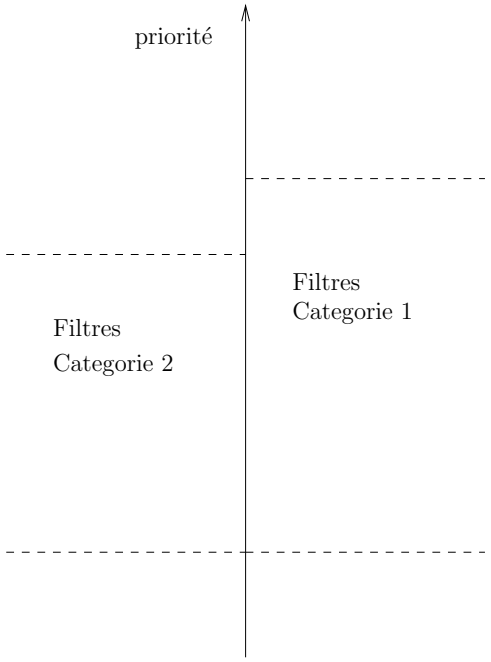


FIGURE 5.5 – Politique de priorité des filtres : la politique différenciée

Politique 4 (Différenciation catégorielle) *Soit* $\{P_{cat1}, P_{cat2}, ..., P_{cati}\}$ *un ensemble de Pdescriptions de catégories d'agent.*

$\mathcal{R} : \mathcal{F} \mapsto I(IP)$

$$\mathcal{R}(f) = \left\{ \begin{array}{ll} [min_e, max_e] & si\ initiator = environnement \\ [min_{cat1}, max_{cat1}] & si\ initiator = a\ et\ P_a \supset P_{cat1} \\ [min_{cat2}, max_{cat2}] & si\ initiator = a\ et\ P_a \supset P_{cat2} \\ ... & ... \\ [min_{cati}, max_{cati}] & si\ initiator = a\ et\ P_a \supset P_{cati} \end{array} \right| \begin{array}{r} avec\ max_{cat1} > max_{cat2} > \\ ... > max_{cati} \end{array}$$

Cette politique permet de différencier les priorités autorisées en fonction des Pdescriptions P_{cati} des agents, et donc des catégories auxquelles ils appartiennent. En l'espèce, les agents

de la catégorie $Cat(P_{cat1})$ peuvent ajouter des filtres de priorité supérieure à ceux des agents appartenant à la catégorie $Cat(P_{cat2})$, et ainsi de suite jusqu'au agents appartenant à la catégorie $Cat(P_{cati})$.

Les raisons d'une différenciation de ce type peuvent être structurelles (les agents possédant certaines propriétés sont connus *a priori*), ou dynamiques (une catégorie doit posséder une plus grande liberté interactionnelle pendant une durée δt pour mener à bien ses tâches).

La deuxième politique est fondamentalement dynamique et s'appuie sur les mécanismes de confiance.

Politique 5 (Différenciation individuelle) *Soit une application $c(a)$ donnant la confiance accordée à l'agent a, dont l'ensemble d'arrivée est ordonné.*

$\mathcal{R} : \mathcal{F} \mapsto I(IP)$

$$\mathcal{R}(f) = \left\{ \begin{array}{ll} [min_e, max_e] & si\ initiator = environnement \\ [min_{a1}, max_{a1}] & si\ initiator = a_1 \\ [min_{a2}, max_{a2}] & si\ initiator = a_2 \end{array} \right. \left| \begin{array}{l} avec\ c(a_1) > c(a_2) \Leftrightarrow \\ \qquad max_{a1} > max_{a2} \end{array} \right.$$

Sans présumer du mécanisme de calcul de confiance du système multi-agents, nous supposons uniquement que l'ensemble d'arrivée de la fonction donnant sa mesure est ordonné. L'application $\mathcal{R}$ doit respecter l'équivalence $c(a_1) > c(a_2) \Leftrightarrow max_{a1} > max_{a2}$, en d'autres termes si un agent a_1 a un niveau de confiance supérieur à celui de a_2, alors a_1 peut poser des filtres de priorité supérieure à ceux de a_2.

A l'aide de cette application, nous permettons aux agents de gagner des capacités interactionnelles relatives en fonction de leur niveau de confiance. De cette façon, un agent qui a une plus forte confiance est assuré de voir ses filtres primer sur ceux des agents de plus faible confiance, ce qui permet :

- de récompenser les agents ayant un comportement positif pour le système multi-agents.
- de réduire les effets des filtres des agents ayant un comportement négatif.

Exemple cité digitale Un exemple de différenciation catégorielle est de permettre aux agents *organisation* de déposer des filtres de priorité supérieure à ceux des agents *utilisateur*.

Au niveau de l'environnement, la différence se fait sur les Pdescriptions : soit $P_{organisation} = \{id, nom, type\}$, l'ensemble des agents de la catégorie *organisation* est $Cat(P_{organisation})$, et soit $P_{utilisateur} = \{id, nom, dispo, age, sexe, lieu\}$, l'ensemble des agents *utilisateur* est alors $Cat(P_{utilisateur})$.

La politique de différenciation catégorielle est donc ici :

$$\mathcal{R} : \mathcal{F} \mapsto I(IP), \quad \mathcal{R}(f) = \left\{ \begin{array}{ll} [-128, 127] & \text{si } initiator = environnement \\ [-128, 127] & \text{si } initiator = a \text{ et } P_a \supset P_{organisation} \\ [-128, 0] & \text{si } initiator = a \text{ et } P_a \supset P_{utilisateur} \end{array} \right.$$

La situation des agents *utilisateur* est, vis à vis des filtres des agents *organisateur*, identique à celle vis à vis d'une prédominance de l'environnement : quelque soient leurs filtres, les filtres des agents *organisateur* priment sur ceux des agents *utilisateur*. Cette politique permet au concepteur du système multi-agents d'accorder *a priori* une plus grande autonomie à certains agents.

5.3.4 Un environnement "naturel"

En opérant une combinaison des politiques 1, 2, et 3, nous aboutissons à un ordre classique des règles (figure 5.6), telles que nous les avons évaluées dans la section 1.4 sur la conscience des autres.

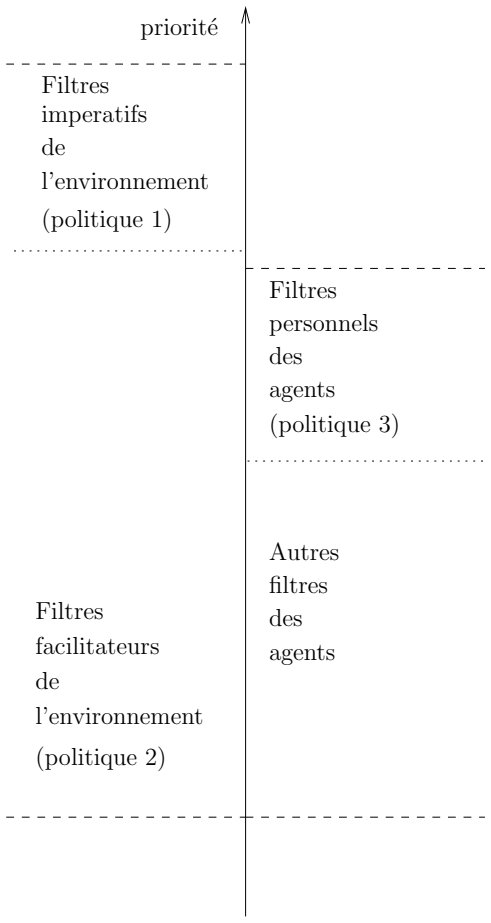


FIGURE 5.6 – Ordonnancement des filtres pour un environnement classique.

– La politique 1 implique que tous les agents doivent se soumettre aux règles inviolables de l'environnement. Par exemple, ce sont les stimuli en provenance de l'environnement que les agents ne peuvent choisir, de même que les lois physiques les empêchant de recevoir des messages.
– Sous ces règles impératives, la politique 3 permet aux agents de définir leurs propres

réceptions. Ce sont les décisions d'actions qui sont au coeur du concept d'*attention*.

- Bien que les agents puissent les forcer, la politique 2 permet à d'autres règles de l'environnement d'exister pour un déroulement classique des interactions. Ce sont les lois de l'environnement dont le résultat peut être altéré par les décisions des agents.
- Enfin, les agents peuvent recevoir d'autres sollicitations par le biais des filtres non personnels. Ce sont les focus ou initiatives que les agents choisissent.

Nous illustrons avec la figure 5.7 l'exemple des conversations humaines, qui sont situées. Soit un environnement avec une topologie à deux dimensions, et soit $dist(a_x, a_y)$ une fonction de calcul de distance dans l'espace. Deux règles non enfreignables de l'environnement peuvent être représentées par f^+_{proxim} et f^-_{eloign}.

- Deux agents situés à une distance inférieure à d l'un de l'autre entendent obligatoirement les messages émis par l'autre. Le filtre $f^+_{proxim} \in \mathcal{F}^+$ représentant cette loi est :
  $\langle [dist(a, a_2) < d], [sender(m) = ?x], [id(a_2) = ?x], \text{“}proxim\text{”}, 127, environnement \rangle$
- Deux agents situés à une distance supérieure à D l'un de l'autre ne peuvent pas entendre les messages émis par l'autre. Le filtre $f^-_{eloign} \in \mathcal{F}^-$ représentant cette loi est :
  $\langle [dist(a, a_2) > D], [sender(m) = ?x], [id(a_2) = ?x], \text{“}eloign\text{”}, 127, environnement \rangle$

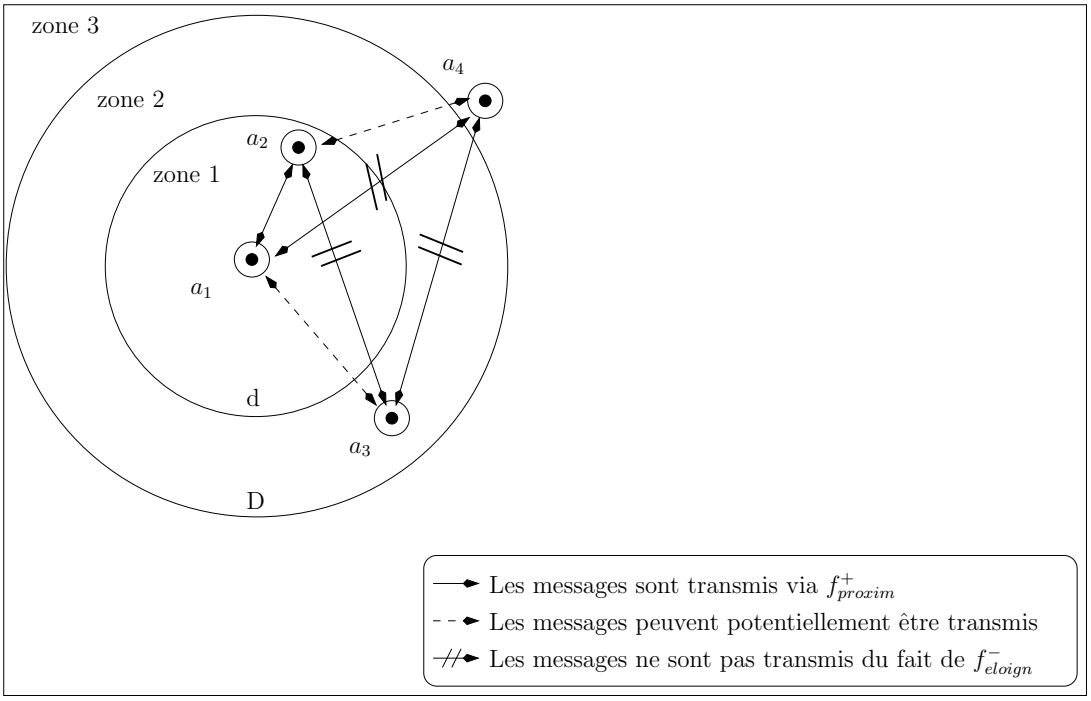


FIGURE 5.7 – Politique de priorité des filtres : exemple d'environnement situé.

Dans l'intervalle compris entre d et D, ce sont les filtres des agents qui déterminent ou non la réception. Dans la figure, a_1 percevra tous les messages de a_2 (et réciproquement), il ne percevra aucun message de a_4, et il recevra éventuellement ceux de a_3 si des filtres le permettent. Notons que la gestion des filtres et des messages est dynamique, ainsi si $dist(a_1, a_4)$ devient inférieure à

D, les nouveaux messages émis par ces deux agents peuvent être écoutés par l'autre.

5.4 Conclusion

Dans le chapitre précédent, nous avons montré comment le modèle EASI permet de mettre en oeuvre les communications multi-parties. Dans ce chapitre, nous avons tout d'abord développé l'idée de régulation des communications en séparant les filtres posés par les agents de ceux attribués à l'environnement. Les premiers représentent les besoins des agents à titre individuel, tandis que les seconds sont des règles communes.

Les filtres du modèle EASI déclenchent la réception des messages. Dans le modèle EARI, Environnement Actif comme Régulateur de l'Interaction, un second type de filtre nommé *filtre négatif* est introduit pour les limiter. Grâce aux filtres négatifs, les agents peuvent contrôler *a priori* les messages qu'ils vont recevoir. Au niveau multi-agents, il s'agit de mettre en place des règles de distribution des messages, et notamment d'assurer que les messages ne seront pas interceptés par des agents non désirés.

L'articulation entre filtres positifs et négatifs est gérée par leurs priorités respectives. La vérification des priorités par l'environnement permet de mettre en oeuvre différentes politiques de prévalence entre les filtres en fonction de leur initiateur.

De cette façon, il est possible de modéliser des lois de l'environnement qui ne peuvent pas être enfreintes par les agents, et des règles de transmissions qui peuvent être outrepassées. Dans la même logique, les filtres déposés pour eux-mêmes par les agents auront une priorité supérieure à ceux des autres agents. Enfin, les priorités pourront être différenciées en fonction des groupes d'agents et/ou à titre individuel.

Le modèle EASI donne le cadre de la modélisation de l'environnement et des filtres, et l'extension aux filtres négatifs complète ce modèle par la prise en compte du contrôle des communications. Dans le chapitre suivant, nous voyons comment mettre en oeuvre l'ensemble de notre modèle.

Chapitre 6

Déploiement de l'environnement d'interaction

Sommaire

Dans les chapitres 3 à 5, nous avons introduit le modèle d'environnement d'interaction EASI et son extension pour le contrôle des communications EARI. Nous nous intéressons dans ce chapitre à la mise en oeuvre du modèle pour la conception des systèmes multi-agents.

En première partie, nous introduisons une architecture abstraite d'environnement pour le support du modèle EASI, et décrivons les différents modules qui la composent. Le modèle EASI est un environnement *de communication*, nous nous intéressons donc également à la façon dont l'architecture peut s'intégrer à un environnement global proposant d'autres services (comme nous les avons présentés dans la section 2.2.5). Nous proposons une architecture d'agent minimale pour l'utilisation de l'environnement d'interaction.

En seconde et troisième partie, nous présentons et discutons deux implémentations du modèle. La première implémentation est un prototype d'environnement distribué, en fonctionnement client/serveur, doté d'une bibliothèque pour les agents et d'un serveur de communications.

La seconde implémentation est une intégration de l'environnement à une plate-forme existante, MadKit. Nous introduisons une extension des communications du modèle Agent-Groupe-Rôle. Les filtres pouvant être vus comme des règles, nous proposons de mettre en oeuvre l'environnement par un système expert.

6.1 Architecture abstraite

Une architecture abstraite est une description d'un système logiciel à un haut niveau d'abstraction. Elle repose sur un ensemble de composants, ou modules, qui représentent les unités de calcul et de stockage, et sur un ensemble de connecteurs, qui représentent les interactions entre les composants. L'architecture est indépendante de toute plate-forme d'implémentation, il s'agit d'une description fonctionnelle du système. Lors du raffinement nécessaire à l'implémentation, les composants peuvent être conservés, fusionnés, ou divisés en sous-composants.

6.1.1 Vue d'ensemble du système multi-agents

L'environnement est un *middleware*, qui fait partie de l'infrastructure des agents (figure 6.1). Plus précisément, l'environnement est un médiateur entre les agents et le contexte de déploiement, formé de leur(s) hôte(s) et des logiciels de bas niveau comme le système d'exploitation. L'environnement de communication que nous proposons peut être tout l'environnement du système multi-agents, ou une partie d'un environnement proposant d'autres services comme la gestion du cycle de vie des agents.

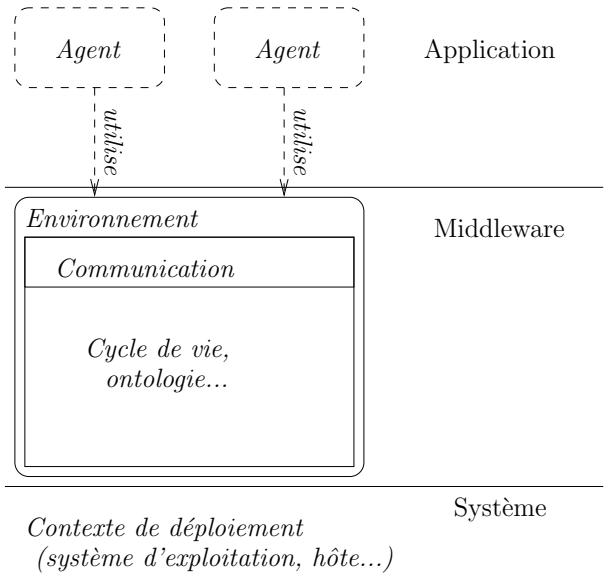


FIGURE 6.1 – Schéma général du déploiement du modèle EASI

L'architecture de l'infrastructure de support d'EASI est donnée en figure 6.2. Nous décrivons

la partie communication de l'environnement, ce dernier pouvant contenir d'autres éléments (section 6.1.3).

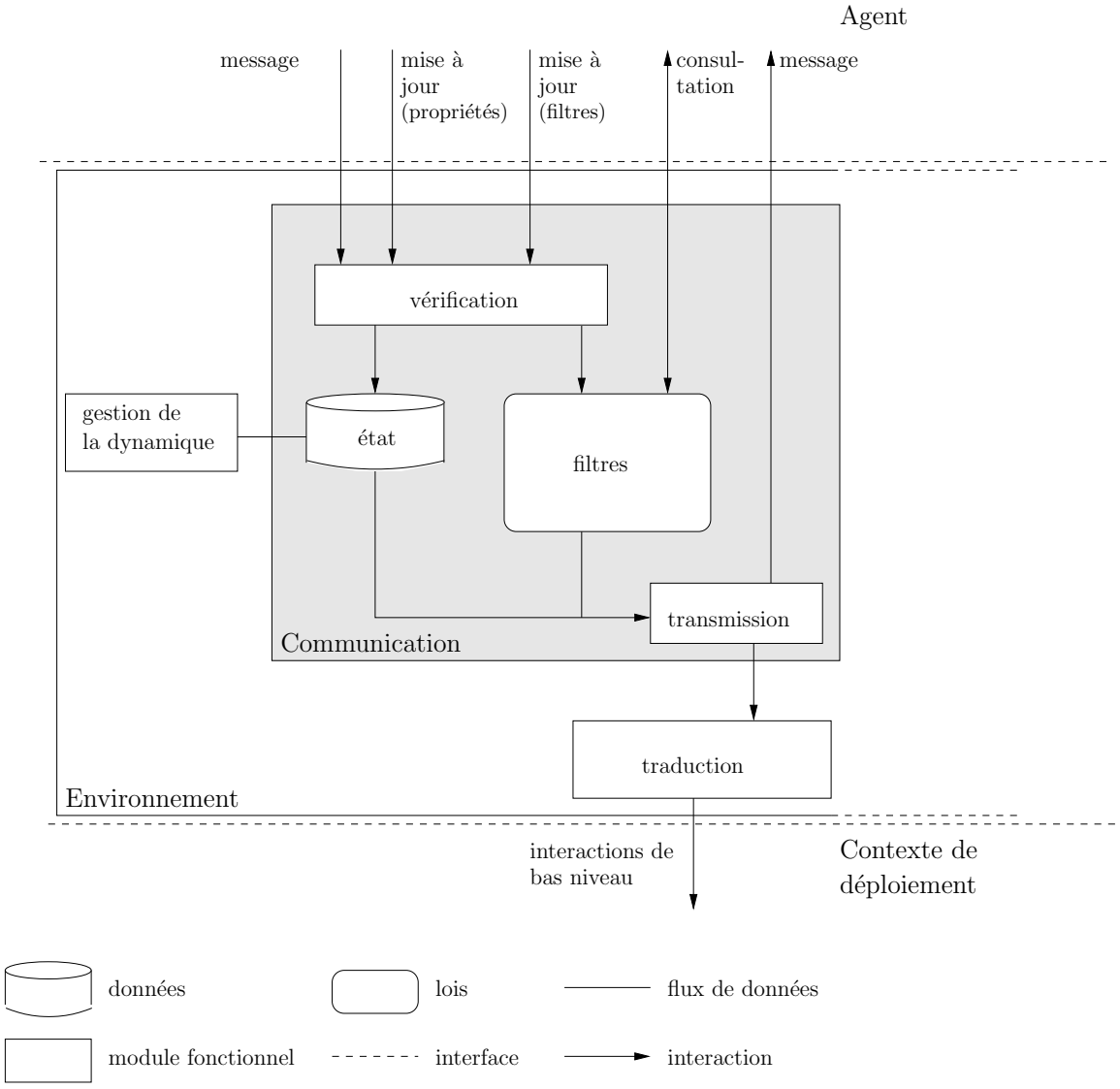


FIGURE 6.2 – Architecture de l'environnement

L'environnement de communication contient quatre modules : l'état, les filtres, la vérification et la transmission. Nous étudions dans la section prochaine chacun de ces composants, et détaillons les interactions entre les agents et les composants de l'environnement.

6.1.2 Description des composants de l'environnement

Module état Le premier module stocke l'état actuel de l'environnement. Il contient deux types de données : les descriptions de toutes les entités de l'environnement ($\mathcal{D}$), et certaines entités elles-mêmes comme les messages persistants. L'état peut être modifié soit par les agents,

soit par un module interne à l'environnement qui gère la dynamique du système. Le module *état* contient donc l'ensemble des descriptions des entités de Ω.

Module filtres Ce module sert à stocker et manipuler les filtres, positifs et négatifs. Il contient donc l'ensemble de filtres $\mathcal{F}$. D'après le modèle, ceux-ci doivennt être triés par ordre de priorité.

L'agent interagit avec le module *filtres* pour consulter les filtres qui ont déjà été déposés dans l'environnement.

Module vérification Lorsque les agents ajoutent ou modifient un filtre, il convient d'effectuer un certain nombre de contrôles pour vérifier la validité de l'opération. Ainsi sont vérifiées la syntaxe du filtre et sa compatibilité avec le format exigé, ainsi que sa priorité en fonction de l'intervalle de priorité autorisé IP et des politiques de priorités en vigueur $\mathcal{R}$. Ce module contrôle aussi l'accès aux descriptions : seul le propriétaire d'une description a le droit de la modifier ou de la retirer.

L'agent interagit avec le module *vérification* pour manipuler les filtres dont il est l'initiateur et mettre à jour les descriptions dans l'environnement. Plus précisément, il peut procéder à l'ajout d'un nouveau filtre, à la modification d'un filtre existant ou au retrait d'un filtre.

L'agent interagit également avec le module *vérification* pour mettre à jour les descriptions dans l'environnement. L'agent, en s'inscrivant auprès de l'environnement, transmet sa propre description à l'environnement. Il peut également ajouter à tout moment des descriptions d'entités. Une fois les descriptions dans l'environnement, l'agent peut modifier les valeurs de ses propriétés, ou celles de toutes les descriptions d'entités qu'il a lui-même ajoutées. Enfin, il peut retirer les descriptions qu'il a ajoutées, et se désinscrire de l'environnement.

Enfin, l'agent interagit directement avec le module *vérification* pour ajouter des entités dans l'environnement. Dans ce cas, c'est l'entité elle-même et non uniquement sa description qui est transmise. Les messages, par exemple, sont transmis de façon à être ensuite distribués aux récepteurs.

Module transmission Le module de transmission sélectionne les récepteurs des données en fonction des filtres et de l'état de l'environnement. Il contient l'algorithme d'appariement entre filtres et descriptions, qui utilise les modules *filtres* et *état*. Une fois les récepteurs choisis, le module effectue la transmission soit directement aux agents concernés, soit par le biais de protocoles de plus bas niveaux via un module de traduction.

Le module de *transmission* de l'environnement interagit avec l'agent pour distribuer les messages à cet agent. Du point de vue de l'agent, c'est la réception d'un message et d'un ensemble de descriptions du contexte (comme nous l'avons vu dans la section 4.4).

6.1.3 Intégration à l'architecture de référence de l'environnement

L'environnement de communication peut être une infrastructure suffisante pour le support du système multi-agents, dans le cadre d'agents purement communiquants. Cependant, la notion

d'environnement au sens de [Weyns et al., 2007] vue en section 2.2.5 peut encapsuler d'autres tâches, nous voyons donc ici comment notre architecture s'intègre à celle de Weyns *et al.* Cette architecture (figure 6.3) est l'architecture de référence de l'environnement issue des travaux de la communauté *Environment for Multi-Agent Systems*[22].

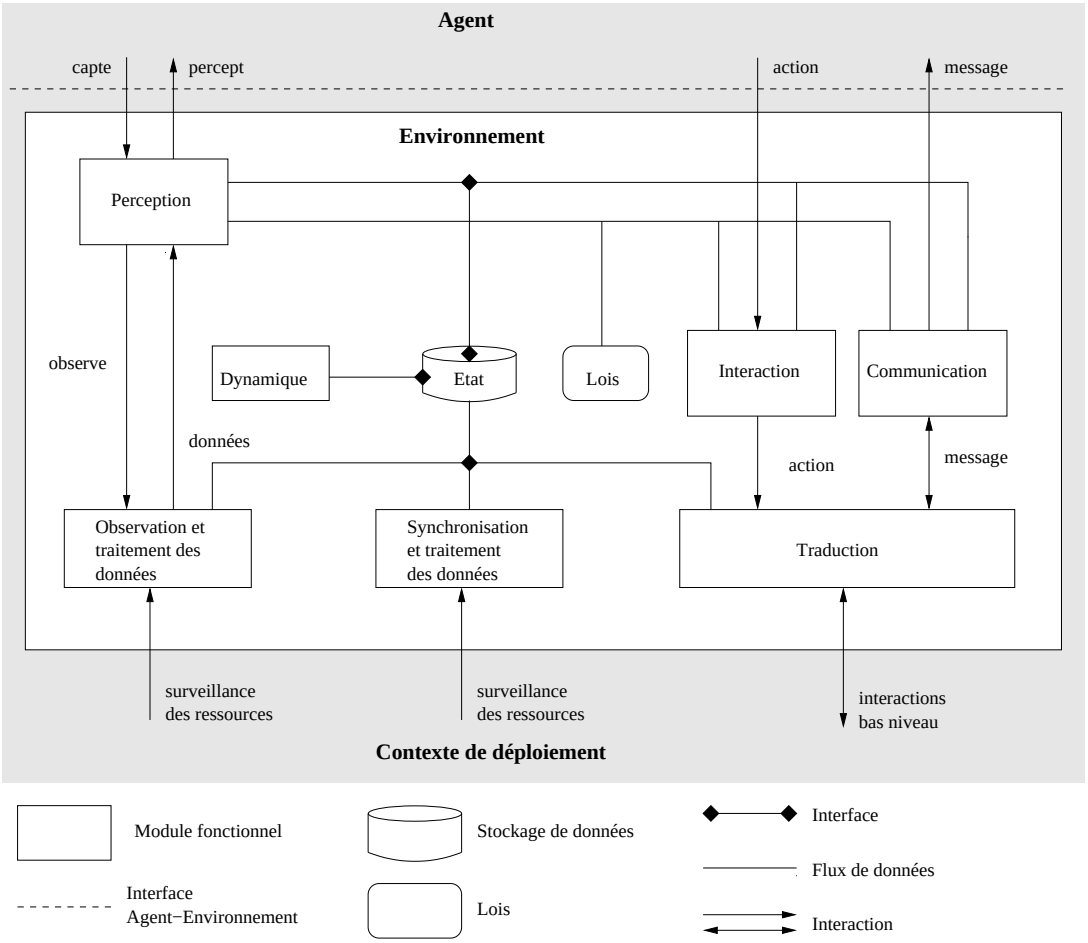


FIGURE 6.3 – Architecture d'environnement [Weyns et al., 2007]

Le module *état* est équivalent dans les deux architectures. Pour les autres modules, le découpage fonctionnel est effectué différemment.

Le module filtres correspond au module *lois*, dont l'objectif est de gérer des lois pour trois fonctions : la perception, les actions et les communications. Les filtres de l'environnement sont, dans notre modèle, les lois de transmission de message. Une des originalités de notre modèle est la gestion des filtres des agents au sein de l'environnement, cette fonctionnalité n'a donc pas d'équivalent.

Notre module de vérification n'est pas explicitement exprimé dans l'architecture de référence.

22. *Environment for Multi-Agent Systems* (E4MAS) a été un groupe de travail actif de l'action européenne AgentLink III, ainsi qu'un *workshop* de la conférence AAMAS de 2004 à 2006

La vérification syntaxique est gérée au niveau de chaque module interagissant avec l'environnement (*perception*, *interaction*, *communication*). La vérification des politiques de priorités correspond au module *lois*.

Le module de transmission correspond au module *communication*, pour la gestion du choix des récepteurs en fonction de l'état et des lois de communication.

6.1.4 Architecture minimale des agents communiquants

Pour pouvoir utiliser l'environnement d'interaction, l'agent doit être capable de le manipuler. Nous présentons en figure 6.4 une architecture d'agent "minimale", dans le sens où seuls les modules fonctionnels indispensables à la gestion des communications sont indiqués. Les agents que nous avons implémentés dans le cadre de cette thèse suivent cette architecture (sections 7.1.2 et 7.2.1.a).

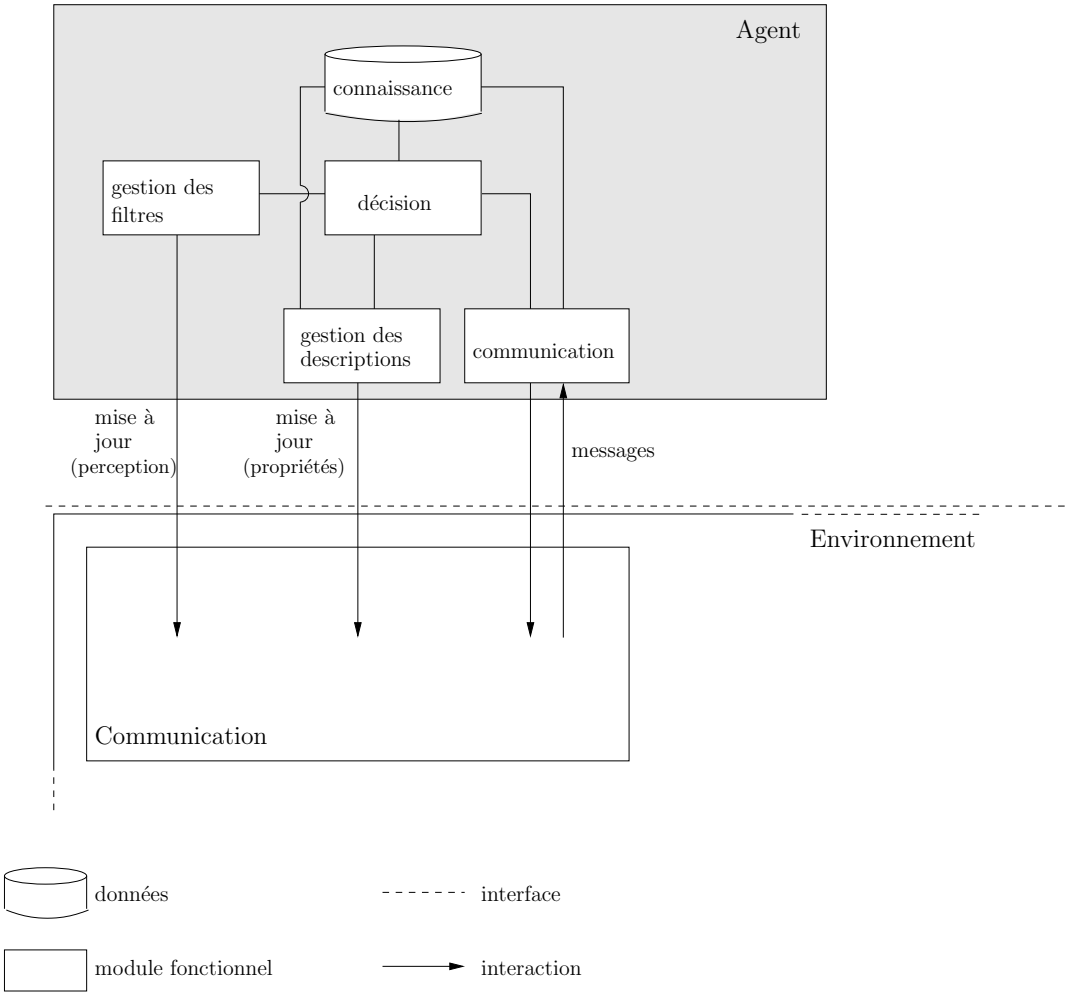


FIGURE 6.4 – Architecture minimale des agents

L'agent est composé de cinq composants : le module de communication, les connaissances

courantes de l'agent, le module de décision, le module de gestion des filtres et le module de gestion des descriptions.

Module de communication Le module de communication fournit les outils nécessaires à l'émission des messages dans l'environnement et à leur réception. Ce module est déclenché par le module de décision, génère les messages selon la syntaxe en vigueur dans le système multi-agents (par exemple FIPA-ACL) et se charge de leur transmission à l'environnement. Il traite les messages entrants, et transmet au module connaissance les modifications de l'état du système multi-agents.

Module connaissance Le module connaissance contient toutes les connaissances de l'agent sur l'état courant du SMA. Les connaissances peuvent être modifiées par le module de communication. Elles sont utilisées par le module de décision et par le module de gestion des descriptions.

Module de décision Le module décision est responsable de la sélection d'action. En fonction de l'état des connaissances de l'agent, il déclenche une action de gestion des filtres de l'agent, de gestion des descriptions ou un envoi de message.

Module de gestion des filtres Le module de gestion des filtres a deux fonctions : la première est de créer les filtres de l'agent, la seconde est de gérer les filtres dans l'environnement.

La génération des filtres de l'agent peut reposer sur une bibliothèque de filtres inclus dans l'agent à la conception, ou sur un processus de génération automatique de filtres en fonction des descriptions des entités du système multi-agents.

La gestion des filtres dans l'environnement contient l'ajout, la modification et le retrait des filtres de l'agent dans l'environnement, ainsi que la consultation des filtres.

Module de gestion des descriptions Le module de gestion des descriptions est responsable de la mise à jour de la description de l'agent dans l'environnement, ainsi que de l'ajout, la modification et le retrait des descriptions des autres entités dont l'agent a la responsabilité.

6.2 Prototype client/serveur

Nous avons présenté en première partie une architecture abstraite, qui donne une vue des différentes fonctionnalités désirées. Après ce premier pas vers la conception du middleware, nous abordons la mise en oeuvre de cette architecture dans la conception d'un support réel des communications multi-parties.

6.2.1 Architecture du prototype

Nous avons implémenté un support distribué de l'environnement de communication. Dans les chapitres précédents, nous avons vu que les données (filtres, descriptions) sont stockées au

niveau de l'environnement, et que l'algorithme d'appariement doit pouvoir accéder à l'ensemble de ces données. Si l'accès aux données est incomplet, l'algorithme ne peut pas assurer que tous les agents qui devraient recevoir un message le recevront bien. A cette fin, nous proposons pour ce prototype une architecture client/serveur, qui centralise les informations au niveau du serveur. Ceci permet à l'algorithme d'accéder aux données sans délai, *i.e.* sans requête à d'autres hôtes distants. En contrepartie, il existe un risque de congestion au niveau du serveur si le nombre d'opérations qu'il doit effectuer devient élevé.

En section 6.2.1.a, nous donnons une vue d'ensemble d'un système multi-agents utilisant notre prototype. Ensuite nous détaillons les côtés client et serveur respectivement en parties 6.2.1.b et 6.2.1.c .

6.2.1.a Vue d'ensemble

La figure 6.5 montre l'architecture du prototype. Les agents sont répartis sur plusieurs hôtes. Un hôte est une unité computationnelle : ordinateur, assistant personnel, *etc.* La communication est médiée par l'environnement de communication, lequel est composé de deux éléments. Le premier élément est un middleware exécuté sur les machines hôtes. Le second élément est un serveur dédié à la gestion des descriptions et à la recherche des récepteurs des communications.

Le middleware est composé d'une bibliothèque de fonctions relatives aux actions des agents, d'un annuaire local, et d'un module de distribution des messages. L'aspect réseau est transparent pour les agents, ceux-ci n'interagissent qu'avec le middleware local. Lorsqu'un agent interagit avec l'environnement, le middleware met localement à jour les informations, et transmet les informations au serveur. Le serveur est mis à jour, et éventuellement la distribution de messages à de nouveaux récepteurs est déclenchée. Le serveur contacte les hôtes concernés et leur transmet le message. Le middleware transmet alors le message aux agents concernés. Ce prototype a été développé en Java [23].

6.2.1.b Côté client

Le middleware est composé de trois modules :

- *Interaction* : l'agent interagit avec l'environnement via ce module. Les agents ont deux types d'interactions avec l'environnement, l'inscription et la modification. Une interaction provoque éventuellement une modification de l'état local, et elle est transmise au serveur.
- *Etat local* : il s'agit d'un annuaire (pages blanches) des agents utilisant le middleware.
- *Distribution* : lorsqu'il doit transmettre à un ou plusieurs agents un message, le serveur transmet le message au middleware, qui se charge de la distribution locale.

Les agents peuvent s'inscrire et se désinscrire de l'environnement. Lors de l'inscription, un agent transmet sa description et la référence de sa boite aux lettres au middleware.

Le middleware crée un identifiant unique pour l'agent nouvellement inscrit, et stocke les informations dans l'état local. La description, à laquelle est ajoutée l'identifiant, est transmise

23. `http://sun.java.com`

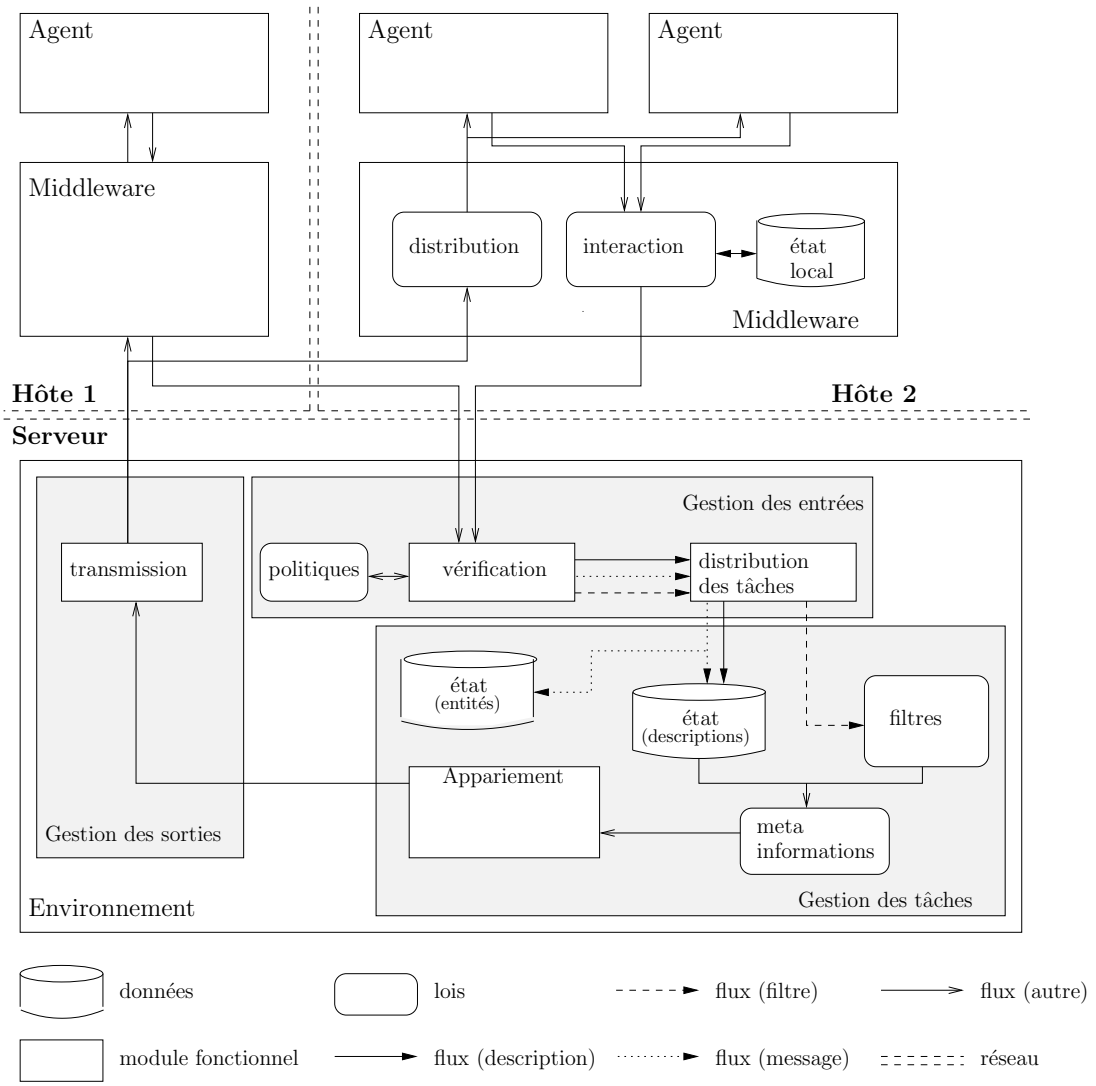


FIGURE 6.5 – Flux d'information dans l'architecture du prototype

au serveur de communication. Ensuite, lorsque l'agent souhaite modifier le comportement de l'environnement, il lui transmet une instruction. Ces instructions (ajout/retrait/modification de filtre ou description) sont écrites dans un langage de balisage.

Symétriquement, les agents peuvent recevoir des messages. Le serveur transmet au middleware le message, ainsi que la liste de récepteurs. Le middleware utilise alors l'état local pour retrouver les récepteurs et ajouter le message dans leurs boites aux lettres.

Chaque agent peut exécuter un middleware, mais il est recommandé de n'exécuter qu'un middleware par hôte. L'objectif est la mutualisation de la bande passante. Pour l'émission de description ou de message, le coût est équivalent. Pour la réception, un middleware par agent entraîne un coût en bande passante entre le serveur et les hôtes de $n \times t$ avec n le nombre de récepteurs et t la taille du message. Au contraire, un middleware par hôte entraîne un coût de $(t + e) \times h$, avec e l'enveloppe du message (la liste des récepteurs) et h le nombre d'hôtes.

En considérant $id < t$, avec id la taille d'identification d'un récepteur dans l'enveloppe, la seconde solution est moins coûteuse dès que le nombre d'agents par hôte moyen est supérieur ou égal à deux.

6.2.1.c Le serveur de communications

Le serveur est composé de 3 modules principaux :

- *Gestion des entrées* : Les middlewares communiquent avec le serveur par le biais de ce module. Il vérifie, trie et distribue les tâches que l'environnement de communication doit gérer.
- *Gestion des tâches* : Il s'agit de la gestion des tâches de l'environnement, *i.e.* la gestion des descriptions, des filtres et l'appariement.
- *Gestion des sorties* : Une fois l'appariement réalisé, ce module effectue la transmission aux middlewares concernés par le message.

Les *politiques* de priorités du modèle EARI sont pour l'instant fixes au niveau du serveur, elles sont récupérées par un fichier de configuration lors du lancement du serveur de communication.

Le serveur de communication reçoit ses entrées par le module *vérification*, lequel vérifie l'adéquation de l'action demandée aux politiques du serveur, puis transmet au module de gestion des tâches celles qui sont validées.

Le module de gestion des tâches active les sous-modules liés à la modification demandée. S'il s'agit d'une description, le sous-module *état (descriptions)* est mis à jour. S'il s'agit d'un filtre, le sous-module *filtres* est mis à jour. S'il s'agit d'un message, celui est stocké dans le sous-module *état (entités)* et sa description est ajoutée au sous-module *état (descriptions)*.

Lorsque l'un des sous-module est modifié, les méta-informations sont mises à jour. Ces méta-informations sont les structures de données permettant d'accélérer le processus d'appariement (regroupement en classes en fonction des Pdescriptions, par exemple). Ensuite, l'algorithme d'appariement relatif (l'algorithme 2, section 4.4) est déclenché.

Enfin, lorsque l'appariement est réalisé, l'ajout de l'enveloppe (liste des récepteurs) et la transmission des informations aux agents par le biais de leurs middlewares sont réalisés par le module *transmission*.

De fait, les trois modules principaux sont gérés par des *threads* séparés, de façon à découpler les éventuels délais liés au réseau de l'exécution des tâches de l'environnement.

6.2.2 Gestion des filtres et descriptions

6.2.2.a Structure des informations

Le stockage des données nécessaires à l'appariement est effectué dans les sous-modules état (pour les descriptions) et filtres. Leur indexation pour l'accélération des recherches est effectuée grâce au module *méta-informations*. Nous avons fondé la gestion des tâches sur les deux types d'algorithmes proposés en section 4.4. En conséquence, les méta-informations sont les classes d'objets (agents, messages, contexte) et leurs liens à travers les filtres.

Nous rappelons dans le tableau ci-dessous les informations concernées.

Nom	Définition
$channel_m$	$\{f \in F \mid m \in Cat(P_{f_m})\}$
$receiver_m$	$\{a \in \mathcal{A} \mid \exists f \in channel_m, a \in Cat(P_{f_a})\}$
$context_m$	$\{C \subset \Omega \mid \exists f \in Channel_m,\ \forall \omega \in C,\ \omega \in Cat(P_{as_\omega})\}$
$reception_a$	$\{f \in F \mid a \in Cat(P_{f_a})\}$
$fContext_C$	$\{f \in \mathcal{F} \mid \forall \omega \in C,\ \omega \in Cat(P_{as_\omega})\}$

Nous tirons parti du lien entre les Pdescriptions et leurs catégories. En effet, les Pdescriptions forment un treillis de Galois (Fig. 6.6, les propriétés indiquées sont les Pdescriptions des agents *utilisateur*, les Pdescriptions des agents *organisation* et les Pdescriptions des messages d'annonce des évènements). En haut du treillis ($\top$) sont les propriétés communes à toutes les entités (dans l'exemple seulement la propriété *nom*), tandis qu'en bas ($\bot$) se trouvent toutes les propriétés utilisées par au moins une entité. La catégorie de la Pdescription ne contenant que *nom* est l'ensemble des entités.

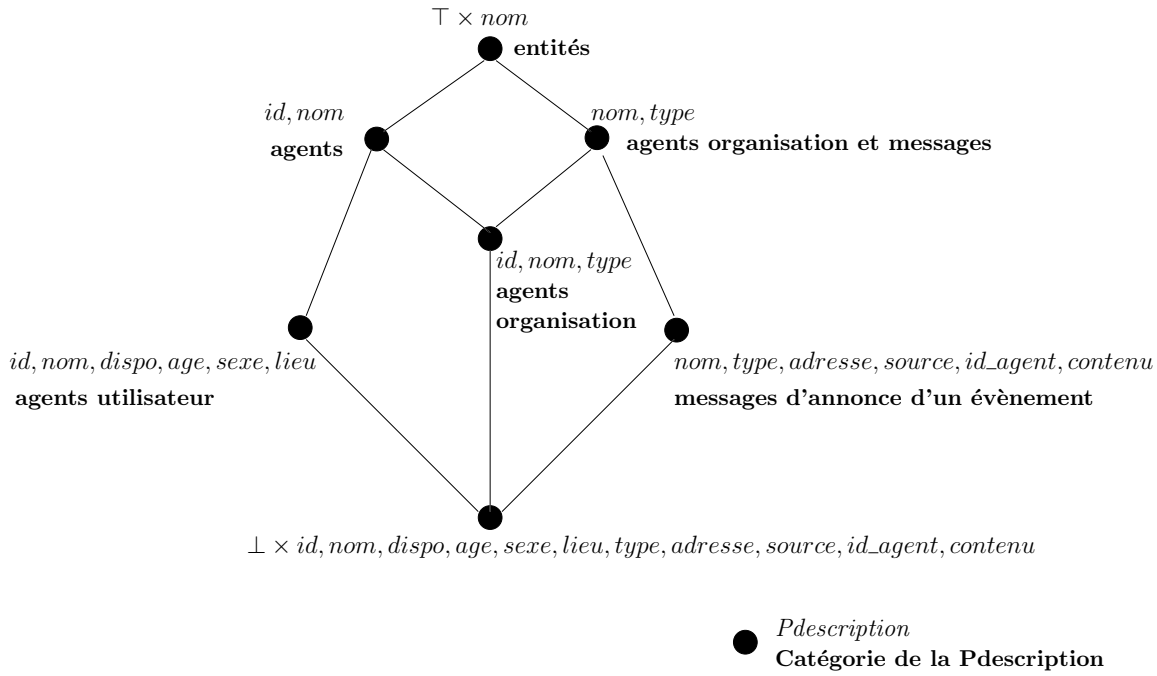


FIGURE 6.6 – Treillis des Pdescriptions

Pour une Pdescription, nous pouvons trouver dans le treillis toutes les Pdescriptions généralisantes (situées plus haut sur le treillis) et toutes les Pdescriptions spécialisantes (situées plus bas). Dans l'exemple de la cité digitale, la Pdescription $\{id, nom\}$ est spécialisée par la Pdescription des agents utilisateurs, et elle est généralisée par la Pdescription $\{nom\}$.

Du point de vue des catégories, cela signifie qu'un agent appartient à la catégorie de sa propre Pdescription et aux catégories de toutes les Pdescription qui généralisent la sienne. Par contre, il n'appartient à la catégorie d'aucune Pdescription qui spécialise la sienne. En contrepoint,

si un filtre s'adresse à une catégorie d'entités, par exemple P_{fa}, alors seuls les agents dont la Pdescription est égale ou spécialise P_{fa} sont concernés. Dans l'exemple de la cité digitale, si P_{fa} est égale à $\{id, nom\}$, les agents concernés sont les catégories des Pdescriptions de tous les noeuds spécialisant P_{fa}, en l'occurrence les agents *utilisateur* et *organisation*.

Nous utilisons ces propriétés du treillis pour classer de façon efficace les Pdescriptions que nous devons stocker à l'aide de structures spécialisées.

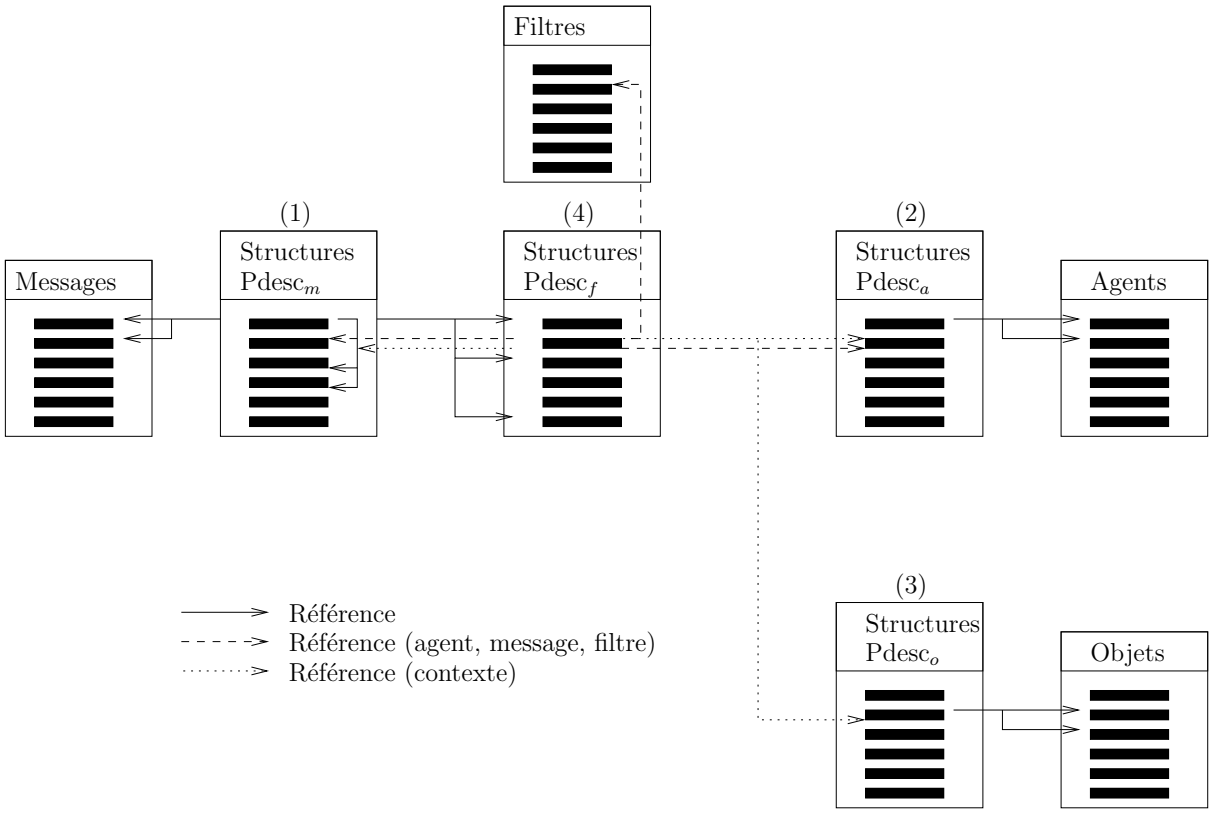


FIGURE 6.7 – Méta-informations stockées par l'environnement

Nous utilisons deux types de structure, illustrées dans la figure 6.7. Le premier est associé aux messages (1), agents (2) et objets(3), tandis que le second est associé aux filtres (4). Nous appelons Pdesc$_m$ la structure associée à chaque Pdescription de message, Pdesc$_a$ la structure associée à chaque Pdescription de message et Pdesc$_o$ la structure associée à chaque Pdescription d'objet. De la même manière, nous appelons Pdesc$_f$ la structure associée à chaque filtre.

Par exemple, à chaque Pdescription de message de $\mathcal{M}$ (liste (1)) est associée une structure dont les éléments sont détaillés en figure 6.8. Le premier élément P_m définit la Pdescription. Le second élément $Cat(P_m)$ permet d'obtenir l'ensemble des descriptions respectant la Pdescription. Le troisième élément *Specialis* permet d'obtenir, par le biais des structures associées, l'ensemble des messages appartenant à l'extension de la Pdescription. Le quatrième élément est exactement $channel_m$, c'est à dire les filtres que cette catégorie de messages peut déclencher. Enfin, le

Pdescription P_m : liste de propriétés de la Pdescription $Cat(P_m)$: liste de liens vers les messages ayant exactement cette Pdescription *Specialis* : liste de liens vers les structures Pdesc_m liées aux Pdescriptions généralisantes $channel_m$: liste de liens vers les structures Pdesc_f liées aux filtres pour lesquelles la Pdescription est égale à P_{fm} $\{f \mid m \in fContext_C\}$: liste de liens vers les structures Pdesc_f liées aux filtres pour lesquelles la Pdescription est égale à au moins une Pdescription d'un objet du contexte P_{as_c}

FIGURE 6.8 – Structure Pdesc_m associée aux descriptions des messages

f : lien vers le filtre $Pdesc_m$: lien vers la structure Pdesc_m liée à la Pdescription du message P_{fm} $Pdesc_a$: lien vers la structure Pdesc_a liée à la Pdescription de l'agent P_{fa} $Pdesc_C$: liste de liens vers les structures Pdesc_a, Pdesc_m et Pdesc_o liées aux Pdescriptions des entités du contexte (agents, messages et objets)

FIGURE 6.9 – Structure $Pdesc_f$ associée aux filtres

cinquième élément $\{f \mid m \in fContext_C\}$ est l'ensemble des filtres f pour lequel l'objet peut faire partie du contexte $fContext_C$.

Une liste de structures similaires est associée aux deux autres ensembles $\mathcal{A}$ (2) et $\mathcal{O}$ (3).

A chaque filtre (Fig. 6.8, liste (4)) correspond une structure de la forme décrite en figure 6.9. Le premier élément permet d'obtenir la syntaxe du filtre. Les deuxième et troisième éléments ($Pdesc_m$ et $Pdesc_a$) sont des références vers les Pdescriptions des messages et agents concernés par le filtre. Le quatrième élément $Pdesc_C$ est une liste de références vers les Pdescriptions des entités composant le contexte, qu'ils soient agents, messages ou objets.

Dans le cadre de l'exemple de cité digitale (figure 6.10), nous explicitons ces structures de données pour le filtre f^+_{groupe}. Lorsque le filtre est ajouté dans la liste de filtres, une structure $Pdesc_f$ lui correspondant est créée. Elle contient un lien vers la définition du filtre f^+_{groupe}, et trois liens vers les structures des Pdescriptions des entités liées à f_a, f_m et f_C. En l'occurrence, f_C est vide, et nous ne détaillons que la structure liée aux agents par soucis de clarté.

Le lien $Pdesc_a$ mène à la structure $Pdesc_a$ liée à la Pdescription $\{dispo, lieu\}$. Aucun agent ne correspond à cette Pdescription, mais il existe dans *Specialis* une Pdescription qui généralise $\{dispo, lieu\}$ et qui contient des agents : $P_{utilisateur} = \{id, nom, dispo, age, sexe, lieu\}$.

La structure $Pdesc_a$ liée à $P_{utilisateur}$ contient un lien vers tous les agents utilisateurs, dans l'exemple $\{\omega_1, \omega_2, \omega_4, \omega_5\}$. Elle contient aussi une liste de liens vers les structures liées aux filtres Pdesc_f pour lesquels $P_{f_a} = P_{utilisateur}$, et une liste de liens vers les $Pdesc_f$ pour lesquels $\exists \omega \in C, P_{as_\omega} = P_{utilisateur}$.

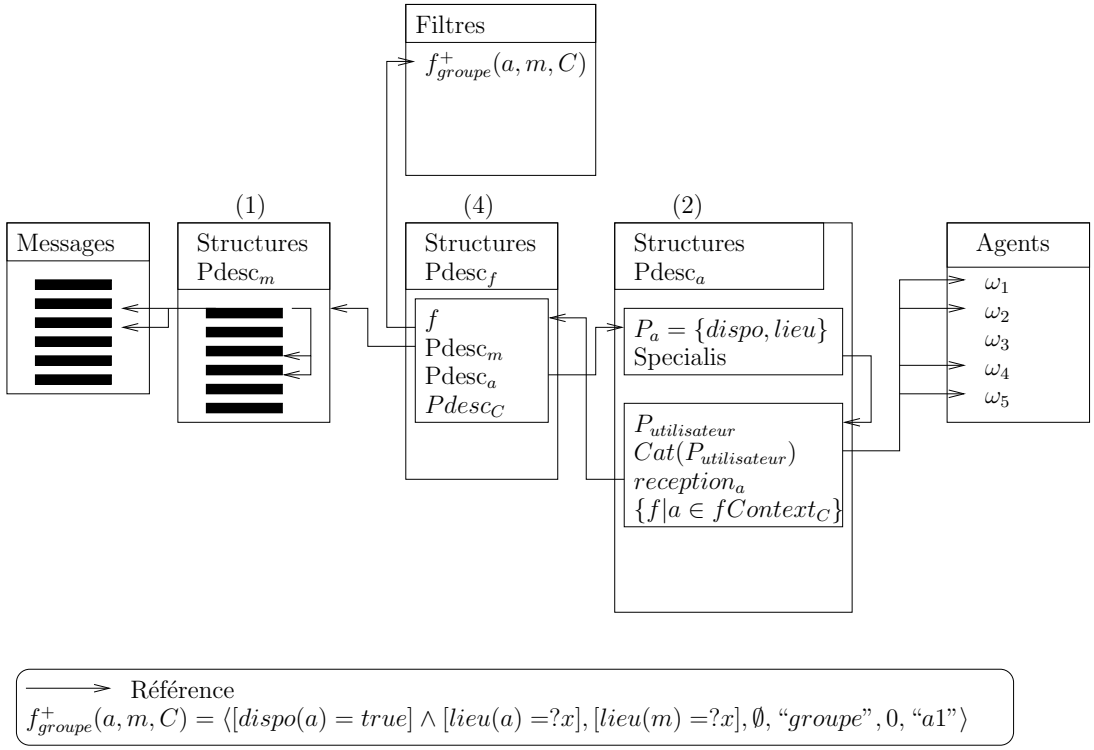


FIGURE 6.10 – Méta-informations : exemple de la cité digitale

6.2.2.b Algorithmes

Nous explicitons maintenant les algorithmes en fonction de cette structuration des données.

Ajout ou modification d'une description- Nous prenons en exemple l'ajout ou la modification d'un message. Lorsqu'un message est ajouté, si la structure $Pdesc_m$ liée à sa Pdescription n'existe pas, celle-ci est créée. Un lien vers sa description est ajouté dans cette structure appartenant à la liste (2). L'algorithme explore alors les filtres appartenant à $channel_m$. Pour chaque filtre, l'algorithme parcourt la liste des agents concernés par le lien avec la structure $Pdesc_a$ liée à P_{f_a}, directement (la structure $Pdesc_a$ contient les liens vers les agents ayant exactement la bonne Pdescription) ou indirectement (par le biais des *Specialis* contenus dans $Pdesc_a$). De la même façon, l'algorithme parcourt l'ensemble des objets du contexte concernés par le filtre. Lors de ce double parcours, l'algorithme effectue l'appariement.

Une fois $Pdesc_m$ parcouru, l'algorithme continue pour $Pdesc_C$. Il parcourt alors les messages et les agents obtenus de la même façon que précédemment pour trouver les éventuels récepteurs.

Ajout ou modification d'un filtre- Lorsqu'un filtre est ajouté, la structure qui lui est liée dans la liste (4) est créée. Ensuite, l'algorithme effectue l'appariement grâce au parcours de tous les messages appartenant à la catégorie de P_{f_m} (via $Pdesc_m$), de tous les agents appartenant à la catégorie de P_{f_a} (via $Pdesc_a$) et pour tous les contexte potentiels (via $Pdesc_C$).

Langage de balisage pour l'expression des filtres et des descriptions

Le langage choisi est une extension de RuleML [24]. Le *Rule Markup Language* est une initiative visant à proposer un langage de règles standardisé fondé sur XML. L'un des objectif est, grâce aux règles de transformation, de permettre une meilleure interopérabilité et substituabilité entre systèmes. Par exemple, l'environnement de développement de systèmes à base de règles JESS est désormais compatible avec un format d'entrées-sorties XML.

Le principal ajout que nous avons réalisé est une balise `modify`, afin d'améliorer l'efficacité. Modifier un fait est moins complexe qu'un retrait suivi d'un ajout, bien que le résultat soit équivalent. En effet, la modification de la valeur d'une propriété n'entraîne aucun changement au niveau de la structure associée à sa Pdescription, tandis qu'un retrait ou un ajout affectent cette structure.

Le langage de balisage est décrit dans l'annexe C.

6.3 Une bibliothèque pour la plate-forme MadKit

Les modèles EASI et EARI sont indépendants de l'implémentation, ce qui permet de les mettre en oeuvre dans différents cadres et avec différents outils. Après avoir présenté un prototype de support du modèle EASI en section 6.2, nous introduisons dans cette partie de quelle façon l'environnement de communication peut être intégré à une plate-forme existante sous forme d'extension de celle-ci.

Pour cela, nous avons choisi la plate-forme MadKit [Gutknecht et Ferber, 2000], qui s'appuie sur le modèle Agent-Groupe-Rôle (AGR). L'avantage d'intégrer EASI à une plate-forme est de permettre au concepteur multi-agents de mettre en oeuvre le modèle facilement, grâce aux bibliothèques de développement et aux services déjà existants, comme la gestion du cycle de vie des agents. Dans cette partie, nous introduisons tout d'abord le modèle AGR et la plate-forme MadKit, et discutons ce choix en regard du modèle EASI. Dans un deuxième temps, nous présentons les fonctionnalités offertes par le paquetage (*package*) implémenté. Enfin, nous introduisons la gestion de la dynamique de l'environnement par le biais de l'environnement de développement de systèmes à base de règles JESS.

6.3.1 Le modèle Agent-Groupe-Rôle et MadKit

Nous avons introduit brièvement en section 2.2.2 le modèle AGR et MadKit. MadKit est une plate-forme d'exécution de systèmes multi-agents, fondée sur le méta-modèle organisationnel Aalaadin [Ferber et Gutknecht, 1998].

Aalaadin est une méthodologie générique incluant deux niveaux conceptuels : concret, incluant les notions d'agent, groupe et rôle, et abstrait, décrivant les interactions valides et la structure des groupes, de façon totalement indépendante de l'architecture des agents. Le

24. `http://www.ruleml.org/`

système multi-agents possède ainsi une structuration de plus haut niveau qu'une simple collection d'agents. L'organisation est un ensemble d'activités et d'interactions définies par des groupes, rôles et leurs relations. Un agent peut faire partie de plusieurs groupes, et il prend un rôle particulier dans chacun des groupes auxquels il participe. Ce rôle est une représentation abstraite d'une fonction ou d'un service.

Du point de vue du modèle d'interaction, le modèle organisationnel est intéressant pour deux raisons :

- La modélisation des interactions définit des relations entre rôles et non entre agents. Ainsi, le récepteur d'une communication n'est pas seulement sélectionné en fonction de son adresse, mais aussi en fonction de son groupe et de son rôle.
- L'organisation implique également des contraintes de communication. Ainsi, deux agents ne peuvent communiquer directement que s'ils font partie d'un même groupe.

La plate-forme MadKit est fondée sur les concepts du modèle AGR. Son architecture est basée sur un micro-noyau qui sert d'environnement d'exécution des agents. En particulier, il gère le cycle de vie des agents, la transmission des messages et le contrôle des groupes et rôles locaux. Deux avantages de MadKit sont la généricité de la plate-forme, celle-ci n'étant pas dépendante d'un domaine d'applications, et la simplicité du micro-noyau qui permet un développement rapide du support pour le modèle EASI.

Le modèle EASI présente une forte analogie avec le modèle AGR et la plate-forme MadKit, récapitulée en figure 6.11. Dans les deux cas, l'organisation est décrite dans l'environnement, l'une par le biais de propriétés et l'autre par le biais des groupes et rôles. Dans les deux cas, la distribution des messages est un rôle confié à l'environnement, selon un adressage qui n'est pas seulement lié à l'identifiant des agents. MadKit gère les notions de groupe et de rôle. Avec EASI, nous pouvons considérer les groupes et rôles comme des propriétés particulières qui décrivent les agents. Enfin, les contraintes de communication de MadKit sont pré-définies dans le modèle AGR, elles sont donc fixées *a priori* et ne changent pas. Ces contraintes peuvent être exprimées dans EASI par des filtres négatifs. De plus, les règles de transmission peuvent être modifiées dynamiquement.

AGR / MadKit	EASI
Description de l'organisation (plate-forme)	Description de l'organisation (environnement)
Délégation de la distribution des messages	Délégation de la distribution des messages
Adressage par groupe et rôle	Adressage via les propriétés
Contraintes de communication pré-définies	Contraintes de communication dynamiques

FIGURE 6.11 – Correspondances entre le modèle AGR et le modèle EASI

Sur l'aspect interaction, le modèle EASI permet d'étendre le modèle AGR, et le principe du support est similaire à MadKit. Nous montrons comment modéliser les interactions du modèle AGR avec EASI.

Les agents possèdent des rôles dans des groupes distincts, qui sont gérés par un *leader* de

groupe. Nous proposons de modéliser les appartenances aux organisations par des descriptions d'entités abstraites, qui possèdent trois propriétés $\{groupe, role, id\}$. Les descriptions correspondant à un groupe donné sont gérées par le créateur du groupe. A chaque fois qu'un agent rejoint le groupe, une nouvelle description est donc ajoutée dans l'environnement, avec pour valeurs le nom du groupe, le nom du rôle et l'identifiant de l'agent.

Les messages échangés peuvent comporter un rôle et un groupe de destination au lieu de l'adresse du récepteur. La composante interactionnelle du modèle AGR régule donc les communications de la façon suivante :

- Les agents peuvent communiquer par leur adresse :
  $\langle [id(a) =?x], [receiver(m) =?x], \text{“}direct'', 0, environnement\rangle$
- Les agents peuvent adresser leurs messages par les critères groupe et rôle :
  $\langle [id(a) =?i], [role(m) =?r] \wedge [groupe(m) =?g], [role(c) =?r]] \wedge [groupe(c) =?g]] \wedge [id(c) = ?i], \text{“}role'', 0, environnement\rangle$
  L'appariement est réalisé entre l'identifiant de l'agent, le rôle et le groupe du message, et les descriptions de l'organisation.
- Les agents ne peuvent communiquer que s'ils appartiennent à un groupe commun, autrement dit la communication est bloquée s'il n'existe pas de groupe en commun :
  $\langle [id(a) =?i], [sender(m) =?s], [not([groupe(c1) =?f] \wedge [groupe(c2) =?f] \wedge [id(c1) =?i] \wedge [id(c2) =?s]), \text{“}group'', 1, environnement\rangle \in \mathcal{F}^-$
  Le filtre négatif bloque la réception si l'émetteur et le récepteur n'ont pas au moins un groupe en commun.

Le modèle AGR a été étendu pour prendre en compte l'environnement. Le modèle AGRE (*AGR + Environment*), [Ferber et al., 2005]) pose les principes de cette extension et le modèle AGREEN (*AGR Environnement et Normes*, [Baez-Barranco et al., 2007]) formalise le modèle AGRE. Ces modèles prennent en compte l'environnement comme abstraction de premier ordre pour modéliser à la fois le contexte spatial et le contexte social du système multi-agents. Du point de vue des interactions, il est fondé sur une forte distinction entre les influences des agents et leurs effets. Cette approche se rapproche de celle de [Platon et al., 2007b], que nous avons vue en section 1.2.3. Les agents essaient de modifier l'environnement, et celui-ci réagit en fonction de ses règles propres.

Cet aspect physique est unifié à l'aspect social du système multi-agents, grâce à la distinction entre ce que les agents sont capables de faire et les normes sociales en vigueur dans le système. L'environnement est ici appelé *institution*, il vérifie le respect des normes par les agents et éventuellement les pénalise. Dans cette optique, le modèle AGR est un environnement social particulier parmi d'autres.

Le modèle AGREEN est une généralisation du modèle AGR sur ses aspects organisationnels et interactionnels, avec une attention particulière portée au contrôle social et spatial. Cependant, les auteurs n'ont pas pris en compte les interactions entre agents par la communication. Parallèlement, le modèle EASI est une généralisation du modèle AGR du point de vue des communications.

Une seconde différence se situe dans le contrôle des interactions : alors que EARI contrôle et bloque *a priori* les messages non-conformes aux règles en vigueur, AGREEN l'envisage sous forme de normes et de sanctions sociales. De plus, dans AGREEN la séparation entre influences des agents et réactions de l'environnement ne permet pas à l'agent d'utiliser l'environnement pour moduler *a priori* ses interactions.

6.3.2 Vue d'ensemble du paquetage EASI

Le paquetage EASI pour MadKit a été développé de façon à permettre les communications multi-parties tout en bénéficiant d'une infrastructure déjà existante. Il s'agit donc d'une bibliothèque pour la mise en oeuvre du modèle EASI, contenant à la fois l'environnement et les primitives d'accession à celui-ci. Les messages sont transmis directement dans la boîte aux lettres des agents, comme pour une communication par le noyau.

La distribution des messages est effectuée par l'environnement et non plus seulement par le noyau MadKit, comme l'illustrent les figures 6.12 et 6.13. La plate-forme distribue les messages par invocation d'une méthode du micro-noyau, laquelle effectue la sélection des récepteurs et dépose le message dans leurs boîtes aux lettres. Dans le cadre du paquetage EASI, le noyau n'est utilisé que lors de la phase d'initialisation des agents, pour découvrir l'environnement. L'agent peut choisir pour communiquer d'utiliser soit le noyau de MadKit, soit l'environnement.

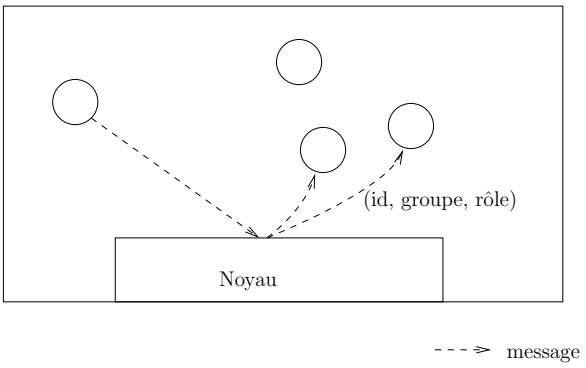


FIGURE 6.12 – Fonctionnement initial de MadKit

La figure 6.14 donne le passage de l'architecture abstraite introduite en début de chapitre à cette implémentation, en précisant les différents flux d'information dans le paquetage. Les modules fonctionnels sont répartis entre deux composants, l'environnement et le système-expert.

Nous avons choisi l'environnement de développement de systèmes à base de règles JESS [Hill, 2003] pour implémenter l'environnement de communication. Il s'agit de systèmes experts fondés sur les arbres RETE [Forgy, 1982] écrit en Java. Nous avons choisi les arbres RETE car, tout comme nos propres algorithmes (section 4.4), il sont fondés sur une pré-classification des entités pour améliorer l'efficacité de l'appariement.

Du fait de la philosophie MadKit du "tout est agent", l'environnement est encapsulé en tant qu'agent. Il est activé par les agents localement par une invocation directe. Il vérifie la syntaxe

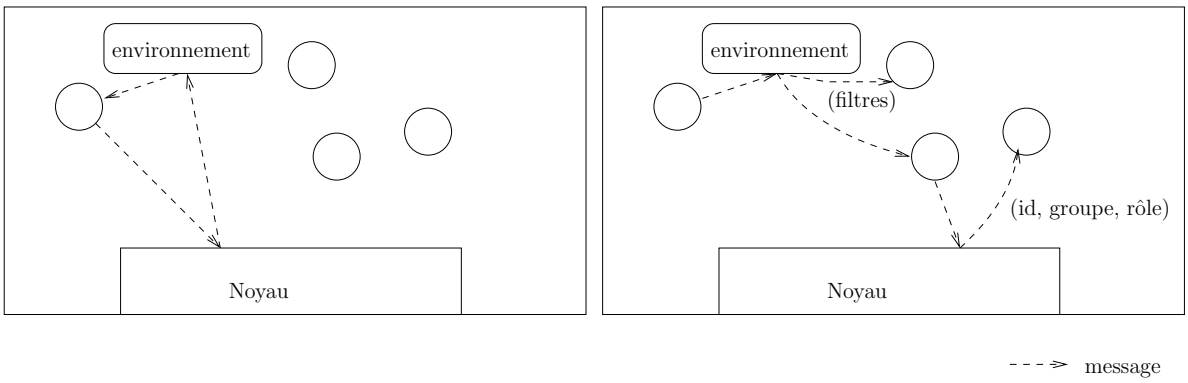


FIGURE 6.13 – Initialisation et fonctionnement du paquetage EASI

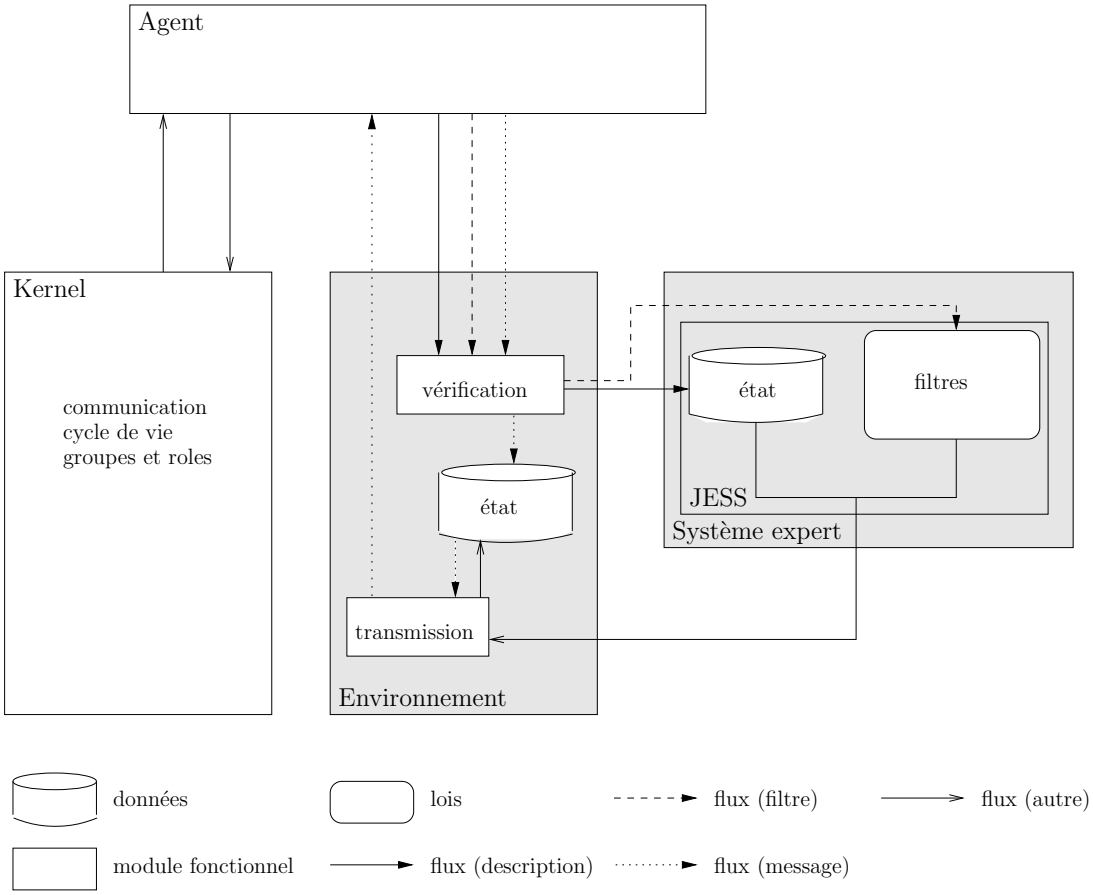


FIGURE 6.14 – Flux d'informations du paquetage pour MadKit

des filtres et des descriptions qui lui sont transmis, puis modifie la base de règles (respectivement de faits) du système-expert. Lorsqu'un message est ajouté, celui-ci est stocké le temps de l'appariement. Lorsqu'un couple agent/message est trouvé par le système-expert, celui-ci active la transmission du message pour l'agent via l'environnement.

Le paquetage a été développé en Java, nous détaillons son contenu en annexe D. L'environ-

nement se charge de faire le lien entre les actions des agents et la gestion du système-expert. Il est à noter que dans cette implémentation, il est possible de créer plusieurs environnements de communication fonctionnant en parallèle, comme cela a été réalisé dans la section 7.1.2.

Une implémentation avec JESS Nous introduisons ici les équivalences entre les actions des agents et leur répercussion sur le système expert.

En langage objet, les catégories d'entités sont des classes différentes. A cette fin, nous déclarons les différentes classes avant de les utiliser pour l'appariement :

```
(defclass agent EASIAgent)
```

Une fois les classes définies, chaque entité est ajoutée grâce à un appel à *definstance* ou enlevée (*undefinstance*). Un raccourci intéressant proposé par JESS est la traduction de tous les attributs d'une classe implémentée sous forme de Java *Beans* en tant que champs de l'objet dans le système. De la même façon, la mise à jour est réalisée de façon automatique via la fonction *update*.

Les filtres sont de la forme (partie gauche implique partie droite), auxquels sont ajoutés le nom et la priorité du filtre.

```
Arguments: name, priority, lhs, rhs

(defrule name
         lhs
         (not
           (exists (messageEnvironment (id ?id) (received ?receiver))
         ))

         => rhs
           (assert (messageEnvironment (id ?id) (received ?receiver)))
         )
```

Nous avons choisi d'ajouter à chaque message une liste de faits indiquant les agents ayant déjà reçu ce message. Grâce à cela, un même agent ne reçoit pas plusieurs fois le même message si plusieurs filtres sont valides. Il est à noter que dans certains contextes applicatifs, il peut être préférable de laisser l'agent recevoir plusieurs fois le même message grâce à des filtres différents. En effet, la connaissance du filtre par lequel la transmission est réalisée apporte une information supplémentaire à l'agent sur son contexte. Ce fonctionnement est le cadre général de l'algorithme.

Le retrait s'effectue à l'inverse par un *undefrule*.

Un exemple de règle écrite avec le langage JESS est un filtre pour l'écoute flottante posé par un agent $a1$:

```
(defrule ecoute
           (agent (id ?receiver)  (OBJECT ?a))
           ?me<-(messageedt  (id ?id) (OBJECT ?m)(sender ?sender))
           (agent (id ?sender) (type "client" ))
```

```
    (test (eq ?receiver "a1"))
  (not
    (exists (messageEnvironment (id ?id) (received ?receiver))
  ))

  => (call ?a perceive ?m)
    (assert (messageEnvironment (id ?id) (received ?receiver)))
  )
```

Le terme *OBJECT* permet de désigner un objet du langage JAVA. Cette règle permet à l'agent $a1$ d'écouter tous les messages émis par les agents dont la propriété *type* a pour valeur *client*.

6.4 Conclusion

Dans ce chapitre, nous avons introduit différents outils de mise en oeuvre de notre modèle d'environnement de communication. Nous avons tout d'abord introduit une architecture abstraite de l'environnement, et développé les différents modules afférants. Nous avons discuté son lien avec l'architecture de référence proposée par [Weyns et al., 2007], et une architecture d'agent minimale.

Ensuite, nous avons proposé un prototype d'environnement de communication réalisé en Java. La distribution du système multi-agents est réalisée par une architecture client-serveur. Nous avons étendu le langage RuleML pour obtenir une description satisfaisante des filtres et montré comment la gestion des algorithmes proposés en section 4.4 est effectuée par des structures appropriées.

Enfin, nous avons montré l'intégration d'une bibliothèque pour l'environnement de communication au sein d'une plate-forme existante - MadKit - et les effets sur son fonctionnement. Nous avons également proposé une gestion des descriptions et filtres par le biais d'un système expert.

Les implémentations des modèles EASI et EARI ont donné lieu à des expérimentations, lesquelles sont exposées dans le chapitre suivant. Ensuite, nous approfondissons la façon d'exploiter nos modèles grâce à des protocoles opportunistes, et nous montrons comment le modèle EARI permet la mise en oeuvre de modèles d'interactions normés.

Chapitre 7

Expériences et éléments d'exploitation des modèles

Sommaire

Dans ce chapitre, nous introduisons tout d'abord les résultats des expérimentations menées sur la gestion des communications multi-parties. En particulier, nous étudions l'implémentation de l'environnement par un arbre RETE, ainsi que par nos algorithmes fondés sur la pré-classification des entités.

En seconde partie, nous exposons des éléments d'exploitation des modèles EASI et EARI. Nous proposons une nouvelle notation AUML pour les communications multi-parties, et l'utilisons pour la définition de protocoles opportunistes. Nous montrons ensuite comment instancier au sein de l'environnement un modèle normé fondé sur la logique déontique.

7.1 Expérimentation du modèle EASI à l'aide d'un système expert

Nous avons décrit en section 6.3 une bibliothèque pour la mise en oeuvre du modèle sur la plate-forme MadKit. Comme nous l'avons souligné en section 6.13, les filtres peuvent être considérés comme des règles, et les descriptions des entités comme des faits. Nous nous intéressons donc ici aux performances liées à l'utilisation d'un système-expert pour gérer l'environnement.

Un système à base de règles est constitué de trois parties, le moteur d'inférence, la base de règles et la base de faits. Un fait peut déclencher une règle, et ce processus est géré par le moteur d'inférence. Nous avons choisi de modéliser les entités en tant que faits, et les conditions de réception en tant que règles. En pratique, l'ajout d'une entité *Message* pourra donc déclencher une ou plusieurs règle(s) d'envoi de ce message. L'agent *Environnement*, une fois lancé, initialise le système expert, puis ne gère que les inscriptions des agents dans l'environnement, par un échange initial de messages avec l'agent demandeur. La bibliothèque est centralisée, ce qui permet aux agents d'utiliser directement les méthodes de l'environnement.

7.1.1 Implémentation du modèle par un arbre RETE

Le système expert que nous avons choisi est JESS [25] [Friedman-Hill, 1998]. JESS est un moteur d'inférence performant, fondé sur l'algorithme RETE [Forgy, 1982; Wright et Marshall, 2003]. RETE permet d'améliorer la rapidité de l'appariement des faits. Il utilise un réseau de discrimination qui représente les dépendances entre les conditions des règles sur les faits. Ce réseau est une façon de pré-classifier les entités en fonction des règles, ce qui rapproche RETE du principe de nos algorithmes et le rend intéressant pour l'implémentation du modèle d'environnement.

Un arbre RETE appliqué à un filtre d'EASI est montré en figure 7.1. Cet arbre peut être décomposé en plusieurs parties, T1 est lié aux tests concernant les agents, T2 est lié aux tests concernant les messages, T3 est lié aux entités du contexte, T4 représente l'appariement entre les agents et les messages résultant des noeuds précédents et T5 les tests entre les couples message/agent résultants et les éléments du contexte.

Lorsqu'un fait est ajouté ou modifié, il "descend" les noeuds de tests de l'arbre. Par exemple, lorsqu'un nouveau message arrive, il est ajouté à l'ensemble *Messages*, puis les tests de T2 lui sont appliqués. S'il passe les tests, une référence est ajoutée en T_m. Ensuite, il passe les tests de T4, et ainsi de suite pour les tests de T5 tant que le résultats des tests sont positifs.

Reprenons l'exemple cité digitale du filtre f_4 permettant à l'agent a_x de ne recevoir les évènements de type concert que si ceux-ci ont lieu dans un restaurant :
$f^{+}_{concert}(a, m, \{c\}) = \langle [id(a) = a_x], [type(m) = \text{“}concert\text{”}] \wedge [lieu(m) = ?x], [nom(c) = ?x] \wedge [type(c) = \text{“}restaurant\text{”}], \text{“}concert\text{”}, 0, a_x \rangle$

25. `http://www.jessrules.com/jess`
La bibliothèque JESS est libre d'accès dans le cadre d'une utilisation académique.

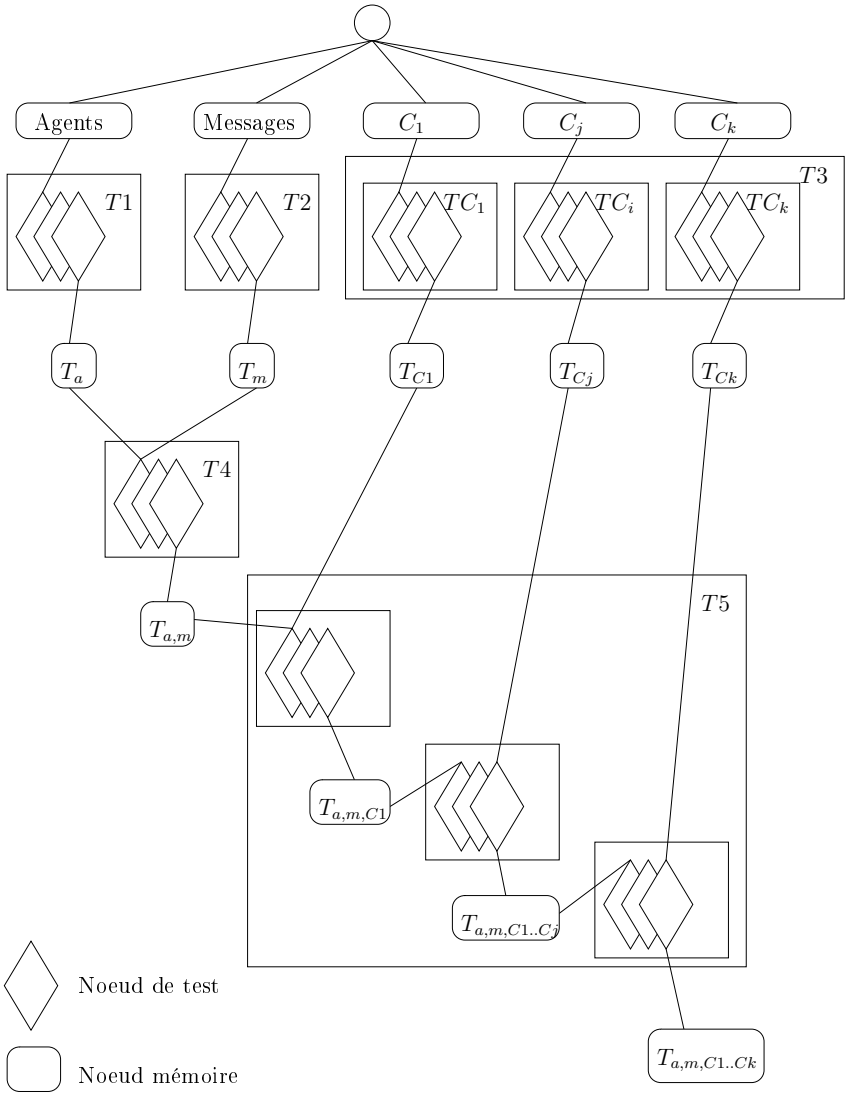


FIGURE 7.1 – RETE appliqué à un filtre du modèle EASI

L'arbre correspondant au filtre $f^{+}_{concert}$ est le suivant : le noeud T1 contient le test $[id(a) = a_x]$ et le noeud T2 contient le test $[type(m) =$ "$concert$"$]$. Seul un élément du contexte est apparié, il y aura donc un seul élément de contexte C, et le test en T3 sur cet élément est $[type(c) =$ "$restaurant$"$]$.

Dans ce filtre, l'agent et le message ne sont pas directement appariés, il n'y a donc pas de test T4. On obtient donc en noeud $T_{a,m}$ tous les messages de type "concert" couplés à l'agent a_x.

L'appariement des messages avec les descriptions du contexte est réalisé dans le premier noeud de T5 par $[lieu(m) =?x] \wedge [nom(c) =?x]$. C'est le dernier test du filtre, et pour tous les triplets $< a_x, m, c >$ ayant passé ce test, le message m est transmis à a_x.

Lorsqu'un objet est retiré, il doit être ôté de tous les noeuds mémoires dans lesquels il se

trouve, y compris les enregistrements où il apparaît couplé suite à des appariements. De même, lorsqu'un fait est modifié, tous les tests sont à nouveau effectués pour pouvoir éventuellement ajouter des références dans les noeuds situés en dessous, ou en retirer.

7.1.2 Expériences

Nous avons réalisé les tests d'efficacité de RETE pour la mise en oeuvre du modèle EASI dans le cadre de plusieurs scénarios, afin d'observer son comportement sur des cas typiques. Ces tests ont été faits à l'aide de la bibliothèque et de l'environnement décrits en section 6.3. Par ailleurs, nous étudions le cas où l'on substitue plusieurs environnements de communications à un environnement unique, et discutons les conséquences de ce choix.

7.1.2.a Description de la simulation

Dans la cité digitale, nous considérons des agents *utilisateur*. Trois propriétés sont utilisées dans les tests : leur identifiant *id*, qui est unique, leur localisation *lieu*, et leur disponibilité. Les agents sont répartis entre trois localisations ("Arcueil", "Dauphine", "Vincennes"), et la moitié des agents dans chaque endroit est disponible. Nous obtenons donc six classes d'agents de tailles égales, avec N le nombre d'agents :

	$lieu(a) =$ "$Arcueil$"	$lieu(a) =$ "$Dauphine$"	$lieu(a) =$ "$Vincennes$"
$dispo(a) = true$	$\frac{1}{6}N$	$\frac{1}{6}N$	$\frac{1}{6}N$
$dispo(a) = false$	$\frac{1}{6}N$	$\frac{1}{6}N$	$\frac{1}{6}N$

Trois scénarios ont été implémentés :

- Dans le premier scénario, les agents envoient des messages "request". Ces messages intéressent les agents localisés au même endroit que l'émetteur, et ayant une disponibilité différente de celle de l'émetteur (si l'émetteur est indisponible, il recherche des récepteurs disponibles, et inversement). Les agents contactés répondent par un message "accept" ou un message "reject". Chaque message "request" intéresse donc $\frac{1}{6}N$ agents, tandis que la réponse est fondée sur l'identifiant, et n'est donc destinée qu'à un seul agent.
- Dans le deuxième scénario, les agents envoient des messages "propose". Ces messages sont intéressants pour les agents ayant une localisation différente de l'émetteur, mais la même disponibilité. Les agents répondent par un message "accept" s'ils sont disponibles. L'initiateur choisit alors un partenaire et initie avec lui un protocole dans lequel ils échangent 40 messages, durant lequel ils sont tous deux indisponibles. Dans ce cas, les messages "propose" sont intéressants pour au maximum $\frac{2}{6}N$ agents, et les autres messages échangés n'intéressent qu'un seul récepteur.
- Dans le troisième scénario, chaque agent échange des messages adressés avec un seul partenaire, en fonction de son identifiant. Tous les messages échangés intéressent un et un seul récepteur.

Nous avons choisi des classes d'agents équitablement réparties et ces scénarios spécifiques de façon à pouvoir exploiter les résultats en fonction de la proportion d'agents du système intéressés par le message. Cette proportion d'agents est ce que nous appelons le *taux de discrimination* d'un filtre, autrement dit le nombre d'agents recevant le message divisé par le nombre total d'agents.

Nous avons effectué trois séries de tests : avec le modèle de diffusion, avec un environnement (EASI-1), et avec plusieurs environnements (EASI-3).

Dans la première série de tests, chaque message est reçu par tous les agents, qui filtrent localement les messages en fonction des propriétés de l'émetteur et de leurs propres propriétés. Dans les deux autres séries, l'appariement est réalisé dans l'environnement par le système-expert JESS.

Pour EASI-1, un environnement est exécuté et gère l'ensemble des filtres et descriptions. Trois filtres sont utilisés : le filtre f^+_{direct} qui gère les communications adressées, le filtre $f^+_{request}$ qui gère les messages "request" et le filtre $f^+_{propose}$ qui gère les messages "propose".

Nous rappelons la syntaxe du filtre pour les communications adressées :
$f^+_{direct}(a, m, C) = \langle [id(a) = ?x], [receiver(m) = ?x], \emptyset, \text{“}direct\text{”}, 0, environnement \rangle$

Les deux autres filtres s'appuient sur le type du message et des tests sur le contexte, relatifs aux propriétés *lieu* et *dispo* des émetteurs et récepteurs :
$f^+_{request}(a, m, \{c\}) = \langle [lieu(a) = ?l] \wedge [dispo(a) = ?d1], [sender(m) = ?a2] \wedge [type(m) = \text{“}request\text{”}],$
$[id(c) = ?a2] \wedge [lieu(c) = ?l] \wedge [dispo(c) \neq ?d1], \text{“}request\text{”}, 0, environnement \rangle$

Dans le scénario 1, le filtre $f^+_{request}$ teste, pour chaque agent ayant une localisation $?l$ et disponibilité $?d1$, si l'émetteur $?a2$ du message de type "request" est situé dans le même endroit $[lieu(c) = ?l]$ et a une disponibilité différente $[dispo(c) \neq ?d2]$.

Le filtre $f^+_{propose}$ a une syntaxe similaire :
$f^+_{propose}(a, m, \{c\}) = \langle [lieu(a) = ?l] \wedge [dispo(a) = ?d1], [sender(m) = ?a2] \wedge [type(m) = \text{“}propose\text{”}],$
$[id(c) = ?a2] \wedge [lieu(c) \neq ?l] \wedge [dispo(c) = ?d1], \text{“}propose\text{”}, 0, environnement \rangle$

Dans le scénario 2, ce filtre teste, pour chaque agent ayant une localisation $?l$ et disponibilité $?d1$, si l'émetteur $?a2$ du message de type "request" est situé dans un endroit différent $[lieu(c) \neq ?l]$ et a la même disponibilité $[dispo(c) = ?d2]$.

Nous avons également voulu étudier de quelle façon l'utilisation de plusieurs environnements peut permettre d'alléger la charge de l'environnement. Nous souhaitons cependant ne pas perdre d'informations, c'est à dire que des agents ne reçoivent plus certains messages du fait de cette division. Utiliser plusieurs environnements a deux impacts principaux : limiter le nombre de descriptions et de filtres par environnement, et éventuellement les simplifier. Nous détaillons les deux choix possibles de répartition dans plusieurs environnements et leurs conséquences respectives :

1. Les filtres sont répartis entre les environnements. Dans chaque environnement, il est alors possible de simplifier les descriptions des entités aux seules propriétés testées ou appariées dans les filtres de cet environnement. Une difficulté apparaît cependant : un message doit

être transmis dans tous les environnements où sa description est potentiellement valable. Cela crée une redondance des traitements de la description des messages dans chacun des environnements où ils sont transmis. Plus encore, c'est le cas pour toutes les descriptions d'entités, telles celles des agents. Dans notre exemple, tous les agents ont les propriétés *id*, *lieu* et *dispo*. S'il y a un environnement pour chaque filtre (f^+_{direct}, $f^+_{request}$ et $f^+_{propose}$), alors tous les agents doivent maintenir leurs descriptions dans chaque environnement. En effet, ils peuvent tous recevoir des messages grâce à f^+_{direct} (du fait de leur propriété *id*), et grâce à $f^+_{request}$ et $f^+_{propose}$ (du fait de leurs propriétés *lieu* et *dispo*).
En conséquence, cette répartition permet un gain limité, liés aux descriptions éventuellement restreintes des entités. La contrepartie est une redondance des messages et des descriptions dans chacun des environnements. Il n'y a pas de différence dans le processus de test et d'appariement.

2. Les descriptions sont réparties dans les environnements. Nous avons souligné que la répartition ne devait pas restreindre les agents recevant les messages, cela implique donc que dans chaque environnement se trouvent les descriptions liées aux filtres de l'environnement.
   – Si tous les agents peuvent potentiellement communiquer à tout instant avec tous les autres agents, ou peuvent faire partie du contexte d'un filtre, alors toutes les descriptions doivent faire partie de tous les environnements. Il n'y a alors pas de gain mais une recopie entre plusieurs environnements, les messages pouvant être transmis par n'importe lequel. C'est un avantage en terme de robustesse puisqu'une panne d'un environnement n'empêche pas de continuer à transmettre les messages, mais le coût de traitement est plus important qu'avec un seul environnement.
   – s'il est possible d'extraire du système multi-agents des situations dans lesquelles des classes d'agents ne communiquent pas entre elles, il est alors possible de séparer leurs descriptions dans des environnements séparés. Il n'y a alors pas de redondance des descriptions à cause de la répartition.
     Outre la répartition, un second gain peut être attendu de cette séparation : si les filtres portent dans leurs conditions des tests discriminants, par exemple dans le scénario 1 les agents localisés au même endroit, alors les deux classes ne communiquent pas ensemble bien qu'elles soient dans le même environnement. Regrouper les agents par classe permet donc de diminuer le nombre de tests des filtres, puisque seuls les agents de la bonne classe font partie de l'environnement. Par exemple, si les agents sont répartis par localisation, il n'est pas nécessaire de faire d'appariement entre les propriétés *lieu* de l'émetteur et du récepteur pour savoir s'ils ont la même localisation.
     Dans ce cas, il est donc possible d'obtenir des gains sur le processus de tests et appariements lui-même, ainsi que le nombre de descriptions par environnement.

Nous étudions ce dernier cas dans nos expériences. Dans le premier scénario, seuls les agents localisés au même endroit communiquent entre eux, or la propriété *lieu* peut prendre trois valeurs. Il y a donc besoin de trois environnements. Dans le second scénario, seuls les agents ayant la

même disponibilité communiquent ensemble, il y a donc besoin de deux environnements. Dans le dernier cas, il peut y avoir jusqu'à $\frac{N}{2}$ environnements, puisque les agents communiquent deux à deux.

Nous détaillons alors les différences avec le modèle EASI-1.

Dans le premier scénario, les agents sont regroupés par localisation, il y a donc $\frac{2}{6}N$ agents par environnement. Les agents communiquent avec ceux qui ont une disponibilité différente, donc dans chaque environnement les messages "request" sont transmis à 50% des agents. Le filtre $f^+_{request}$ est simplifié comme suit :
$f^+_{request'}(a, m, \{c\}) = \langle [dispo(a) = ?d1], [sender(m) = ?a2] \wedge [type(m) = \text{“}request\text{”}], [id(c) = ?a2] \wedge [dispo(c) \neq ?d1], \text{“}request'\text{”}, 0, environnement \rangle$

La substitution de la propriété *lieu* du récepteur par une variable et son appariement avec la propriété *lieu* de l'émetteur sont inutiles avec les trois environnements, ils n'apparaissent donc pas dans le nouveau filtre $f^+_{request'}$.

Dans le second scénario, les agents sont d'abord regroupés dans deux environnements par groupes de disponibilité lors de la recherche de partenaire. Il y a donc $\frac{1}{2}N$ agents par environnements, et dans chaque environnement les messages "propose" sont envoyés à 66% des agents (ceux dans une localisation différente de l'émetteur). Le filtre $f^+_{propose}$ est simplifié comme suit :
$f^+_{propose'}(a, m, \{c\}) = \langle [lieu(a) = ?l], [sender(m) = ?a2] \wedge [type(m) = \text{“}propose\text{”}], [id(c) = ?a2] \wedge [lieu(c) \neq ?l], \text{“}propose'\text{”}, 0, environnement \rangle$

De la même façon que dans le premier scénario, la substitution de la propriété *dispo* du récepteur par une variable et son appariement avec la propriété *dispo* de l'émetteur du filtre sont inutiles avec les deux environnements.

Lorsqu'un agent trouve un partenaire, ces deux agents changent d'environnement vers un troisième environnement dédié aux communications en couple, pour ne pas recevoir les messages "propose" alors qu'ils sont indisponibles. Lorsque le protocole d'échange de messages est achevé, ils retournent dans leur environnement initial.

Dans le troisième scénario, nous avons choisi de diviser les agents en trois groupes comportant les couples communiquant entre eux, pour obtenir un nombre d'agents par environnement comparable avec les autres scénarios. Il y a à nouveau $\frac{2}{6}N$ agents par environnement.

Chaque test est une série de trois fois chaque scénario. Nous fixons la durée totale et observons le nombre de messages échangés sur cette période. Un message envoyé à n agents est compté une seule fois. De cette façon, le nombre de messages est directement lié au nombre de protocoles achevés par les agents. Il ne prend pas en compte les messages reçus par des agents alors que ceux-ci ne sont pas intéressés.

7.1.2.b Résultats

Les tests sont réalisés de façon centralisée pour pouvoir comparer le coût du filtrage par le système expert avec le coût du filtrage *a posteriori* par chacun des agents dans le cas de la diffusion.

Les résultats présentés en figure 7.2 montrent une moyenne de 45 tests sur chaque scénario, d'une durée de 45", avec respectivement 60 et 150 agents.

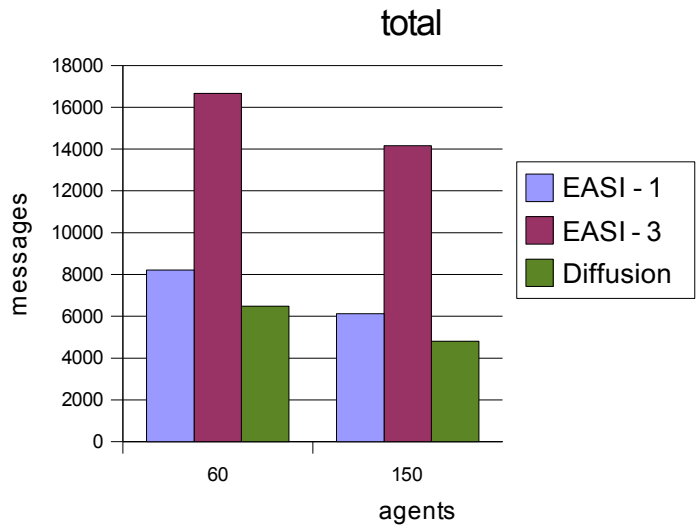


FIGURE 7.2 – Résultats comparatifs du modèle de diffusion, du modèle EASI (un environnement), et du modèle EASI (trois environnements), en fonction du nombre d'agents.

Nous pouvons observer que les proportions sont stables lorsque le nombre d'agents augmente. Nous expliquons la baisse du nombre de messages comptés malgré l'augmentation du nombre d'agents par deux facteurs :

- Nous avons vu qu'un message transmis à n agents est compté une seule fois. Le nombre de messages comptés n'est donc pas affecté par la hausse du nombre d'agents.
- Chaque agent consomme des ressources pour ses calculs internes. L'augmentation du nombre d'agents entraîne donc une hausse de la part du temps processeur dédié à ces calculs.

Ensuite, nous étudions l'impact d'une variation du temps d'exécution. Les résultats donnés en figures 7.3, 7.4, 7.5, 7.6 sont les moyennes des tests sur 60 agents, chaque test ayant été réalisé 15 fois. La durée des tests varie entre 23 secondes et 3 minutes.

Dans le premier scénario (figure 7.3), où la discrimination dépend de la localisation et de la disponibilité des agents, EASI-1 est meilleur que la diffusion. La distance entre les deux courbes est significative, puisque près de 20% de messages de plus sont échangés avec 1 environnement. Nous observons également avec EASI-3 un gain de 130% du nombre de messages par rapport à EASI-1. La simplification du filtre $f^{+}_{request}$ a donc un impact important sur l'efficacité de l'environnement.

Dans le second scénario (figure 7.4), les résultats sont plus ambivalents : EASI-1 est très proche de la diffusion, et EASI-3 montre un nombre plus faible de messages. Par rapport au premier scénario, l'abaissement de l'écart entre EASI-1 et la diffusion s'explique par le nombre

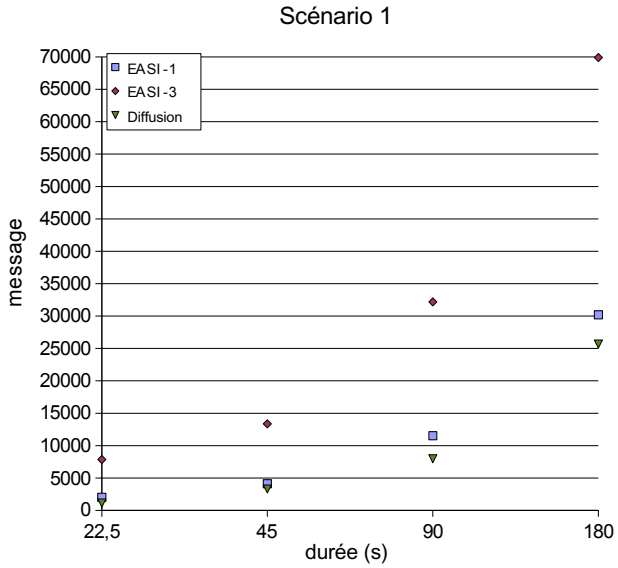


FIGURE 7.3 – Résultats d'exécution du premier scénario : comparatif du modèle de diffusion, du modèle EASI (un environnement), et du modèle EASI (trois environnements) en fonction de la durée.

plus important d'agents contactés par le "propose" que par le "request". Autrement dit, plus la classe d'agents trouvés par le filtre est grande, moins le modèle EASI est avantageux par rapport au modèle de diffusion. Quant à la faiblesse d'EASI-3, elle est dûe aux changements fréquents d'environnements qui ont lieu lors des protocoles, les coûts d'inscription et retraits étant importants. Nous observons donc que les coûts contrebalancent la simplification du filtre $f^{+}_{propose}$.

Dans le troisième scénario (figure 7.5), où la discrimination a lieu sur l'identifiant, et où elle est donc la plus forte, nous observons un avantage net d'EASI-1 sur la diffusion (62% de messages supplémentaires), et EASI-3 est lui-même meilleur qu'EASI-1 de 58%.

Le dernier graphique (figure 7.6), qui est une agrégation des trois précédents, montre l'avantage moyen d'EASI-3 par rapport aux deux autres solutions, et un léger avantage d'EASI-1 par rapport à la diffusion. Ce résultat montre que le taux de discrimination des messages, c'est à dire la proportion d'agents recevant un message et étant intéressé par lui, est la condition principale qui détermine l'avantage du modèle EASI par rapport au modèle de diffusion. Nous notons par ailleurs que plus il y a de messages, plus EASI est intéressant.

Les principales différences entre EASI-1 et EASI-3 sont le nombre de descriptions dans les environnements et le nombre de tests réalisés dans les filtres. Ce nombre de tests correspond au nombre d'appariements à chaque niveau de l'arbre. Les résultats montrent clairement qu'une diminution du nombre d'appariements a un très fort impact positif sur l'efficacité du système (résultats du scénario 1), et que la diminution du nombre de descriptions a un impact positif

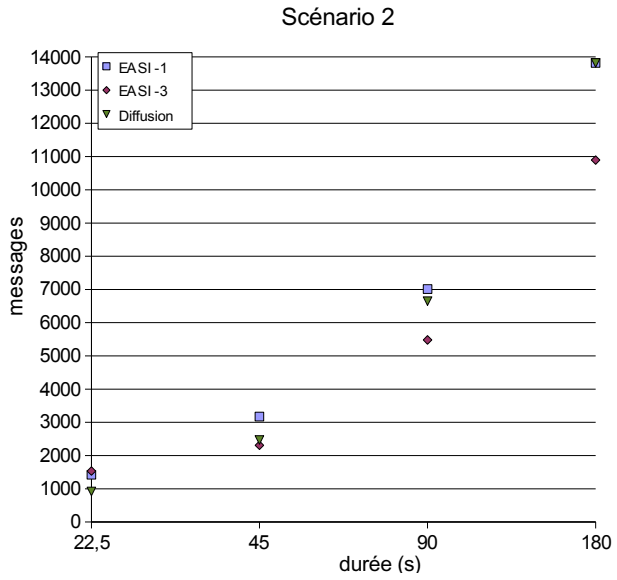


FIGURE 7.4 – Résultats d'exécution du deuxième scénario : comparatif du modèle de diffusion, du modèle EASI (un environnement), et du modèle EASI (trois environnements) en fonction de la durée.

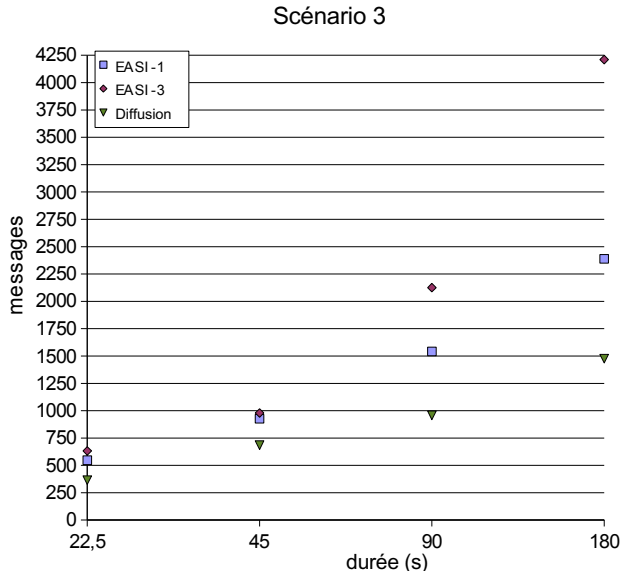


FIGURE 7.5 – Résultats d'exécution du troisième scénario : comparatif du modèle de diffusion, du modèle EASI (un environnement), et du modèle EASI (trois environnements) en fonction de la durée.

notable (résultats du scénario 3). A l'inverse, l'ajout et le retrait de faits provoqués par les entrées et sorties des agents entre les environnements ont un impact négatif non négligeable (résultats du scénario 2).

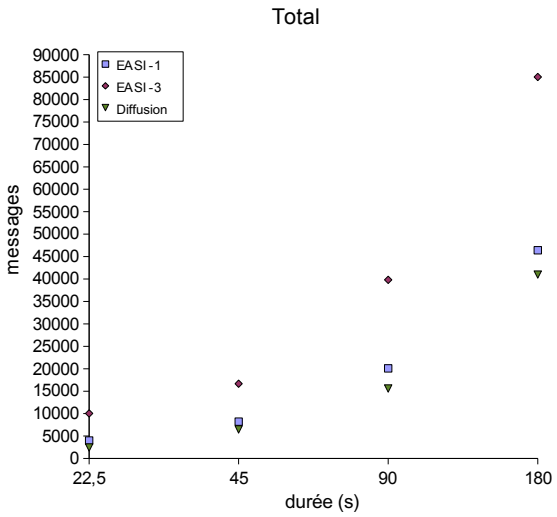


FIGURE 7.6 – Résultats d'exécution totaux : comparatif du modèle de diffusion, du modèle EASI (un environnement), et du modèle EASI (trois environnements), en fonction de la durée.

Cette série d'expériences montre que l'utilisation d'un système-expert est une solution plus efficace que la diffusion pour la transmission des informations si, en moyenne, le taux de discrimination des messages est supérieur à $\frac{1}{2}$.

Par contre, les tests ont montré que la différence du nombre de messages transmis entre l'algorithme RETE et la diffusion est relativement faible. Cela est dû au coût de construction de l'arbre : chaque fois qu'une entité est ajoutée ou retirée, l'arbre est modifié en conséquence. L'ajout d'un message, même s'il est éphémère, entraîne donc un coût élevé puisqu'il est retiré dès la fin de son traitement. C'est pourquoi nous proposons d'étudier l'efficacité des algorithmes proposés en section 4.4.2, qui tirent profit des particularités d'EASI.

7.2 Implémentation des modèles EASI et EARI à l'aide d'algorithmes adaptés

Nous avons vu que l'implémentation de l'environnement par un système-expert a deux avantages principaux, qui sont d'une part une applicabilité aisée et d'autre part une meilleure efficacité que la diffusion. Cependant, les tests réalisés montre que ce gain est relatif. En pratique, il sera limité assez rapidement par le nombre d'agents et messages transmis par le système. En section 4.4 nous avons introduit deux algorithmes spécialisés, fondés sur la structure des descriptions et des filtres. Nous évaluons ici les performances de ces algorithmes.

7.2.1 Evaluation du modèle EASI

Pour évaluer nos algorithmes, nous avons implémenté les tests dans le cadre d'une Application d'Intelligence Ambiante [Saunier et Balbo, 2007a]. Cette application tire partie des communications multi-parties.

7.2.1.a Cadre des expérimentations : l'Intelligence Ambiante

Nous développons ici un exemple de service de communication dédié à une Application d'Intelligence Ambiante (AIA). Cette application doit faciliter les interactions entre les employés d'une entreprise et leurs visiteurs. Un agent *employé* appartient à un service et a une disponibilité. L'entreprise est composée de salles de service, de salles de réunion et d'une réception. L'objectif est de proposer une application permettant le support des différents besoins d'interaction de façon standardisée. Pour un visiteur, les besoins d'interaction directe seront liés à la recherche d'un certain employé (situation notée $S_{direct1}$), ou à la recherche d'un employé disponible dans un service déterminé ($S_{direct2}$). Un exemple d'interaction indirecte est lié à la libération d'une salle. Cet évènement devra être perçu par les agents *employé* intéressés ($S_{indirect}$). Enfin, l'application doit supporter des modèles d'interaction plus complexes comme l'écoute flottante. Par exemple, un agent peut surveiller l'activité du système multi-agents en écoutant les employés en présence de clients dans les salles d'un service particulier ($S_{monitor}$).

La Figure 7.7 décrit un exemple de notre modélisation pour l'AIA. Il y a quatre entités, $\Omega = \{\omega_1, \omega_2, \omega_3, \omega_4\}$ qui sont respectivement l'agent *visiteur* v_1, les agents *employé* e_1 et e_2 et le message m_1. Les agents *employés* ont pour propriétés :

- Identifiant (*id*) : l'agent possède un identifiant unique
- Disponibilité (*dispo*) : l'agent est disponible (*true*) ou non (*false*)
- Service (*service*) : l'agent appartient à un service de l'entreprise dans l'ensemble {*R&D, marketing, SAV, support, développement*}.
- Une localisation (*lieu*) : l'agent est situé à un endroit, le domaine de description de cette propriété D_{lieu} est l'ensemble des salles du bâtiment.

Pour un agent *visiteur* a, la Pdescription $P_a = \{lieu, id, visitSujet\}$ est l'ensemble des propriétés des agents *visiteur*.

Les définitions des filtres correspondants aux situations décrites sont les suivantes.

$f^+_{direct1}$ est le filtre pour les interactions dyadiques, qui met donc en rapport la propriété *id* de l'agent avec la propriété *receiver* du message :
$f_{direct1}(a, m, C) = \langle [id(a) = ?x], [receiver(m) = ?x], \emptyset, \text{“}direct\text{”}, 0, environnement \rangle$

La situation $S_{direct2}$ recouvre la recherche d'un employé disponible dans un service déterminé :
$f^+_{direct2}(a, m, C) = \langle [service(a) = ?x] \wedge [dispo(a) = true], [service(m) = ?x], \emptyset, \text{“}direct2\text{”}, 0, a_x \rangle$
Il s'agit d'un appariement entre le service recherché, inséré sous forme de propriété dans le message, et le service de l'agent récepteur, qui doit être disponible.

Le filtre d'interaction indirecte proposé pour percevoir les libérations de salles est :
$f_{indirect}(a, m, C) = \langle [id(a) = \text{“}ax\text{”}], [lieu(m) \in MR] \wedge [sujet(m) = \text{“}disponible\text{”}], \emptyset, \text{“}indirect\text{”}, 0,$

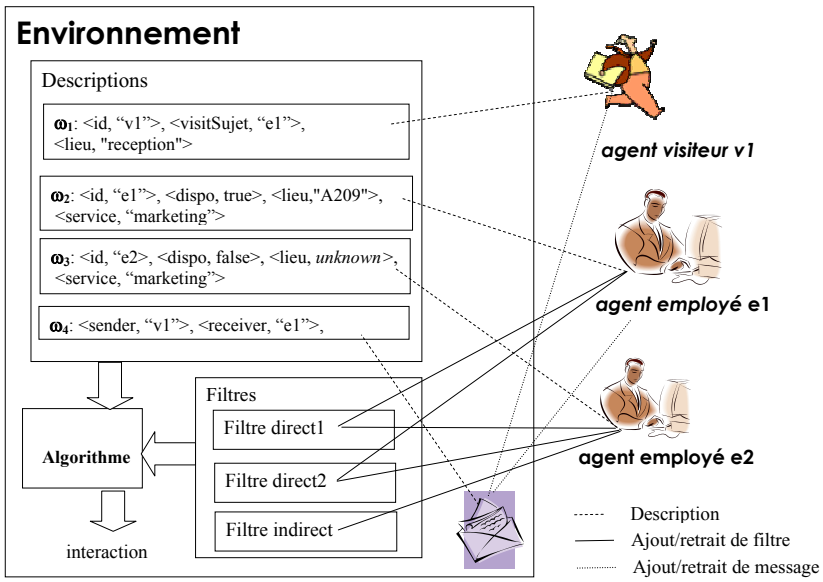


FIGURE 7.7 – Exemple de modélisation pour l'intelligence ambiante

$a_x\rangle$

L'agent avec la propriété id à valeur "ax" reçoit tous les messages liés à la disponibilité $[sujet(m) =$ "$disponible$"$]$ des salles de réunion $[lieu(m) \in MR]$.

Enfin, le filtre pour la surveillance a pour définition :
$f_{surveillance}(a, m, \{a1, a2\}) = \langle [id(a) =$ "ax"$], [sender(m) =?y], [lieu(a1) \in MR] \wedge [lieu(a1) = ?x] \wedge [lieu(a2) =?x] \wedge [service(a1) =$ "$R\&D$"$] \wedge [id(a1) =?y],$ "$surveillance$"$, 0, a_x\rangle$
L'agent a_x écoute les messages émis par les agents du service "R&D" (entité $a1$) qui se trouvent dans la même salle qu'un agent visiteur (entité $a2$).

7.2.1.b Paramètres de la simulation

De façon à évaluer la performance de nos algorithmes, nous avons mis en place une série de tests comparatifs comprenant la diffusion classique et nos deux algorithmes, respectivement notés EASI-algo1 (section 4.4.2) et EASI-algo2 (section 4.4.3). Nous nous sommes intéressés en particulier à la dépendance entre le taux de mise à jour, le nombre d'agents et la performance du système. Les tests sont des simulations de l'application décrite précédemment.

En utilisant le modèle de diffusion, la difficulté est, pour l'agent qui reçoit un message, de savoir s'il est un récepteur au sens des communications multi-parties. Pour cela, il doit connaître les besoins de tous les autres agents, ainsi que les descriptions des autres entités du système. Il est donc nécessaire de diffuser les mises à jour de propriétés de toutes les entités. En effet, certains filtres nécessitent une connaissance du contexte pour choisir ses récepteurs, par exemple $f_{surveillance}$. Un agent recevant un message ne peut calculer ce contexte que si les propriétés des entités lui ont été préalablement envoyées.

Cependant, si tous les agents connaissent les besoins et descriptions de tous les agents, il n'est plus nécessaire de diffuser les messages à tous les agents : l'émetteur peut calculer localement les récepteurs de ses messages. Nous avons donc choisi cette solution, de façon à ce que les émetteurs puissent choisir les récepteurs en fonction des propriétés à jour sans élever le coût de la diffusion de façon artificielle. Nous considérons que l'émetteur connaît les besoins des récepteurs sans surcoût, ce qui signifie que le coût de la diffusion sera légèrement sous-évalué [26].

De cette façon, nous pouvons utiliser exactement les mêmes critères que pour les algorithmes EASI, et vérifier qu'il y a une parfaite équivalence [27] entre les exécutions des trois supports de communication. Dans cette série de tests, nous calculons le temps d'exécution de la simulation pour un nombre de pas fixés.

Par diffusion, la résolution du problème de connexion est décentralisée, *i.e.* chaque agent calcule les récepteurs de ses messages, tandis que le modèle EASI centralise ce calcul au niveau de l'environnement. Pour pouvoir comparer ces deux approches, nous devons donc évaluer la performance du système dans son ensemble. Ainsi, à la fois le processus de décision des agents et la gestion de l'environnement sont mesurés dans un simulateur centralisé. Il est à noter que nous ne mesurons donc pas les coûts en bande passante des différentes solutions.

La simulation procède de façon classique : à chaque pas de temps et dans un ordre aléatoire, chaque agent vérifie ses messages, puis choisit et exécute un comportement, comme répondre à un message, ajouter un filtre, etc. La moitié des agents sont des agents *employé*, l'autre moitié des agents *visiteur*. Chaque agent met à jour sa propriété *dispo* lorsque c'est nécessaire, et les agents *employé* modifient leur propriété *service* en fonction du taux de mise à jour. De façon à recréer les conditions incitant les agents à avoir besoin de recevoir des messages indirects, le nombre de salles est de 20% inférieur aux besoins.

7.2.1.c Résultats

Les constantes de base sont un taux de mise à jour de 1/10 (une fois tous les 10 pas), pour 40 agents, sur 8000 pas. Chaque résultat est une moyenne de 100 exécutions. Le premier graphique (Fig. 7.8, haut) donne le temps d'exécution de la simulation en fonction du nombre d'agents. Nous observons que la diffusion est le modèle le moins efficace quel que soit le nombre d'agents. C'est la limite classique de la diffusion, qui la rend inutilisable si le nombre de messages et/ou d'agents devient important. Nous avons pu vérifier que nos algorithmes pouvaient gérer un nombre assez important d'agents avec les deux algorithmes EASI : nous avons exécuté des tests jusqu'à 1000 agents en 9 minutes, ce qui représente 22 millions de messages.

Pour moins de 30 agents, le temps d'exécution d'EASI-algo2 est plus grand que celui d'EASI-algo1, tandis que pour plus d'agents c'est le contraire. Ce résultat est conforme à nos prévisions

26. L'ajout d'un filtre est une souscription à l'environnement. Son équivalent est, dans le cadre de la diffusion, l'envoi d'un message de mise à jour des préférences à tous les agents.

27. L'équivalence est vraie pour les messages et les comportements, excepté les messages de mise à jour qui n'ont lieu que pour la diffusion. Chaque simulation est lancée trois fois avec la même graine aléatoire, une fois pour chaque support.

de la section 4.4.2 : la création et la gestion des ensembles utilisés par EASI-algo2 nécessitent plus de calculs que pour les ensembles utilisés par EASI-algo1, tandis que l'appariement d'un message est plus rapide pour EASI-algo2. Pour un petit nombre d'agents, le surcoût du calcul des ensembles n'est pas rentabilisé par le gain en vitesse d'appariement. Par contre, plus le nombre d'agents augmente, plus l'avantage d'EASI-algo2 par rapport à EASI-algo1 croît.

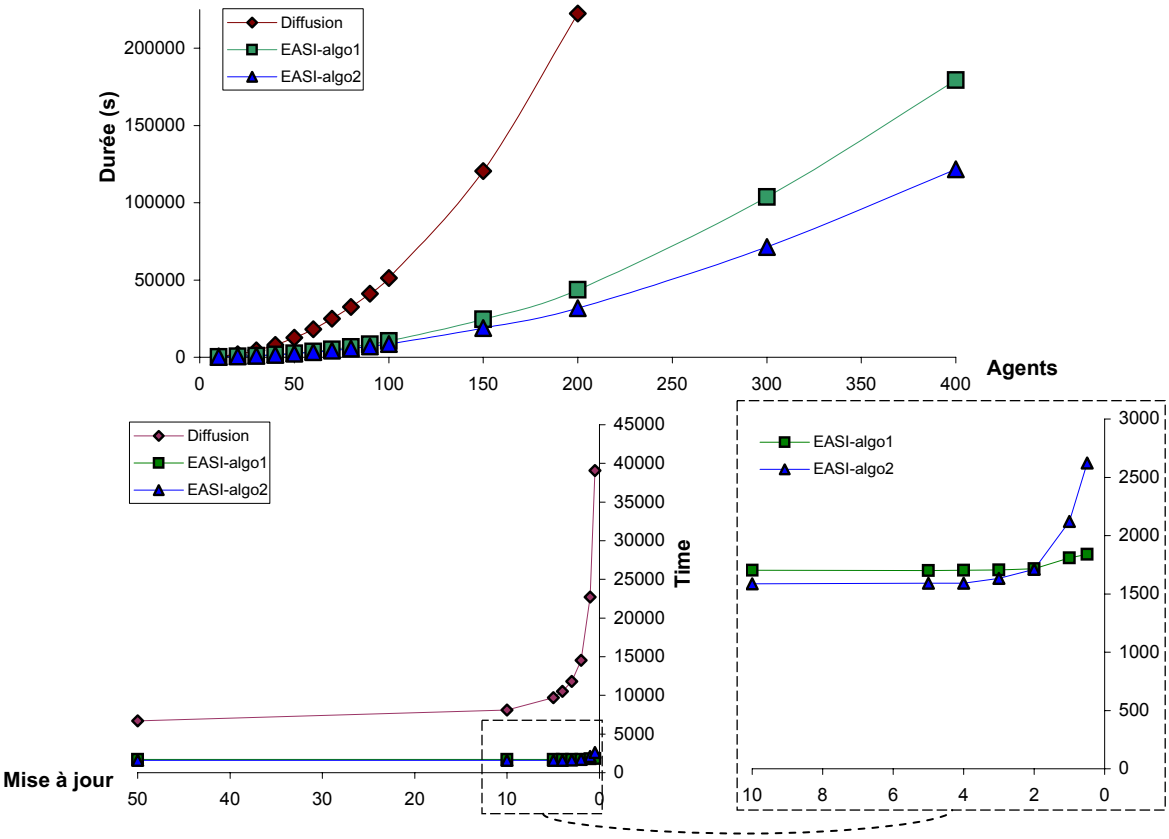


FIGURE 7.8 – Diffusion et Algorithmes EASI. Temps d'exécution en fonction du nombre d'agent (haut) et du taux de mise à jour (bas).

Le second graphique (Fig. 7.8, bas et droite) montre le temps d'exécution en fonction du taux de mise à jour. A nouveau, nos algorithmes sont plus efficaces que la diffusion. Puisqu'EASI-algo1 utilise des ensembles calculés à partir de la classification structurelle des entités, le taux de mise à jour n'a pas d'effet sur l'algorithme lui-même. La légère hausse du temps de calcul observée à partir d'un taux de mise à jour de 1/4 est due aux calculs internes aux agents.

Finalement, le temps d'exécution d'EASI-algo2 est sensible à une forte dynamique des propriétés : lorsque la fréquence de mise à jour est supérieure à 1/2 il devient moins efficace qu'EASI-algo1 à cause du coût de mise à jour des ensembles. Autrement dit, l'algorithme EASI-algo2 est rentabilisé à partir du moment où, en moyenne, deux messages sont envoyés dans l'intervalle de temps entre les mises à jour des propriétés.

Ces tests montrent que, par rapport à la diffusion, le modèle EASI et nos algorithmes sont

une solution efficace au problème de connexion. Le choix entre EASI-algo1 et EASI-algo2 doit être fait en fonction de la taille et de la dynamicité du système multi-agents : EASI-algo1 est le plus efficace si il y a peu d'agents, ou si le taux de mise à jour des propriétés est très important (si les agents mettent plus souvent à jour leurs propriétés qu'ils n'envoient de messages). Dans les autres cas, EASI-algo2 est la meilleure solution.

7.2.2 Evaluation du modèle EARI

Nous évaluons ensuite le coût d'utilisation de filtres de contrôle. Cette fois, les tests comparent la diffusion classique, l'environnement sans contrôle et l'environnement avec contrôle. Nous avons choisi de mettre en oeuvre les algorithmes fondés sur l'organisation statique (EASI-algo1). Le cadre des tests est à nouveau l'application dans le cadre l'Intelligence Ambiante.

7.2.2.a Description de l'application

Nous avons vu qu'il n'est pas possible d'assurer la régulation du système multi-agents par le biais de la diffusion (chapitre 2). Ainsi, les simulations de test d'EASI et de la diffusion ne sont pas régulées, tandis que celles d'EARI mettent en oeuvre des filtres négatifs.

Pour cette série de tests, nous avons ajouté un nouveau scénario de communication multi-parties. Il s'agit de la politique de l'entreprise lorsque les visiteurs et les employés parviennent à un accord (figure 7.9). Le message signifiant cet accord est envoyé par l'agent employé ("$e5$" dans la figure) au visiteur ("$v4$" dans la figure), grâce au filtre $f^{+}_{direct1}$, le visiteur est donc un destinataire. Les employés de plus haut rang désirent également être informés des accords, l'employé émettant l'accord ajoute pour cela le filtre suivant :
$f^{+}_{inform}(a, m, C) = \langle [rang(a) > 3] \wedge [service(a) = \text{“}R\&D\text{”}], [sujet(m) = \text{“}accord\text{”}] \wedge [sender(m) = \text{“}e5\text{”}], \emptyset, \text{“}inform\text{”}, 0, e5 \rangle$

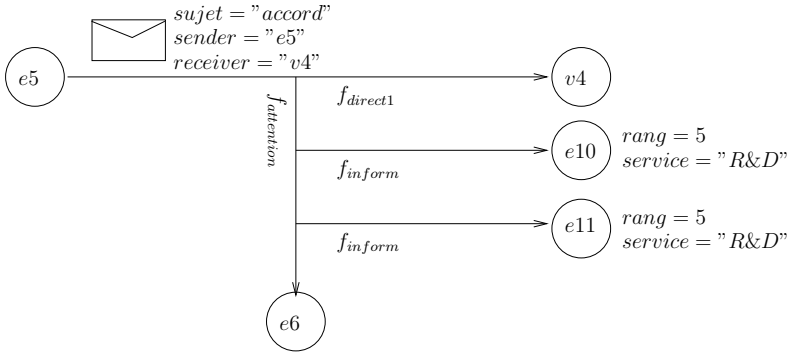


FIGURE 7.9 – Exemple de communication multi-parties

Le message émis par "$e5$" au visiteur sera donc entendu par les employés du département ayant un rang strictement supérieur à 3, qui sont donc dans ce cas des auditeurs : l'initiative est celle de l'émetteur, et les récepteurs sont attendus, mais ils ne prennent pas de part active dans le dialogue.

Enfin, s'ils sont libres, les autres employés du département peuvent souhaiter être informés des accords passés, un employé "e6" ajoutera dans ce cas le filtre :
$f^{+}_{attention}(a, m, \{ax\}) = \langle [id(a) = \text{“}e6\text{”}], [sujet(m) = \text{“}accord\text{”}] \wedge [sender(m) = ?x], [id(ax) = ?x] \wedge [service(ax) = \text{“}R\&D\text{”}], \text{“}attention\text{”}, 0, e6 \rangle$

Ainsi, chaque accord mené dans le département "$R\&D$" sera écouté par "$e6$", qui est un écouteur. Dès qu'il est occupé ou ne souhaite plus recevoir ces messages, l'employé retire son filtre.

Grâce à ces filtres, la communication multi-parties est mise en oeuvre, puisque la prise en compte des initiatives de l'émetteur et des récepteurs permet à plusieurs rôles (destinataire, groupe de réception et écouteur) de coexister pour un même message.

Nous incluons également les règles de l'environnement. En particulier, nous souhaitons assurer la possibilité de transmettre des messages privés, dans le sens où ceux-ci ne peuvent pas être écoutés par d'autres agent ($S_{private}$). De plus, les visiteurs ne doivent pas pouvoir écouter les messages des employés ($S_{visiteur}$), et les employés ne peuvent pas écouter les messages des employés de plus haut rang dans la hiérarchie de l'entreprise ($S_{hierarchie}$). Nous prenons comme cadre la politique de prédominance de l'environnement, où les filtres dont l'environnement est initiateur ont une priorité plus grande que les filtres des agents (les filtres de l'environnement ont une priorité strictement positive).

Le support des messages privés est réalisé de la même façon que dans l'exemple de la cité digitale, à l'aide des deux filtres $f^{+}_{direct1}$, qui assure la transmission du message au destinataire, et $f^{-}_{private}$, qui empêche la réception par d'autres agents. En accord avec la politique de prédominance de l'environnement, nous donnons au filtre $f^{+}_{direct1}$ une priorité de 10. Nous rappelons ici la syntaxe du filtre $f^{-}_{private}$:
$f^{-}_{private}(a, m, C) = \langle [id(a) = ?x], [receiver(m) \neq ?x] \wedge [private(m) = true], \emptyset, \text{“}private\text{”}, 10, environnement \rangle$

Concernant la situation $S_{hierarchie}$, en considérant une propriété $rank$ qui donne le rang de l'employé, le filtre sera exprimé de la façon suivante :
$f^{-}_{hierarchie}(a, m, \{ax\}) = \langle [rang(a) < ?r], [sender(m) = ?x], [id(ax) = ?x] \wedge [rang(ax) = ?r], \text{“}hierarchie\text{”}, 5, environnement \rangle$

L'agent a ne peut pas recevoir de messages des agents dont le rang est supérieur au sien. Nous avons choisi de donner au filtre une priorité de 5, de façon à ce que les messages adressés directement à ces agents (par le filtre $f^{+}_{direct1}$ de priorité 10) lui soient tout de même transmis.

Enfin, le filtre pour la situation $S_{visiteur}$ doit prendre en compte le type d'agent, *visiteur* ou *employé*. Cela peut être obtenu par l'étude de la Pdescription de chacun, les agents *visiteur* ayant seuls une propriété *visitSujet* et les agents employés ayant seuls une propriété *service*. Autrement dit, les agents dont la propriété *visitSujet* a pour valeur *unknown* ou une valeur appartenant au domaine de définition $D_{visitSujet}$ sont nécessairement des visiteurs. Le filtre obtenu est donc :
$f^{-}_{visiteur}(a, m, \{ax\} = \langle [visitSujet(a) \in D_{visitSujet} \cup \{unknown\}], [sender(m) = ?x], [id(ax) = ?x] \wedge [service(ax) \in D_{service} \cup \{unknown\}], \text{“}visiteur\text{”}, 10, environnement \rangle$

Les filtres utilisés pour comparer le modèle de diffusion et le modèle EASI seront donc $f^+_{direct1}$, $f^+_{direct2}$, $f^+_{indirect}$ et $f^+_{surveillance}$. Par rapport aux simulations précédentes ont aussi été ajoutées les communications liées aux accords entre employés et visiteurs, avec les filtres f^+_{inform} et $f^+_{attention}$. Enfin, les filtres $f^-_{private}$, $f^-_{hierarchie}$ et $f^-_{visiteur}$ sont implémentés pour le modèle EARI.

7.2.2.b Résultats

Les tests sont effectués avec un taux de mise à jour de 1/5, pour 50 agents, sur 6000 pas. Chaque résultat est une moyenne de 100 exécutions. Le premier graphique (figure 7.10, gauche) donne le temps d'exécution en fonction du nombre d'agents. La diffusion est la moins efficace, comme lors des simulations précédentes. Cela s'explique par le fait que seul l'environnement doit être averti des modifications de valeur des propriétés.

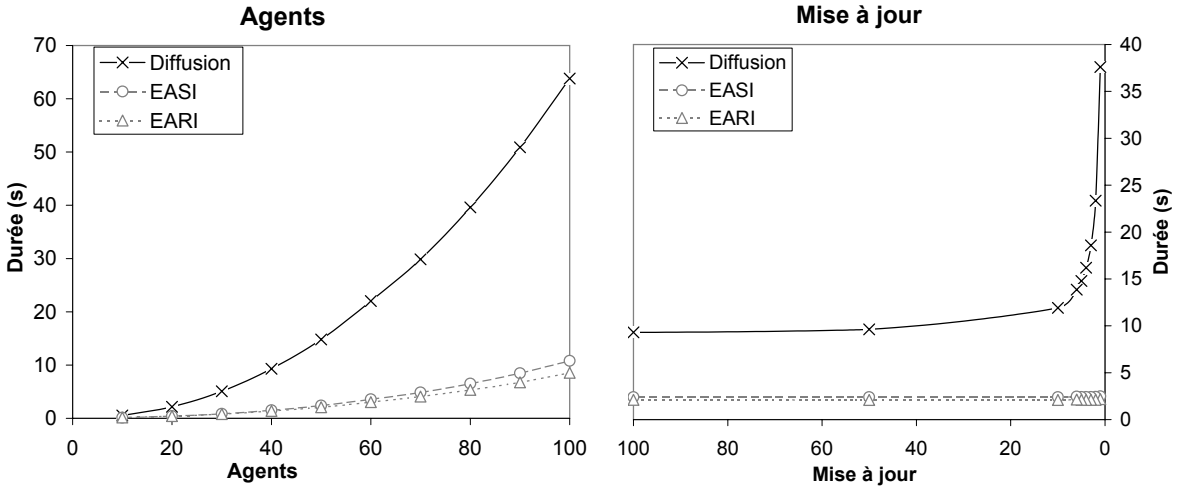


FIGURE 7.10 – Temps d'exécution en fonction du nombre d'agents (gauche) et de la fréquence de mise à jour (droite)

Le second graphique (figure 7.10, droite) montre le temps d'exécution de la simulation en fonction du taux de mise à jour. Sur ce graphique, nous constatons que le modèle EASI est toujours plus efficace que la diffusion. Ce résultat est accentué par le fait que les ensembles utilisés par l'algorithme sont basés sur les ensembles statiques qui ne sont pas sensibles aux mises à jour.

Enfin, les courbes pour le modèle EARI montrent que le coût additionnel des filtres est atténué par le court-circuit de l'algorithme : le calcul est arrêté pour un couple message/agent dès qu'un filtre négatif est valide. Ainsi, bien que le nombre de filtres soit plus grand que pour les simulations du modèle EASI, on observe une légère baisse du temps d'exécution.

Nous avons également testé le cas où des agents malveillants ajoutent des filtres f^-_{sourd} pour empêcher les autres agents de recevoir des messages [28]. Nous rappelons que cela n'affecte pas le

28. Rappel : $f^-_{sourd}(a, m, C) = \langle [id(a) = \text{“}e5\text{”}], \emptyset, \emptyset, \text{“}sourd\text{”}, 0, a_1 \rangle$

résultat de l'appariement grâce à la politique de prédominance de l'environnement et à celle de prédominance des filtres personnels. Du point de vue de l'efficacité, les résultats montrent que le court-circuit de l'algorithme permet une nouvelle fois d'être efficace quel que soit le nombre de filtres posés par les agents malveillants, puisqu'au pire un seul de ces filtres négatifs valide est calculé.

Ces deux résultats montrent que, si la baisse du coût induit par le contrôle est dépendant des filtres choisis, cette observation est généralisable grâce aux propriétés de l'algorithme. Un filtre négatif ne génère de surcoût que si le gain du court-circuit est contrebalancé par le coût de l'appariement. Hormis pour les filtres qui ne sélectionnent jamais d'agent, et sont donc inutiles, le coût d'ajout d'un filtre négatif est faible.

Cette série de tests montre donc que le modèle EARI est aussi efficace que le modèle EASI, malgré l'ajout de la régulation.

7.3 Exploitation des modèles EASI et EARI

Dans cette section, nous étudions deux façons d'exploiter notre support des communications multi-parties et leur régulation.

Dans la première partie, nous étudions la mise en place de protocoles opportunistes grâce aux communications multi-parties. Dans la seconde partie, nous montrons comment mettre en oeuvre un système normé.

7.3.1 Définition de protocoles opportunistes

Nous avons vu dans l'état de l'art que l'écoute flottante permet la mise en place de comportements opportunistes [Kaminka et al., 2002; Legras et Tessier, 2004; Platon et al., 2005]. Cependant, ces travaux ne proposent pas d'outils de conception *a priori* de ces interactions. Pour pallier cette limite, nous proposons dans cette section un nouveau connecteur d'interaction contextuelle.

7.3.1.a Un connecteur d'interaction contextuelle

Le cadre que nous avons choisi est celui de la formalisation Agent UML [Odell et al., 2000], qui est le résultat d'un effort de standardisation de la modélisation multi-agents. Plus précisément, nous nous situons au niveau de la spécification des protocoles par les diagrammes d'interaction afin de modéliser l'écoute flottante. Le choix d'un nouveau support d'interaction entraîne la nécessité de nouveaux connecteurs AUML. Les connecteurs AUML classiques sont fondés sur l'adressage par l'émetteur et ne permettent donc pas de prendre en compte les cas où un message est reçu de par la volonté du récepteur ou de l'environnement. Ces travaux ont été publiés dans [Balbo et al., 2004].

Ce filtre négatif concerne l'agent $e5$, il est valide pour tous les messages.

Nous nous plaçons donc dans le cadre où un agent choisit de poser un filtre afin d'écouter certains messages qui ne lui sont pas adressés. Dans le cas où un message correspond au filtre, il reçoit le message, mais n'a pas de rôle pré-attribué, ne faisant pas partie des interlocuteurs prévus.

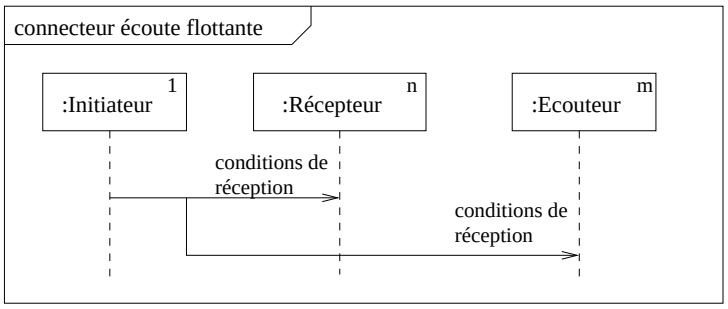


FIGURE 7.11 – Connecteur AUML d'écoute flottante

Le connecteur AUML proposé est celui montré en figure 7.11. Le message adressé d'origine reste le même, pour n récepteurs cibles, représenté par la flèche horizontale, tandis que l'écoute flottante est représentée par les m agents écouteurs, qui reçoivent également le message mais de façon différenciée par la flèche décrochée. Il est possible de poser le filtre ayant présidé à la réception du message au-dessus de la flèche décrochée. Les possibilités de changement de rôle et les rôles multiples d'AUML sont conservés, par exemple un agent peut être écouteur et devenir participant.

Nous pouvons alors rediscuter certains protocoles existants en introduisant des agents utilisant l'écoute flottante, et en utilisant donc de façon explicite ce nouveau connecteur.

7.3.1.b Composition de protocoles

Dans l'exemple illustratif (Fig. 7.12), nous comparons deux protocoles de requête, le second étant un dérivé du premier rendu possible par l'écoute flottante.

Par souci de clarté, nous avons simplifié le protocole **request**, en n'indiquant pas les cas d'erreur ou d'absence de réponse. Le protocole consiste en l'envoi d'une requête *request*, de tâche par exemple, par l'agent initiateur, et d'une réponse *accept* ou *refuse* de la part du participant concerné.

Le protocole **ef request** (ef comme écoute flottante) est une extension du premier. Il signifie qu'en plus du participant initialement prévu, un ou plusieurs agents écouteurs sont susceptibles de recevoir les messages *request* et leurs réponses, grâce aux conditions de réception c_i. Dans le cas où le participant refuse la requête, cas reconnu grâce à la réception du message réponse, les agents écouteurs ont l'opportunité d'entamer un protocole *propose* avec l'initiateur.

Dans le protocole classique, en cas d'échec du protocole (*refuse* ou *failure*), l'agent Initiateur doit contacter d'autres agents afin d'entamer de nouveaux protocoles *request* pour poursuivre ses buts, chaque instance de ces protocoles ayant un coût en terme de messages et de traitements.

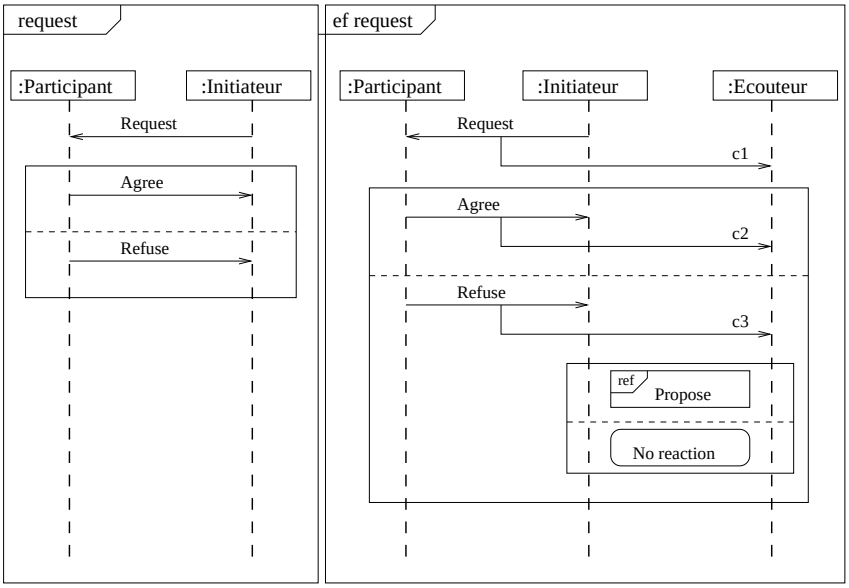


FIGURE 7.12 – Connecteur d'écoute flottante : Protocole **request** simplifié à gauche. Protocole **ef request** étendu par l'écoute flottante à droite.

Avec le protocole *ef request*, dans le cas où un agent écouteur peut satisfaire la requête, il lui est possible d'entamer spontanément un protocole *propose* afin de satisfaire la requête non aboutie.

C'est donc bien une action opportuniste que l'agent écouteur met en oeuvre, puisque sur la base d'un message qui ne lui était pas destiné, il engage un protocole permettant éventuellement une résolution rapide du problème, dans le sens d'une plus grande efficacité, ce qui n'était pas réalisable dans le cadre classique.

Par construction, il est fait en sorte d'éviter les cas de protocoles pouvant s'enchaîner de manière récursive, en finissant l'enchaînement de protocole $ef\ request \rightarrow propose$ sur un propose classique, cela afin d'éviter les risques d'inter-blocage, et/ou une absence de condition de terminaison.

Le second exemple (Fig. 7.13) montre une proposition de protocole *ef propose* améliorant le protocole *propose*. Suivant les mêmes principes que le protocole *ef request*, l'idée est de permettre aux écouteurs d'entamer proactivement un nouveau protocole en cas d'échec du premier protocole.

L'agent initiateur envoie un message *propose* à un participant. Le participant répond par un message *agree* ou *refuse*. Le protocole peut s'achever sur une erreur (*failure*).

Avec notre alternative, les messages du protocole peuvent être reçus par des agents écouteurs. Ces agents écouteurs peuvent, en fonction du déroulement du protocole, entamer un nouveau protocole *propose* avec le participant. Dans l'exemple AIA, un exemple concret de mise en oeuvre de ce protocole concerne les agents *employé*. Si un agent *employé* reçoit un message *refuse* de la part d'un agent *visiteur*, d'autres agents employés peuvent entamer un protocole *propose* pour soumettre au *visiteur* des contre-propositions.

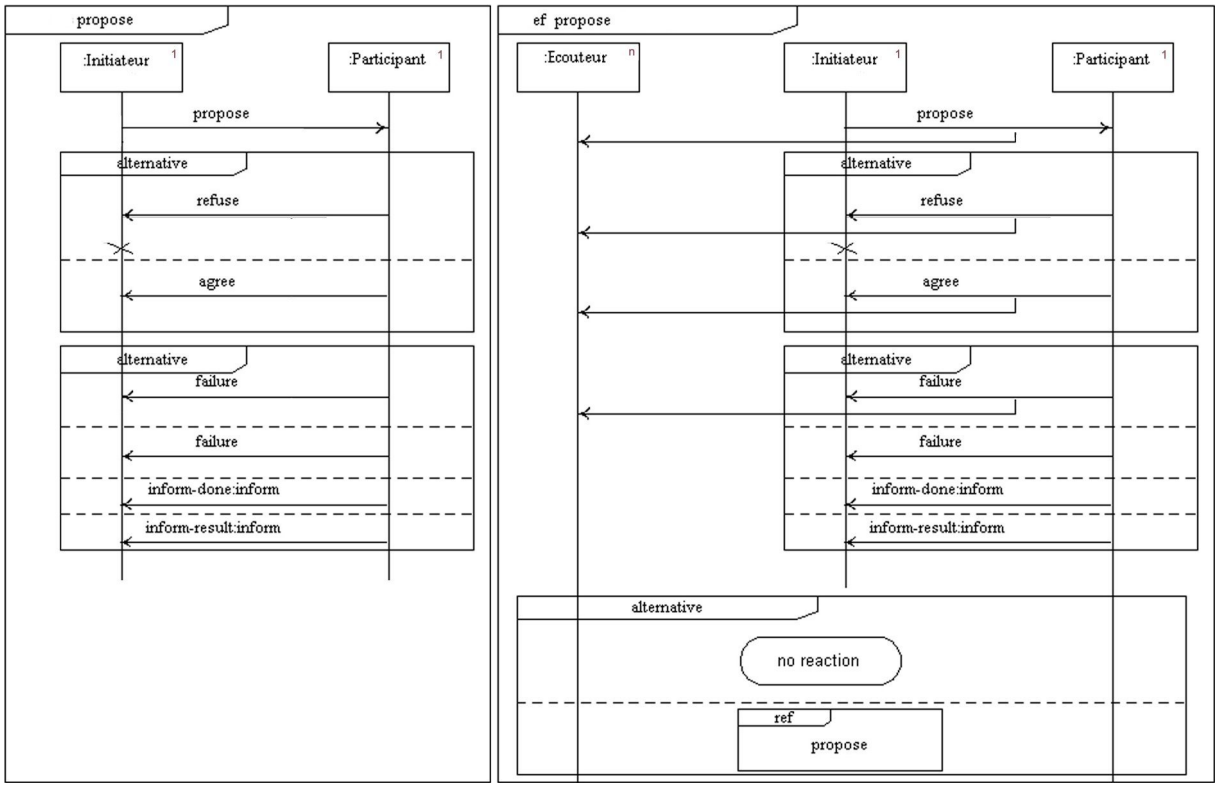


FIGURE 7.13 – Protocole **Propose** simplifié à gauche. Protocole **alternative Propose** étendu par l'écoute flottante à droite.

Ces deux exemples de protocoles montrent comment les agents écouteurs peuvent tirer profit du non-aboutissement d'un protocole pour cause de refus ou d'échec, pour mener à bien leurs objectifs. Nous avons choisi de permettre la mise en oeuvre proactive de protocoles pré-existants par les écouteurs afin de montrer comment ces comportements peuvent s'insérer dans des systèmes multi-agents classiques. De cette façon, les protocoles opportunistes peuvent co-exister dans un système multi-agents prévu pour utiliser les protocoles classiques.

7.3.2 Mise en oeuvre d'un système multi-agents normé fondé sur la logique déontique

Nous avons montré dans le chapitre 5 comment former des politiques de priorités permettant la mise en oeuvre de la régulation des communications. Nous étudions ici comment cette régulation permet de mettre en oeuvre des politiques de partage et de sécurité des informations. Une politique de partage d'information concerne la façon dont l'information est diffusée entre participants aux statuts différents. En particulier, une catégorie d'agents peut avoir l'obligation de transmettre certaines informations à d'autres agents, et l'interdiction de les transmettre à d'autres.

Nous prenons comme exemple le cas des plans de secours ORSEC, qui a été spécifié formellement dans [Cholvy et al., 2007]. Dans cet article, les auteurs proposent une modélisation formelle de politiques de partage d'information par une logique déontique. Elle permet la vérification des propriétés de cohérence et de complétude des politiques. Nous montrons ici comment le modèle EARI permet la mise en oeuvre de cette modélisation fondée sur la logique déontique.

Les concepts utilisés pour la modélisation du plan ORSEC sont les organisations et le contexte global. Plusieurs organisations sont appelées à interagir (police, pompiers, SAMU, *etc.*). Dans chacune de ces organisations les agents tiennent des rôles. Le contexte dépend de l'environnement de la crise et de son évolution, il est considéré comme exogène.

La logique déontique est ici fondée sur SDL (*Standard Deontic Logic*) [Chellas, 1980] : la permission et l'interdiction sont exprimées en fonction de l'obligation : faire l'action A est permis si et seulement si ne pas faire A n'est pas obligatoire et faire l'action A est interdit si et seulement si ne pas faire A est obligatoire.

Soit une politique de partage (R1,R2) qui exprime qu'en contexte de crise :

- (R1) tout agent x doit envoyer à tout agent y toute information concernant le thème "risque d'explosion" (noté ExpRisk) dès qu'il l'apprend :
  $(R1) \quad \forall x \forall i \forall y \forall t \forall t' \quad Crisis(t) \wedge Learn(x,i,t) \wedge Learn(x, topic(i, ExpRisk), t') \rightarrow Obligatory(send(x,i,y,max(t,t')))$
- (R2) il est interdit pour tout agent d'envoyer une information concernant le thème « risque bactériologique » (noté Bac) à quelqu'un qui ne joue pas un rôle officiel (modélisé par un rôle NonOff) :
  $(R2)\ \forall x \forall i \forall y \forall t \forall t' \forall t'' \ Crisis(t) \wedge Learn(x,i,t) \wedge (t'' > max(t,t')) \wedge Learn(x, topic(i, Bac), t') \wedge Playsrole(y, NonOff) \rightarrow Forbidden(send(x,i,y,t''))$

Cette situation est illustrée en figure 7.3.2. Le système est composé d'agents qui peuvent jouer un rôle officiel dans le plan ORSEC. En contexte de crise, les agents doivent transmettre toutes les informations qu'ils possèdent sur les risques d'explosion à tous les autres agents. Par contre, ils leur est interdit de fournir des informations sur les risques bactériologiques aux agents non-officiels.

Nous montrons alors comment cet exemple peut être représenté par le modèle EARI. L'environnement actif de communication EARI permet de réguler les actions de type *send* qui correspondent à l'envoi de message. Nous modélisons de façon directe les prédicats, par exemple *topic* et *Playsrole*, en tant que propriétés des agents. Les éléments exogènes, par exemple *Crisis* qui dénote une situation de crise, sont modélisés comme les propriétés d'un objet de l'environnement.

Nous devons modéliser trois situations : l'obligation, l'interdiction et la permission. Pour cela, nous nous situons dans le cadre de la prédominance de l'environnement :

$$\mathcal{R} \mapsto I(IP), \quad \mathcal{R}(f) = \left\{ \begin{array}{ll} [min_e, max_e] & \text{si } initiator = environnement \\ [min_a, max_a] & \text{si } initiator \in \mathcal{A} \end{array} \right. \Bigg| \text{ avec } max_a > max_e$$

- Les obligations sont des filtres positifs de l'environnement dont le comportement ne peut pas être court-circuité par les filtres des agents, ce qui est vrai avec la prédominance de

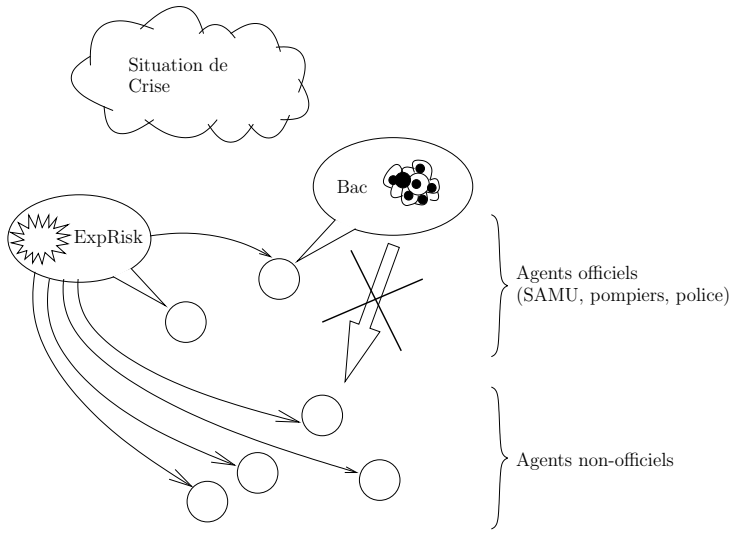


FIGURE 7.14 – Illustration d'une politique de partage d'information dans le cadre de la gestion de crise

l'environnement. Par exemple, la règle (R1) est modélisée par le filtre $f^+ \in \mathcal{F}^+$ suivant :
$\langle \emptyset, [topic(m) = \text{“}ExpRisk''], [Crisis(c) = true], environnement, \text{“}Obligation'', 100 \rangle$
Avec ce filtre, tous les agents reçoivent tous les messages dont le thème est un risque d'explosion en contexte *Crisis*.

– Les interdictions sont des filtres négatifs de l'environnement dont le comportement ne peut pas être court-circuité par les filtres des agents. Par exemple, la règle (R2) est modélisée par le filtre $f^- \in \mathcal{F}^-$ suivant :
$\langle [PlaysRole(a) = NonOff], [topic(m) = \text{“}Bac\text{”}], [Crisis(c) = true], environnement, \text{“}Interdiction\text{”}, 200 \rangle$
Ce filtre négatif empêche les agents non-officiels de recevoir les informations concernant les risques bactériologiques.

Cet exemple met en lumière les avantages de notre modèle EARI : grâce aux règles de l'environnement, il suffit qu'une information soit émise une fois pour que tous les agents devant la recevoir la reçoivent effectivement. De plus, cela assure au niveau multi-agents que les agents ayant interdiction de recevoir une information ne peuvent pas la recevoir, même en cas de message adressés explicitement.

Les axiomes régissant les trois modalités de la logique déontique que sont l'interdiction, la permission et l'obligation sont directement transcrits à l'aide de la formalisation du modèle EARI. L'obligation de faire quelque chose est modélisée par un filtre positif. La définition de la permission est :

Faire l'action A est permis si et seulement si ne pas faire A n'est pas obligatoire

L'obligation étant modélisée par un filtre, les cas où l'environnement n'a pas déposé de filtre pour un couple agent/message correspondent à l'absence d'obligation .

Faire l'action A est interdit si et seulement si ne pas faire A est obligatoire

Nous avons vu que l'obligation de faire quelque chose est un filtre positif, l'obligation de ne pas faire est donc un filtre négatif.

De cette façon, la modélisation EARI est un moyen efficace pour mettre en oeuvre des systèmes multi-agents dans lesquels les propriétés des communications ont été vérifiées au préalable par la logique déontique.

7.4 Conclusion

Nous avons montré dans ce chapitre que si les arbres RETE sont une façon effective d'implémenter le stockage et la dynamique de l'environnement, les gains sont décevants en comparaison de la diffusion. Dans un système multi-agents, les descriptions des agents peuvent être mises à jour souvent, ce qui génère des coûts importants de construction et de modification. Ces coûts contrebalancent en partie l'efficacité de l'appariement de l'algorithme.

Ensuite, nous avons montré que l'implémentation du modèle EASI par nos propres algorithmes apporte une amélioration importante du temps d'exécution par rapport à la diffusion. Nous avons mis en évidence les deux critères entrant en compte dans le choix de l'algorithme par le concepteur du système multi-agents : le nombre d'agents et la fréquence de mise à jour des propriétés par les agents. Si le nombre d'agents est faible et/ou si le taux de mise à jour des propriétés est élevé (les descriptions sont modifiées plus souvent que des messages ne sont envoyés), l'algorithme statique est le plus efficace ; dans les autres cas le choix doit se porter sur l'algorithme dynamique. Cela est dû à la nature même des informations stockées par ces algorithmes : l'algorithme statique ne conserve que les informations structurelles, et n'est donc pas sensible aux mises à jour des propriétés, tandis que l'algorithme dynamique est plus efficace lors du processus d'appariement.

Nos expérimentations ont également montré que le modèle EARI, qui étend le modèle EASI grâce à l'ajout du contrôle des communications, n'est pas plus coûteux que le modèle de base. Bien que le nombre de filtres dans l'environnement soit plus important, le raccourci opéré par les filtres négatifs valides sur l'algorithme contrebalance le coût de ces filtres.

Nous avons ensuite apporté des éléments sur la façon d'exploiter nos modèles. Nous avons proposé un nouveau connecteur AUML pour la conception de protocoles utilisant l'écoute flottante. Ces protocoles sont flexibles et permettent la mise en oeuvre de comportements opportunistes. Enfin, nous avons montré comment le modèle EARI permet de modéliser et mettre en oeuvre un système normé fondé sur la logique déontique.

Bilan et perspectives

La principale contribution de cette thèse est d'envisager les communications d'un point de vue global, sans distinguer *a priori* communications directes et indirectes. Ceci nous permet d'offrir une solution standardisée pour le support et le contrôle des communications au sein d'un même système multi-agents.

Au sein de l'équipe, nous avons proposé un principe, la coordination fondée sur les propriétés, pour la gestion des interactions dans les systèmes multi-agents.

Dans cette thèse, nous avons formalisé ce principe général dans le cadre d'un premier modèle, l'Environnement Actif comme Support de l'Interaction (EASI). Ce modèle utilise la *description* des composants du système multi-agents pour choisir les récepteurs des communications. Les besoins des agents en émission et en réception de messages sont modélisés par des *filtres*.

Nous avons ensuite proposé un second modèle, l'Environnement Actif comme Régulateur de l'Interaction (EARI), qui généralise le premier modèle en prenant en compte la régulation des communications du système multi-agents. Les *filtres négatifs* permettent d'interdire la réception d'un message, et les *politiques de priorités* permettent de modifier la prise en compte des filtres en fonction du type du filtre et de son initiateur.

Nos modèles EASI et EARI possèdent les propriétés nécessaires pour mettre en oeuvre les communications multi-parties, concevoir des protocoles opportunistes et concevoir des systèmes multi-agents normés. Pour une seule communication, les communications multi-parties permettent, de mettre en oeuvre les concepts liés aux *communications directes*, aux *communications indirectes* et à l'*écoute flottante*. Nous avons aussi proposé un connecteur AUML pour la *conception de protocoles* opportunistes, où les agents ont un *comportement proactif* grâce à la flexibilité des communications. Enfin, les systèmes normés permettent de *réguler les communications à l'exécution*.

Nous avons illustré les modèles EASI et EARI par la modélisation de deux applications : dans le cadre des cités digitales, nous avons modélisé un *portail de communauté* permettant de regrouper les informations et communications de la communauté. Dans le cadre de l'Intelligence Ambiante, nous avons proposé un *serveur de communications* permettant de faciliter la mise en relation des employés et visiteurs d'une entreprise.

Les modèles ont été mis en oeuvre dans deux contextes de déploiement. Le premier est un prototype de serveur de communications en fonctionnement *client/serveur*. Le second est une *bibliothèque* pour la plate-forme MadKit qui permet de généraliser le modèle de communication

lié au modèle Agent-Groupe-Role.

Enfin, nous avons réalisé deux implémentations de la dynamique de l'environnement, grâce au système à base de règles JESS et grâce à nos algorithmes adaptés, que nous avons évaluées. JESS utilise l'*algorithme RETE* pour gérer l'environnement sous forme de faits et de règles. Nous avons également proposé deux algorithmes, l'un fondé sur la description *"statique"* du système multi-agents, et l'autre fondé sur sa description *"dynamique"*.

La figure 2 donne une vue d'ensemble du travail réalisé dans le cadre de cette thèse.

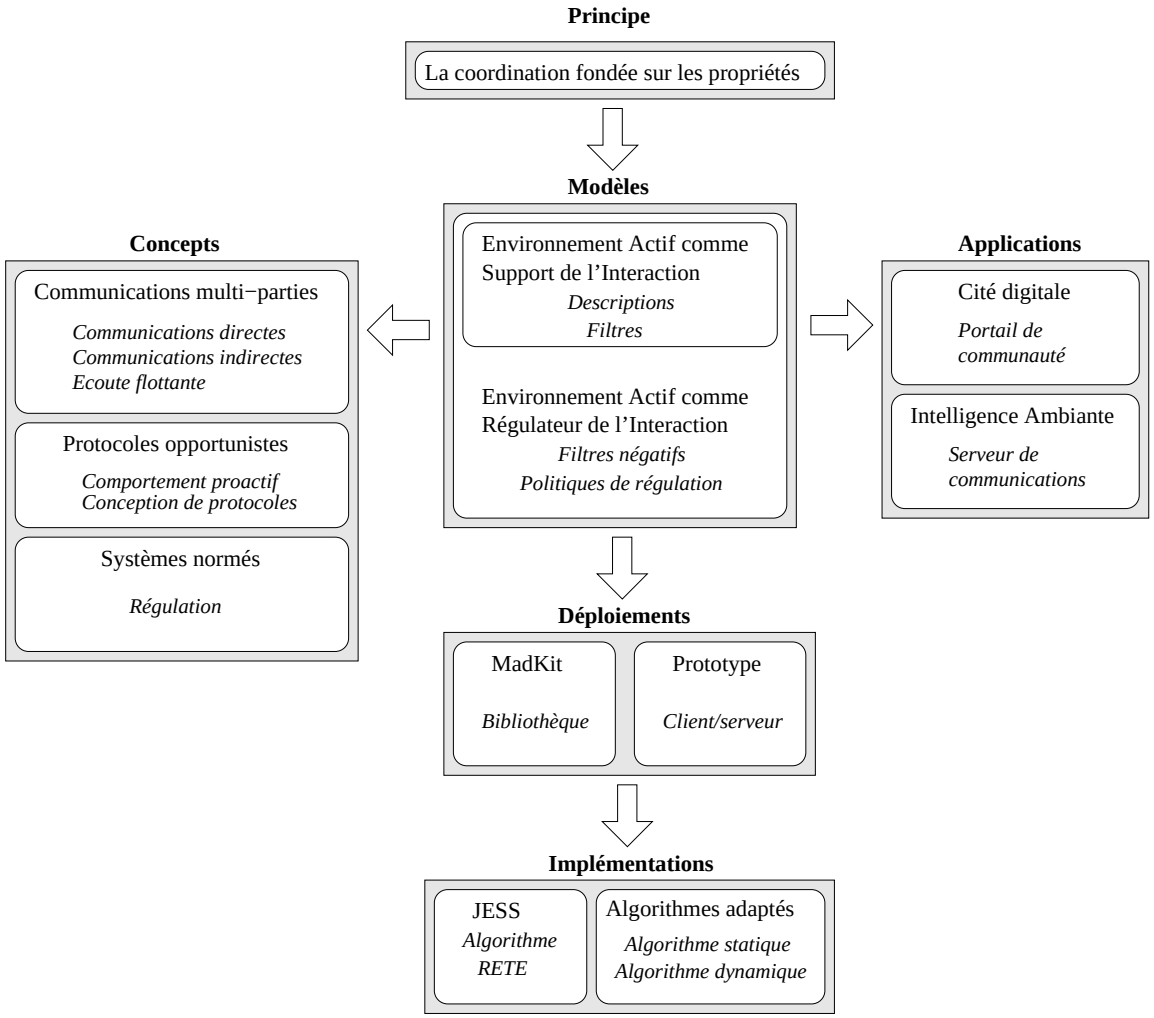


FIGURE 2 – Récapitulatif des apports de la thèse

Dans la suite, nous détaillons les points importants de notre proposition :

- **La mise en oeuvre des communications multi-parties**

Notre approche s'inspire de travaux dans les domaines de la sociologie, de la psychologie et du travail coopératif. Nous avons défini les communications multi-parties selon deux points de vue : du point de vue multi-agents, les communications multi-parties sont des communications dont les récepteurs sont choisis par une conjonction des initiatives des émetteurs et des récepteurs. Du

point de vue de l'agent, nous avons montré que l'expression de leurs initiatives donne aux agents la capacité de diriger leur attention en fonction de leurs besoins. Cette capacité s'apparente aux interactions indirectes, puisque le futur récepteur récupère lui-même l'information.

• **Un principe : la coordination fondée sur les propriétés**

Dans le cadre des travaux de l'équipe sur le thème de l'environnement, nous avons proposé le principe de *coordination fondée sur les propriétés* :

> Une communication a lieu si un ensemble de conditions sur l'état du système multi-agents est satisfait.

Ces conditions peuvent être choisies à l'initiative de l'émetteur et/ou du récepteur, mais dans tous les cas il existe des règles explicites ou implicites qui permettent l'interaction.

Notre modélisation repose sur une description des entités du système multi-agents au sein de l'environnement. Chaque entité est décrite dynamiquement grâce à un ensemble de propriétés, et l'ensemble des descriptions des entités représente l'état courant du système. Nous avons proposé la mise en place de *filtres* pour représenter les besoins en communication des agents. Ces filtres sont des conditions sur l'état du système. Un filtre met en relation les messages avec leurs récepteurs dans certains contextes grâce aux descriptions.

• **Le modèle "Environnement Actif comme Support de l'Interaction"**

Dans cette thèse, nous proposons une modélisation des interactions permettant l'expression standardisée des besoins en communication des agents, issue de l'étude des communications multi-parties. Cette modélisation, appelée EASI (Environnement Actif comme Support des Interactions) est celle d'un environnement actif, les agents lui déléguant les tâches d'adressage des données et de recherche d'information. Nous avons choisi une formalisation adaptée à ce type de problème, issue de l'analyse de données symboliques (ADS).

En s'appuyant sur les concepts logiques d'intension et d'extension, cette formalisation permet de décrire et de manipuler des classes de données. Cette formalisation est expressive car elle permet l'utilisation de tous types de données, qu'elles soient qualitatives, quantitatives ou complexes. Enfin, le modèle est à la fois indépendant de l'implémentation et exploitable, car les définitions de classes peuvent être traduites directement en formules de logique du premier ordre.

La formalisation ADS permet de prendre en compte le contexte courant pour mener à bien les tâches de l'environnement, et ainsi d'intégrer au sein d'un même système multi-agents l'utilisation de modèles d'interaction différents. Une des originalités du modèle EASI est d'unifier les interactions directes, qui sont fondées sur la sélection dynamique des récepteurs, et les interactions indirectes, qui sont fondées sur la sélection dynamique des messages.

• **Les algorithmes de gestion de l'environnement**

Les filtres effectuent des tests et appariements entre les propriétés des entités du système multi-agents. Toutes les entités du système multi-agents possédant une description, le temps de calcul des appariements devient rapidement grand quand le nombre d'entités augmente. Or, notre environnement doit effectuer le transfert des messages rapidement.

Pour accélérer le processus de test et d'appariement, nous avons introduit un niveau de description intermédiaire qui permet de classifier *a priori* les entités selon les filtres. Il s'agit de regrouper les entités par catégories décrites par les mêmes propriétés, mais n'ayant pas nécessairement les mêmes valeurs. Cette classification *a priori* a un coût lors de la création des classes, mais la recherche des récepteurs d'un filtre est plus rapide puisqu'elle se fait sur des ensembles de descriptions plus petits. Cette organisation en catégories s'appuie sur la formalisation de l'analyse de données symboliques.

Un second moyen pour accélérer le filtrage des messages est de prendre également en compte les valeurs des propriétés pour classifier les agents. Le coût de la classification est plus important que la classification par propriétés du fait de la dynamicité des valeurs mais, à nouveau, la recherche des récepteurs est plus rapide grâce à une taille de classe plus restreinte.

Nous avons donc proposé deux algorithmes fondés respectivement sur ces deux pré-classifications, *statique* et *dynamique*. Ces algorithmes de traitement et de mise à jour des descriptions tirent profit de l'organisation des informations du modèle EASI.

• **Le modèle "Environnement Actif comme Régulateur de l'Interaction"**

EASI est une approche au niveau multi-agents, et non de l'agent lui-même. Dans cette optique, l'environnement est un outil pour concevoir et gérer le système dans son ensemble. Pour cela, nous avons approfondi le modèle sur les aspects de facilitation et de régulation des communications. Faciliter, pour que le concepteur puisse intégrer un comportement par défaut pour la transmission des informations entre les agents du système. Réguler et contrôler, de façon d'une part à assurer un service aux agents, et d'autre part à limiter les risques liés aux agents malveillants ou mal conçus.

La régulation des interactions par l'environnement permet la mise en oeuvre de politiques d'interactions différenciées en fonction des comportements souhaités. Ces extensions du modèle EASI sont regroupées au sein d'un nouveau modèle nommé EARI (Environnement Actif comme Régulateur de l'Interaction). Par rapport aux modèles existants, l'originalité du modèle EARI est de permettre de contrôler les communications en temps réel et non *a posteriori*, et donc de mettre en place des règles impératives, tout en conservant la richesse et la flexibilité du modèle EASI.

• **Validation des modèles EASI et EARI**

La dernière partie de notre travail montre l'exploitation des modèles. Nous avons évalué les performances de deux méthodes de mise en oeuvre de l'environnement de communication, grâce à un système à base de règles et grâce à nos algorithmes. Nous les avons comparé avec le modèle généralement utilisé pour le support des communication multi-parties, la diffusion.

Les systèmes à base de règles sont une façon effective d'implémenter le stockage et la dynamique de l'environnement, mais ses gains s'avèrent limités en comparaison de ceux obtenus avec le modèle de diffusion. En comparaison, nos algorithmes proposés avec les modèles EASI et EARI montrent un gain d'efficacité important par rapport à la diffusion. Si le nombre d'agents est faible et/ou si le taux de mise à jour des valeurs des propriétés est élevé (les descriptions sont modifiées plus souvent que des messages ne sont envoyés), l'algorithme statique est le plus

efficace ; dans les autres cas le choix doit se porter sur l'algorithme dynamique. L'algorithme statique ne conserve que les informations structurelles, et n'est donc pas sensible aux mises à jour des valeurs des propriétés. En revanche, l'algorithme dynamique est plus efficace pour mener à bien les appariements.

Nous avons également étudié la définition de protocoles utilisant les communications multi-parties qui permettent la mise en oeuvre de comportements opportunistes.

Enfin, nous avons montré comment modéliser un système normé grâce au modèle EARI. Cette modélisation est un moyen efficace de mise en oeuvre de systèmes multi-agents dans lesquels les propriétés des communications ont été vérifiées au préalable par une formalisation à l'aide de la logique déontique.

Perspectives

Notre travail de thèse ouvre un certain nombre de perspectives, à la fois sous forme d'extensions et de thèmes à plus grande échéance :

• **Une méthodologie pour la conception des systèmes multi-agents avec EASI et EARI**

Nous avons introduit un nouveau connecteur AUML pour aider le concepteur du système multi-agents à modéliser les communications multi-parties. De plus, tout au long de cette thèse, nous avons souligné la répartition des tâches entre l'environnement et les agents : le premier propose des normes (les filtres facilitateurs) et impose des règles (les filtres impératifs) de communications, tandis que les seconds utilisent les filtres pour exprimer leurs besoins en fonction des filtres déjà présents dans l'environnement.

Une piste de recherche est alors de formaliser le processus de conception du système multi-agents au sein d'une méthodologie, qui permette de dissocier les phases de conception de l'environnement et celles de conception des agents, ainsi que les informations nécessaires à une conception décentralisée -à plusieurs concepteurs- d'agents hétérogènes.

• **La distribution des données**

Nous prenons en compte l'état complet du système multi-agents, modélisé par les descriptions des entités. La contrepartie est une centralisation des informations pour pouvoir y accéder lors de la distribution des messages. Dans l'optique de distribuer les descriptions, nous envisageons deux possiblités : la première est un compromis entre distribution de l'environnement et efficacité du modèle en ne partageant plus toutes les informations au niveau global, par exemple par un système de type TuCSoN permettant une hiérarchisation de la portée des données entre le niveau local (de l'hôte) et le niveau global (du système multi-agents). La seconde est d'étudier des mécanismes de redondance et de séparation des contextes différents permettant d'améliorer la robustesse et de limiter l'effet de goulot d'étranglement du système.

• **Le contrôle des accès aux descriptions**

Nous avons vu que le contrôle des accès aux descriptions est pour l'instant limité, puisqu'il est associé à une notion de propriété stricte : les agents peuvent ajouter des descriptions et des

filtres, et seul le propriétaire d'une description ou d'un filtre peut les modifier ou supprimer.

Pour flexibiliser le contrôle d'accès, une perspective est de proposer un contrôle d'accès fondé sur les propriétés, inspiré à la fois du contrôle d'accès à base de rôles et du modèle EASI : l'idée est d'associer des permissions d'accès à des descriptions en intension.

Dans le modèle d'environnement EASI, il s'agit de modifier le module de *vérification* des accès par un ensemble de *filtres de permissions*, chacun composé de deux éléments : une assertion décrivant l'initiateur de l'accès, et les droits d'accès qui lui sont adjoints. A chaque tentative de modification d'une description ou d'un filtre, l'environnement vérifie si cet accès est autorisé à l'aide des filtres de permissions.

• **Etendre l'utilisation de l'Analyse de Données Symboliques**

Notre formalisation utilise celle des données symboliques. L'analyse de données symboliques permet d'analyser et classifier des données complexes. La sauvegarde des données d'exécution du système multi-agents peut permettre un retour d'analyse *a posteriori*, pour le concepteur, du déroulement de l'exécution du SMA. En particulier, il peut permettre de découvrir des redondances de communications entre agents, et plus généralement extraire l'organisation du système. Ce retour est un atout pour mettre en place de nouvelles règles de l'environnement, soit pour favoriser les communications ayant déjà été observées, soit pour empêcher des comportements non souhaités.

• **La dynamique du SMA**

Nous avons souligné que nos modèles sont des environnements de communication qui peuvent s'intégrer à des environnements prenant en charge d'autres responsabilités, comme le cycle de vie des agents.

Une responsabilité à explorer en lien avec les communications est la dynamique des entités confiées à l'environnement. Par exemple, une phéromone est un message dont l'intensité varie au cours du temps, et qui a la particularité de pouvoir fusionner avec d'autres phéromones.

Pour la mise en oeuvre de ce type d'interactions, il est donc nécessaire de pouvoir gérer au niveau de l'environnement les modifications d'états des messages. Dans nos modèles, cette gestion est implicitement réalisée par les agents. Une seconde perspective liée à cette question est celle de la fusion des données. Il s'agit de permettre une manipulation de transformation et d'agrégation des données du contexte.

De la même façon, la consultation des filtres dans l'environnement permet d'obtenir les filtres "bruts". Pour raisonner sur les filtres dans leur ensemble, les agents doivent eux-mêmes réaliser le traitement de ces informations. D'autres formats de présentation condensée des données sont envisagés : un arbre RETE, un treillis des descriptions utilisées, *etc.*.

• **L'introduction du temps dans le modèle**

L'introduction de la dynamique est un enjeu majeur pour les modèles EASI et EARI. La formalisation du modèle présentée dans cette thèse ne prend pas en compte le temps. Il n'est donc pas possible de modéliser la dynamique du système multi-agents *a priori*. Nous envisageons de formaliser cette dynamique par l'introduction d'une logique temporelle, qui contienne les transitions entre états de l'environnement.

La prise en compte du temps permet aussi d'envisager la question de l'auto-adaptabilité des règles de l'environnement et des politiques de priorité en fonction de l'évolution de la topologie des descriptions du système multi-agents.

• **Les politiques de régulation**

La modélisation présentée en section 7.3.2 montre la forte imbriquation des concepts de notre modèles avec les trois opérateurs (Obligation, Interdiction et Permission) de la logique déontique.

Nous comptons introduire la logique déontique dans la formalisation des filtres positifs et négatifs. La formalisation permet de vérifier les propriétés du système multi-agents lors de sa conception. Ces propriétés peuvent ensuite être contrôlées lors de l'exécution.

De façon cohérente avec le point précédent, la logique déontique choisie doit être temporelle.

• **L'amélioration des algorithmes**

Notre implémentation sur la base de l'algorithme RETE a montré que celui-ci n'était pas adapté au cadre de notre modèle, notamment à cause des coûts induits par la reconstruction de l'arbre à chaque ajout de message. Nous avons présenté les algorithmes d'exécution dynamique des filtres dont les résultats sont satisfaisants.

Il reste nécessaire d'optimiser le traitement des descriptions. Une piste est de différencier le traitement des propriétés mises à jour souvent (par exemple la localisation) de celles rarement mises à jour (par exemple l'age).

• **Les domaines applicatifs**

Nos tests ne prennent pas en compte de façon particulière les réseaux sans fils, où par exemple le broadcast est aussi coûteux en bande passante qu'une communication point-à-point. Pour autant, des travaux existent sur les systèmes Publish and Subscribe en environnement mobile, ainsi que dans le cadre de l'intelligence ambiante. Il serait donc intéressant d'étudier les spécificités à mettre en oeuvre pour prendre en compte ce type d'environnement.

Annexe A

Niveaux d'interaction et couches réseau

A.1 Modèles de couches réseau

A.1.1 OSI : interconnexion des systèmes ouverts

La norme complète, de référence ISO 7498 est globalement intitulée « Modèle basique de référence pour l'interconnexion des systèmes ouverts (OSI) » et est un modèle comprenant sept couches :

- 7 : Application *ex. HTTP, SMTP, SNMP, FTP, Telnet, NFS*
- 6 : Présentation *ex. XDR, ASN.1, AFP*
- 5 : Session *ex. ISO 8327 / CCITT X.225, RPC, SMB, ASP*
- 4 : Transport *ex. TCP, UDP, RTP, SPX, Netbios, ATP*
- 3 : Réseau *ex. IP (IPv4 ou IPv6), ICMP, IGMP, X.25, CLNP, ARP, OSPF, RIP, IPX, DDP*
- 2 : Liaison *ex. Ethernet, Token Ring, PPP, HDLC, Frame relay, RNIS, ATM, Wi-Fi*
- 1 : Physique *ex. techniques de codage du signal (électronique, radio, laser, Bluetooth, ...) pour la transmission des informations sur les réseaux physiques (réseaux filaires, optiques, radioélectriques ...)*

A.1.2 TCP/IP

Le modèle TCP/IP, dont le document de référence est le RFC 1122, comprend quand à lui seulement quatres couches. Habituellement, les trois couches supérieures du modèle OSI sont considérées comme une seule couche Application dans TCP/IP. Comme TCP/IP n'a pas de couche session unifiée sur laquelle les couches plus élevées peuvent s'appuyer, ces fonctions sont généralement remplies par chaque application (ou ignorées).

- 5 : Application *ex. HTTP, FTP, DNS*
- 4 : Transport *ex. TCP, UDP, RTP*
- 3 : Réseau *IP*
- 2 : Liaison *ex. Ethernet, Token Ring, etc.*
- 1 : Physique *ex. la boucle locale (transmission par modulation sur lignes analogiques : lignes téléphoniques RTC, numériques, ADSL ...), les grandes artères de communication (transmission par multiplexage, commutation, ...), les réseaux de radiocommunication (radio, téléphonie sans fil, satellite, ...)*

A.2 Correspondances

La figure 1.2 présentée au chapitre 1 est une adaptation du principe de couches conceptuelles aux systèmes multi-agents. Cependant, celles-ci ne se situent pas au même niveau. En effet, les couches réseaux représentent la pile des protocoles utilisés dans une communication entre applications distantes, tandis que notre approche concerne les concepts d'ingénierie des SMA.

De cette façon, les deux premières couches décrites dans les modèles OSI et TCP/IP correspondent au niveau que nous avons dénommé *support physique*, et les deux suivantes (couches 3 et 4) sont gérées par le niveau *support logique*. En revanche, le support logique de notre classification ne gère pas que les aspects réseaux, puisque nous lui adjoignons les notions de plate-forme spécifiques au SMA, *cf.* 1.1.2. Enfin, la couche application concerne, pour les systèmes multi-agents, soit le support logique qui fournit l'encapsulation des protocoles, par exemple par l'utilisation d'une plate-forme gérant le transport de messages, soit le niveau agent lui-même si aucun middleware ou facilitateur ne leur est proposé.

Annexe B

Plate-formes aux normes FIPA

Nous présentons ici trois des principales plateformes compatibles avec les normes de la FIPA (*Foundation for Intelligent Physical Agents*).

B.1 JADE

Jade [Bellifemine et al., 2001] a été développé par le groupe de recherche de Gruppo Telecom, en Italie, avec pour objectif de simplifier le développement de systèmes multi-agents inter-opérables par le biais de la norme FIPA. Bien que la plate-forme soit librement utilisable, le codes source n'est pas accessible.

Elle inclut donc les trois rôles prédéfinis dans la norme FIPA, dans le cadre de la plate-forme en elle-même, ainsi que des bibliothèques de développement des agents. On trouve dans JADE une gestion des messages par allocation de queues privées de messages à chaque agent, lequel pourra parcourir ses messages aussi bien séquentiellement que par le biais d'appariements de motifs (*pattern-matching*). Il est possible de transférer des messages aussi bien en interne que vers des systèmes multi-agent externes aux normes FIPA.

JADE utilise une abstraction des comportements afin de modéliser les tâches que l'agent peut exécuter, et les agents instancient leurs comportements en fonction de leurs besoins et de leurs capacités. Une super classe Agent donne toutes les propriétés desquelles les autres éléments héritent, ce qui permet d'obtenir un comportement fondamental global qui sera ensuite instancié suivant les tâches de chaque Agent. JADE offre une bibliothèque de support pour la plupart des comportements de base type gestion des messages ou décomposition de tâches complexes.

JADE comprend aussi des outils de gestion, observation et déboguage de l'exécution, tâches elle-mêmes réalisées à base d'agents.

JADE constitue donc un lien entre simplicité de développement et support normatif, afin de garantir l'inter-opérabilité dans des environnements ouverts.

B.2 ZEUS

ZEUS [Nwana et al., 1999] a été développé par British Telecom, afin de développer des applications collaboratives.

ZEUS se base sur la notion de rôle, consistant pour chaque agent à contrôler un système externe qui réalise une tâche du domaine d'application, système externe pouvant être par exemple un autre système multi-agents. Le système est surtout accompagné d'une méthodologie en quatre points : analyse du domaine, conception, réalisation et support d'exécution.

L'analyse n'est pas supportée par un outil logiciel particulier, mais la modélisation des rôles peut se faire en UML, de nombreux patrons existant déjà.

La conception d'un agent sera constituée de trois niveaux : définition [29], organisation, coordination. Le modèle sous-jacent de ZEUS tend à la création d'agents coopératifs orientés-tâche,

29. Capacités de l'agent à raisonner et à apprendre (architecture interne)

ce qui nécessite un effort de conception du fait de la difficulté de systématiser la recherche de solution. Techniquement, le support est simple, mais la méthodologie n'est pas formalisée.

D'un point de vue développement, toutes les phases passent par des outils graphiques cherchant à simplifier le travail de conception. Ces phases sont la création d'ontologie, d'agent, configuration de l'agent utilitaire, tâche et enfin implémentation des agents. Les agents ne sont par contre pas réutilisables d'un projet à l'autre.

Le déploiement est également géré graphiquement et permet de nombreuses visualisations, notamment d'un point interactionnel et organisation, ainsi que la modification dynamique de l'état interne des agents.

ZEUS est donc une plate-forme complètement intégrée, gérant toutes les étapes de la construction d'un SMA, aussi bien par des outils théoriques que logiciels. On pourra cependant lui reprocher de ne supporter que le modèle des agents coopératifs.

B.3 FIPA-OS

Implémenté par Nortel Networks en tant que logiciel libre, FIPA-OS [Poslad et al., 2000] adhère totalement aux spécifications citées au début de cette section :

- Les trois rôles principaux de gestion de la plate-forme et des communications.
- Un support complet de FIPA-ACL, et donc des agents fondés de logiques multimodales de type FIPA-SL.
- L'interface d'inter-opérabilité des plate-formes.

L'accent est donc porté sur l'ouverture de FIPA-OS, lequel a déjà été utilisé avec succès en interaction avec d'autres systèmes hétérogènes.

La plate-forme supporte la production d'agents en tous langages, et propose deux modèles instanciables. Le développeur dispose d'outils de gestion des communication, de la configuration dynamique de la plate-forme, des interfaces abstraites et des outils de diagnostic et visualisation du SMA.

Les agents communiquent par le biais des standards FIPA, et proposent les outils d'envoi, réception, construction de message. FIPA-OS permet aussi de construire des interactions stables en créant des agents gestionnaires de communication traitant des protocoles complets, comme FIPA-Request.

La plate-forme a été validée pour des applications allant du commerce électronique aux services de réservation.

FIPA-OS est donc une plate-forme générale de support de systèmes multi-agents, dont les principaux avantages sont la parfaite adéquation aux standards FIPA et donc les capacités d'interopérabilité avec des systèmes hétérogènes, et le support professionnel du développement de la plate-forme.

Annexe C

Langage de balisage pour l'expression des filtres et des descriptions dans le prototype

Dans cette annexe, nous introduisons brièvement le langage de balisage que nous avons choisi pour l'expression des filtres et des descriptions.

Le langage choisi est une extension de RuleML [30]. Le *Rule Markup Language* est une initiative visant à proposer un langage de règles standardisé fondé sur XML.

C.1 Modification de la base de connaissance

La modification de la base de connaissance (faits et règles) que nous utilisons pour l'implémentation est réalisée à l'aide des trois balises suivantes :

```
<assert>
</assert>

<retract>
</retract>

<modify>
</modify>
```

Les règles et faits sont ajoutés entre les balises correspondantes.

Nous avons ajouté la balise `modify` à des fins d'efficacité. Modifier un fait est moins complexe qu'un retrait suivi d'un ajout, bien que le résultat soit équivalent. En effet, la modification de la valeur d'une propriété n'entraîne aucun changement au niveau de la structure associée à sa Pdescription, tandis qu'un retrait ou un ajout affectent cette structure.

De façon à respecter le modèle de données, les descriptions sont des multi-champs non-ordonnés composés d'un nom et d'une ou plusieurs valeurs.

```
<Atom>
  <Rel>Description</Rel>
  <slot>
    <Ind>p1</Ind>
    <Ind>value</Ind>
  </slot>
  <slot>
    <Ind>p2</Ind>
    <Ind>value</Ind>
  </slot>
</Atom>
```

30. `http://www.ruleml.org/`

C.2 Expression des filtres

Les filtres sont de la forme (*body* implique *head*). En général, ils auront la forme suivante :

```
<assert>
   <implies>
        <head>
          <expr>
           <fun> receive </fun>
             <var> agent </var>
             <var> message </var>
          </expr>
        </head>
        <body>
          <and>
            <atom>
              <rel> Description </rel>
              <slot> <ind> id </ind>
                <var> agent </var>
              </slot>
              <slot>
                 ...
              </slot>
            </atom>
            <atom>
              <rel> Description </rel>
              <slot> <ind> id </ind>
                <var> message </var>
              </slot>
              <slot>
                 ...
              </slot>
            </atom>
             ...
          </and>
        </body>
   </implies>
</assert>
```

Les fonctions `fun` sont définies par l'utilisateur, et correspondent dans le cadre du modèle EASI à la transmission des messages.

Par exemple, un filtre pour l'écoute flottante posé par un agent $a1$ peut s'exprimer sous forme de règle :

```
<assert>
   <implies>
        <head>
          <expr>
           <fun> receive </fun>
             <var> a1 </var>
             <var> message </var>
          </expr>
        </head>
        <body>
          <and>
            <atom>
              <rel> Description </rel>
              <slot> <ind> id </ind>
                <var> message </var> </slot>
              <slot> <ind> sender </ind>
                <ind> ag </ind> </slot>
            </atom>
            <atom>
              <rel> Description </rel>
              <slot><ind> id </ind>
                <var> ag </var> </slot>
              <slot><ind> type </ind>
                <ind> client </ind>
              </slot>
            </atom>
          </and>
        </body>
   </implies>
</assert>
```

Cette règle permet à l'agent $a1$ d'écouter tous les messages émis par les agents dont la propriété *type* a pour valeur *client*.

D'origine, seul l'opérateur d'égalité est défini. Le concepteur peut définir d'autres opérateurs, tels $>$, $\geq$, $<$, $\leq$. Par exemple, nous avons défini l'opérateur `greaterThan`, qui compare deux valeurs A et B. Cet opérateur s'utilise de la façon suivante :

```
  <Atom>
    <Rel>greaterThan</Rel>
    <Var>B</Var>
    <Var>A</Var>
  </Atom>
```

Notons qu'une adaptation de RuleML aux bases réactives nommée *Reaction RuleML*[31] est actuellement développée. Bien qu'encore naissante, nous envisageons de faire évoluer notre langage vers ce futur standard.

31. `http://ibis.in.tum.de/research/ReactionRuleML/`

Annexe D

Paquetage EASI pour la plate-forme MadKit

D.1 Le diagramme de classes

Le paquetage a été développé en Java, son diagramme de classes est donné en figure D.1. L'un des principes de MadKit est l'agentification de tous les services, autrement dit l'encapsulation des services dans des agents. A ce titre, l'environnement que nous avons développé est en fait un agent de MadKit. Il repose sur un système expert, dont l'interface est définie par *ESLayer*. Nous proposons une implémentation de cette interface par le système expert JESS (*JessLayer*). Les primitives pour manipuler l'environnement sont contenues dans la classe *EASIAgent*, et un prototype de message pour EASI est instancié par la classe *EASIMessage*.

Pour développer son application, le concepteur dispose donc d'un environnement prêt à l'emploi. L'implémentation des agents et messages en tant que classes héritant respectivement de *EASIAgent* et *EASIMessage* permet d'utiliser le modèle de communication EASI.

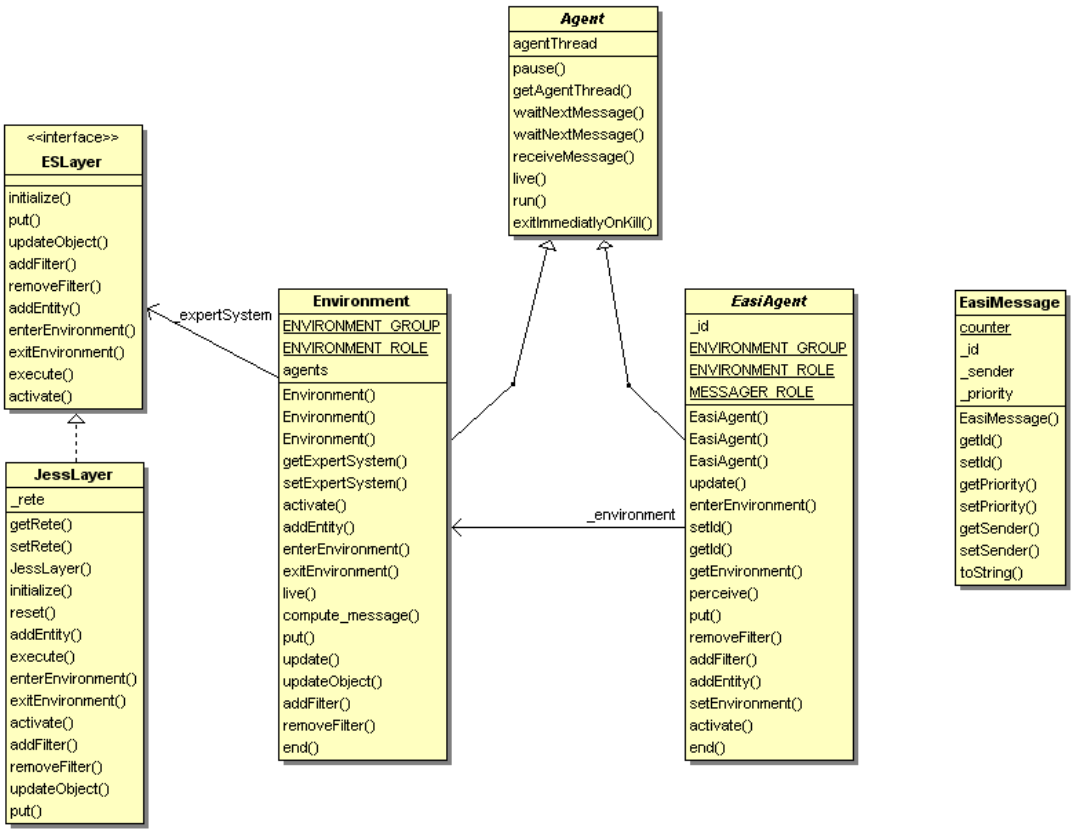


FIGURE D.1 – Plugin EASI pour MadKit

Nous voyons dans la section suivante les éléments importants du paquetage.

D.2 Le système expert pour la gestion des communications

Le stockage des données et l'appariement sont implémentés à l'aide d'un système expert, dont nous détaillons l'interface.

L'interface avec le système expert

ESLayer
activate()
addEntity(String st1, String st2)
addFilter(String name, int priority, String premises, String actions)
enterEnvironment(String st1, Object obj)
execute(String st)
exitEnvironment(Object obj)
initialize()
put(String classname, easi.EasiMessage msg)
removeFilter(String name)
updateObject(Object obj)

Nous pouvons regrouper les fonctions en trois catégories : la vie de l'environnement, l'inscription des agents, et la gestion des filtres et descriptions.

Les fonctions initialize() et activate() sont utiles à MadKit lors de la création et du lancement de l'environnement.

Les fonctions enterEnvironment(String st1, Object obj) et exitEnvironment(Object obj) sont dédiés à l'inscription des agents dans l'environnement, ainsi qu'à leur retrait.

L'ajout et le retrait des filtres sont effectués par les fonctions addFilter(String name, int priority, String premises, String actions) et removeFilter(String name) .

La fonction addEntity(String st1, String st2) permet de déclarer un nouveau type d'objet dont la description sera utilisée par la suite. Les entités sont, dans cette plate-forme, gérées d'une façon particulière : elles sont enregistrées directement auprès de l'environnement, qui extrait de façon automatique leur description. L'enregistrement se fait de la même façon que les agents, par enterEnvironment(String st1, Object obj) et exitEnvironment(Object obj). Lorsque la valeur d'une propriété est modifiée, la fonction updateObject(Object obj) permet de mettre à jour les données au sein du système expert.

La fonction put(String classname, easi.EasiMessage msg) permet de déposer un message dans l'environnement. La fonction execute(String st) n'est pas disponible aux agents, mais à l'environnement, dans le cas où l'instruction que le système-expert doit effectuer n'est pas couverte par les autres fonctions.

D.3 Agents : Création et Primitives

Les agents ont une interface similaire avec l'environnement.

EasiAgent()
EasiAgent(easi.Environment _environmentValue)
EasiAgent(java.lang.String id, easi.Environment _environmentValue)

Ces constructeurs permettent de créer les agents. Par défaut, la référence à l'environnement est nulle et l'identifiant est créé automatiquement à partir d'un attribut non instancié *static* incrémenté.

activate()
addEntity(java.lang.String nameEnvironment, java.lang.String nameClass
addFilter(java.lang.String name, int priority, java.lang.String lh, java.lang.String rh)
end()
enterEnvironment(java.lang.String nameclass, java.lang.Object object)
getEnvironment()
getId()
perceive(easi.EasiMessage m)
put(java.lang.String className, easi.EasiMessage m)
removeFilter(java.lang.String name)
setEnvironment(easi.Environment environmentValue)
setId(java.lang.String id)
update()

Nous retrouvons dans ce tableau les fonctions évoquées du côté du système-expert. Disponibles aux agents, elles se chargent de faire le lien entre les actions des agents et l'environnement. La fonction *perceive* est celle activée par l'environnement pour déposer les messages dans la boîte aux lettres de l'agent.

Annexe E

Publications relatives au travail de thèse

Cette annexe présente les différents articles publiés dans le cadre de cette thèse.

Le principe de property-based coordination a été introduit dans [Zargayouna, Saunier, et Balbo, 2006], avec sa mise en oeuvre grâce au modèle EASI et une application au transport.

Le modèle EASI a fait l'objet d'une première publication dans [Balbo, Zargayouna, et Saunier, 2005b]. Sa formalisation et une application, AIS (*Agent Information Server*), ont également fait l'objet de deux articles [Balbo, Saunier, Pinson, et Zargayouna, 2005a; Zargayouna, Balbo, et Saunier, 2005].

Son rapport aux communications multi-parties a été explicité dans deux articles [Saunier et Balbo, 2007a,b], qui replacent le modèle EASI dans le contexte de ces communications et expose nos algorithmes et leurs expérimentations.

L'extension d'EASI pour la régulation des interactions EARI a été introduite dans [Saunier, Balbo, et Badeig, 2007], puis approfondie dans [Saunier et Balbo, 2008], avec l'exemple d'intelligence ambiante, la mise en oeuvre des politiques de priorités et les résultats d'expérimentation.

Le connecteur UML pour l'écoute flottante a été publié dans [Balbo, 2004].

Liste des publications

– [Balbo, Maudet, et Saunier, 2004b] Flavien Balbo, Nicolas Maudet, et Julien Saunier. Interactions opportunistes par l'écoute flottante. Dans O. Boissier et Z. Guessoum, editors, *Actes des Journées Francophones sur les Systèmes Multi-Agents*, pages 265–270, November 2004. Short paper.

– [Balbo, Saunier, Pinson, et Zargayouna, 2005a] Flavien Balbo, Julien Saunier, Suzanne Pinson, et Mahdi Zargayouna. An operational model for mutual awareness. Dans Michal Pechoucek, Paolo Petta, et László Zsolt Varga, editors, *Multi-Agent Systems and Applications IV, 4th International Central and Eastern European Conference on Multi-Agent Systems, CEEMAS 2005, Budapest, Hungary, September 15-17, 2005, Proceedings*, volume 3690 of *Lecture Notes in Computer Science*, pages 531–534. Springer, 2005. ISBN 3-540-29046-X.

– [Balbo, Zargayouna, et Saunier, 2005b] Flavien Balbo, Mahdi Zargayouna, et Julien Saunier. Informational middleware based on mutual awareness. Dans *IAT '05 : Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 258–261, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2416-8.

– [Saunier et Balbo, 2007a] Julien Saunier et Flavien Balbo. Vers un support des communications multi-parties pour les systèmes multi-agents. Dans *Actes des quatrièmes journées francophones Modèles Formels de l'Interaction (MFI'07)*, pages 397–404. Annales du LAMSADE, 2007. Short paper.

– [Saunier et Balbo, 2007b] Julien Saunier et Flavien Balbo. An environment to support multi-party communications in multi-agent systems. Dans *Multi-Agent Systems and Applications V, 5th International Central and Eastern European Conference on Multi-Agent Systems, CEEMAS 2007, Leipzig, Germany*, pages 52–61. Springer Verlag, 2007.

– [Saunier et Balbo, 2007c] Julien Saunier et Flavien Balbo. Regulated multi-party communications and context awareness through the environment. *International Journal on Multi-Agent and Grid Systems*, 2008. à paraitre.

– [Saunier, Balbo, et Badeig, 2007] Julien Saunier, Flavien Balbo, et Fabien Badeig. Environment as active support of interaction. Dans *Proceedings of Environment for Multi-Agent Systems, Workshop held at the Fifth Joint Conference on Autonomous Agents and Multi-Agent Systems*, volume 4389 of *LNAI*, pages 87–105. Springer Verlag, 2007.

– [Zargayouna, Balbo, et Saunier, 2005] Mahdi Zargayouna, Flavien Balbo, et Julien Saunier. Agent information server : a middleware for traveler information. Dans *6th International Workshop on Engineering Societies in the Agents World (ESAW'05)*, volume 3963 of *LNAI*, pages 3–16. Springer Verlag, 2005.

– [Zargayouna, Saunier, et Balbo, 2006] Mahdi Zargayouna, Julien Saunier, et Flavien Balbo. Property based coordination. Dans Jerome Euzenat et John Domingue, editors, *Artificial Intelligence : Methodology, Systems, Applications*, volume 4183 of *Lecture Notes in Artificial Intelligence*, pages 3–12. Springer Verlag, 2006.

Index

Index

Bibliographie

Aknine, S., Pinson, S., et Shakun, M. F. An extended multi-agent negotiation protocol. *Autonomous Agents and Multi-Agent Systems*, 8(1) :5–45, 2004.

Arcos, J. L., Noriega, P., Rodriguez-Aguilar, J. A., et Sierra, C. E4mas through electronic institutions. Dans *Proceedings of Environment for Multi-Agent Systems, Workshop held at the Fifth Joint Conference on Autonomous Agents and Multi-Agent Systems*, tome 4389 de *Lecture Notes in Artificial Intelligence*, pages 184–202. Springer Verlag, 2007.

Baez-Barranco, J.-A., Stratulat, T., et Ferber, J. A unified model for physical and social environments. Dans *Proceedings of Environment for Multi-Agent Systems, Workshop held at the Fifth Joint Conference on Autonomous Agents and Multi-Agent Systems*, tome 4389 de *Lecture Notes in Artificial Intelligence*, pages 41–50. Springer Verlag, 2007.

Bajcsy, R. Active perception. *IEEE Proceedings*, 76(8) :996–1005, 1988.

Balbo, F. A model of environment, active support of the communication. Dans *American Association of Artificial Intelligence Conference, AAAI-99, Workshop on Reasoning in Context for AI Applications*, pages 1–5. AAAI Press, 1999.

Balbo, F. *Un modèle d'Interaction Multi-Agent utilisant l'Environnement comme Support Actif de Communication. Application à la gestion des transports urbains.* Thèse de doctorat, Université Paris-Dauphine, 2000.

Balbo, F. A new interaction model for agent based simulation. Dans G. Horton, rédacteur, *Proc. of ESM'04, the 18th European Simulation Multiconference*, pages 372–376. SCS–European Publishing House, Magdeburg, DE, 2004.

Balbo, F., Maudet, N., et Saunier, J. Interactions opportunistes par l'écoute flottante. Dans O. Boissier et Z. Guessoum, rédacteurs, *Actes des Journées Francophones sur les Systèmes Multi-Agents*, pages 265–270. Hermès - Lavoisier, 2004. Short paper.

Balbo, F. et Pinson, S. Toward a multi-agent modelling approach for urban public transportation systems. Dans A. Omicini, P. Petta, et R. Tolksdorf, rédacteurs, *Engineering Societies in the Agent World II*, tome 2203 de *Lecture Notes in Artificial Intelligence*, pages 160–174. Springer Verlag, 2001.

Balbo, F., Saunier, J., Pinson, S., et Zargayouna, M. An operational model for mutual awareness. Dans M. Pechoucek, P. Petta, et L. Z. Varga, rédacteurs, *Multi-Agent Systems and Applications IV, fourth International Central and Eastern European Conference on Multi-Agent Systems, CEEMAS 2005, Budapest, Hungary, September 15-17, 2005, Proceedings*, tome 3690 de *Lecture Notes in Computer Science*, pages 531–534. Springer Verlag, 2005a.

Balbo, F., Seghrouchni, A. E. F., et Pinson., S. *Organisations et Applications des Systèmes Multi-Agents*, chapitre Coordination d'actions par planification multi-agent, pages 93–108. Hermès - Lavoisier, 2002.

Balbo, F., Zargayouna, M., et Saunier, J. Informational middleware based on mutual awareness. Dans *IAT '05 : Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 258–261. IEEE Computer Society, Washington, DC, USA, 2005b.

Baldauf, M., Dustdar, S., et Rosenberg, F. A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, 2 :263–277(15), 2007.

Bellifemine, F., Poggi, A., et Rimassa, G. Jade : a fipa2000 compliant agent development environment. Dans *AGENTS '01 : Proceedings of the fifth international conference on Autonomous agents*, pages 216–217. ACM Press, New York, NY, USA, 2001.

Benerecetti, M., Bouquet, P., et Bonifacio, M. Distributed context-aware systems. *Human-Computer Interaction*, 16(2/4) :213–228, 2001.

Benini, M., Cindio, F. D., et Sonnante, L. Virtuose, a virtual community open source engine for integrating civic networks and digital cities. Dans *Digital Cities III*, tome 3081 de *Lecture Notes in Computer Science*, pages 217–232. Springer Verlag, 2005.

Bergenti, F. et Ricci, A. Three approaches to the coordination of multiagent systems. Dans *SAC '02 : Proceedings of the 2002 ACM symposium on Applied computing*, pages 367–372. ACM Press, New York, NY, USA, 2002.

Billard, L. et Diday, E. From the statistics of data to the statistics of knowledge : Symbolic data analysis. *Journal of the American Statistical Association*, 98(462) :470– ? ?, 2003.

Billard, L. et Diday, E. *Symbolic Data Analysis : Conceptual Statistics and Data Mining (Wiley Series in Computational Statistics)*. John Wiley & Sons, 2007.

Branigan, H. Perspectives on multi-party dialogue. *Research on Language & Computation*, 4 (2-3) :153–177, 2006.

Bucur, O., Beaune, P., et Boissier, O. Steps towards making contextualized decisions : How to do what you can, with what you have, where you are. Dans T. Roth-Berghofer, S. Schulz, et D. B. Leake, rédacteurs, *MRC*, tome 3946 de *Lecture Notes in Computer Science*, pages 62–85. Springer Verlag, 2005.

Busetta, P., Donà, A., et Nori, M. Channeled multicast for group communications. Dans *AAMAS '02 : Proceedings of the First International Joint Conference on Autonomous Agents and Multi-agent Systems*, pages 1280–1287. ACM Press, New York, NY, USA, 2002.

Camarinha-Matos, L. M. et Afsarmanesh, H. Design of a virtual community infrastructure for elderly care. Dans *PRO-VE '02 : Proceedings of the IFIP TC5/WG5.5 Third Working Conference on Infrastructures for Virtual Enterprises*, page 635. Kluwer, B.V., Deventer, The Netherlands, The Netherlands, 2002.

Carriero, N., Gelernter, D., et Leichter, J. Distributed data structures in linda. Dans *popl'86 : Proceedings of the 13th ACM Sigact-Sigplan symposium on Principles Of Programming Languages*, pages 236–242. 1986.

Castelli, G., Rosi, A., Mamei, M., et Zambonelli, F. A simple model and infrastructure for context-aware browsing of the world. Dans *PERCOM '07 : Proceedings of the Fifth IEEE International Conference on Pervasive Computing and Communications*, pages 229–238. IEEE Computer Society, Washington, DC, USA, 2007.

Chellas, B. F. *Modal logic, an introduction.* Cambridge University Press, 1980.

Cholvy, L., Garion, C., et Saurel, C. Modélisation de réglementations pour le partage d'information dans un sma. Dans *Actes des quatrièmes journées francophones Modèles Formels de l'Interaction (MFI'07)*, pages 389–396. Annales du LAMSADE, 2007.

Coutaz, J., Crowley, J. L., Dobson, S., et Garlan, D. Context is key. *Commun. ACM*, 48(3) :49–53, 2005.

Cremonini, M., Omicini, A., et Zambonelli, F. Coordination and access control in open distributed agent systems : The tucson approach. Dans *COORDINATION '00 : Proceedings of the fourth International Conference on Coordination Languages and Models*, pages 99–114. Springer Verlag, London, UK, 2000.

Daniel, G., Muchnik, L., et Solomon, S. Traders imprint themselves by adaptively updating their own avatar. Dans O. B. Philippe Mathieu, Bruno Beaufils, rédacteur, *Artificial Economics, Agent-Based Methods in Finance, Game Theory and Their Applications*, tome 564 de *Lecture Notes in Economics and Mathematical Systems*, pages 27–38. Springer Verlag, 2005.

Davis, R. et Smith, R. G. Negotiation as a metaphor for distributed problem solving. pages 333–356, 1988.

Denti, E., Natali, A., et Omicini, A. On the expressive power of a language for programming coordination media. Dans *1998 ACM Symposium on Applied Computing (SAC'98)*, pages 169–177. ACM, Atlanta, GA, USA, 1998. Special Track on Coordination Models, Languages and Applications.

Diday, E. et Noirhomme-Fraiture, M. *Symbolic Data Analysis and the SODAS Software*. Wiley-Interscience, New York, NY, USA, 2008.

Dignum, F. et Vreeswijk, G. Towards a testbed for multi-party dialogues. Dans *Workshop on Agent Communication Languages*, tome 2922 de *Lecture Notes in Computer Science*, pages 212–230. Springer Verlag, 2003.

Drogoul, A., Corbara, B., et Lalande, S. MANTA : New experimental results on the emergence of (artificial) ant societies. Dans N. Gilbert et R. Conte, rédacteurs, *Artificial Societies : The Computer Simulation of Social Life*, pages 190–211. UCL Press, London, 1995.

Dugdale, J., Pavard, J., et Soubie, B. A pragmatic development of a computer simulation of an emergency call center. Dans *Designing Cooperative Systems : The Use of Theories and Models*, pages 241–256. IOS Press, 2000.

Engelmore, R. S. et Morgan, A. J. *Blackboard Systems*. Addison-Wesley, 1988.

Esteva, M., Rodriguez-Aguilar, J., Rosell, B., et Arcos, J. Ameli : An agent-based middleware for electronic institutions. Dans R. Jennings, C. Sierra, L. Sonenberg, et M. Tambe, rédacteurs, *Proceedings of the third International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2004)*, pages 236–243. ACM Press, 2004.

Eugster, P. T., Felber, P. A., Guerraoui, R., et Kermarrec, A.-M. The many faces of publish/-subscribe. *ACM Comput. Surv.*, 35(2) :114–131, 2003.

Fahy, P. et Clarke, S. CASS : Middleware for mobile, context-aware applications. Dans *Workshop on Context Awareness at MobiSys 2004*. ACM Press, 2004.

Ferber, J. *Les Systèmes Multiagents : Vers une Intelligence Collective*. InterEditions, 1995.

Ferber, J. et Drogoul, A. Using reactive multi-agent systems in simulation and problem solving. pages 53–80, 1992.

Ferber, J. et Gutknecht, O. A meta-model for the analysis and design of organizations in multi-agent systems. Dans *ICMAS '98 : Proceedings of the third International Conference on Multi-Agent Systems*, pages 128–135. IEEE Computer Society, Washington, DC, USA, 1998.

Ferber, J., Michel, F., et Baez, J. Agre : Integrating environments with organizations. Dans *Proceedings of Workshop on Environments for Multi-Agent Systems*, tome 3374 de *Lecture Notes in Artificial Intelligence*, pages 48–56. Springer Verlag, 2005.

Finin, T., Labrou, Y., et Mayfield, J. KQML as an agent communication language. Dans J. M. Bradshaw, rédacteur, *Software Agents*, chapitre 14, pages 291–316. AAAI Press / The MIT Press, 1997.

FIPA. *FIPA SL Content Language Specification*. Foundation for Intelligent Physical Agents, 2000. Statut : « Standard », http ://www.fipa.org/specs/fipa00008/.

FIPA. *FIPA Abstract Architecture Specification*. Foundation for Intelligent Physical Agents, 2002a. Statut : « Standard, version L », http ://www.fipa.org/specs/fipa00001/.

FIPA. *FIPA ACL Message Structure Specification*. Foundation for Intelligent Physical Agents, 2002b. Statut : « Standard, version G », http ://www.fipa.org/specs/fipa00061/.

Fontoura, M., Lehman, T. J., Nelson, D., Truong, T., et Xiong, Y. TSpaces services suite : Automating the development and management of web services. Dans *WWW (Alternate Paper Tracks)*. 2003.

Forgy, C. L. Rete : A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, 19 :17–37, 1982.

Freeman, E., Hupfer, S., et Arnold, K. *JavaSpaces : Principles, Patterns, and Practice*. Addison-Wesley, 1999.

Friedman-Hill, E. J. Jess : The java expert system shell. Rapport technique, Sandia National Laboratories, 1998.

Genesereth, M. R. et Fikes, R. E. Knowledge interchange format, version 3.0 reference manual. Rapport technique Logic-92-1, Computer Science Department, Stanford University, Stanford, CA, USA, 1992.

Gouaïch, A., Michel, F., et Guiraud, Y. Mic* : a deployment environment for autonomous agents. Dans *Proceedings of Workshop on Environments for Multi-Agent Systems*, tome 3374 de *Lecture Notes in Artificial Intelligence*, pages 109–126. Springer Verlag, 2005.

Gutknecht, O. et Ferber, J. MadKit : A generic multi-agent platform. Dans C. Sierra, M. Gini, et J. S. Rosenschein, rédacteurs, *Proceedings of the Fourth International Conference on Autonomous Agents*, pages 78–79. ACM Press, Barcelona, Catalonia, Spain, 2000. Poster announcement.

Gutnik, G. Monitoring large-scale multi-agent systems using overhearing. Dans *AAMAS '05 : Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 1377–1377. ACM Press, New York, NY, USA, 2005.

Gutnik, G. et Kaminka, G. A. A scalable petri-net representation of interaction protocols for overhearing. Dans R. van Eijk, M. . P. Huget, et F. Dignum, rédacteurs, *Developments in Agent Communication*, tome 3396 de *Lecture Notes in Artificial Intelligence*, pages 50–64. Springer Verlag, 2005.

Hattori, F., Ohguro, T., Yokoo, M., Matsubara, S., et Yoshida, S. Socialware : multiagent systems for supporting network communities. *Commun. ACM*, 42(3) :55–ff., 1999.

Heath, C., Svensson, M. S., Hindmarsh, J., Luff, P., et vom Lehn, D. Configuring awareness. *Comput. Supported Coop. Work*, 11(3) :317–347, 2002.

Hill, E. F. *Jess in Action : Java Rule-Based Systems*. Manning Publications Co., Greenwich, CT, USA, 2003.

Huang, Y. et Garcia-Molina, H. Publish/subscribe in a mobile environment. Dans *MobiDe '01 : Proceedings of the second ACM international workshop on Data engineering for wireless and mobile access*, pages 27–34. ACM Press, New York, NY, USA, 2001.

Huget, M.-P. *Une ingénierie des protocoles d'interaction pour les systèmes multi-agents*. Thèse de doctorat, Université Paris-Dauphine, 2001.

Huget, M.-P. et Demazeau, Y. First steps towards multi-party communication. Dans R. M. van Eijk, M.-P. Huget, et F. Dignum, rédacteurs, *Agent Communication*, tome 3396 de *Lecture Notes in Computer Science*, pages 65–75. Springer Verlag, 2004.

JAAMAS, 2007. International journal on autonomous agents and multi-agent systems (14 (1)), special issue on environments for multi-agent systems, kluwer academic publisher. 2007.

Julien, C. et Roman, G.-C. Supporting context-aware interaction in dynamic multi-agent systems. Dans D. Weyns, H. V. D. Parunak, et F. Michel, rédacteurs, *E4MAS*, tome 3374 de *Lecture Notes in Computer Science*, pages 168–189. Springer Verlag, 2004.

Kamali, K., Fan, X., et Yen, J. Formal semantics for multiparty proactive communication in agent teams. Dans *Advances in Agent Communication*, Lecture Notes in Artificial Intelligence. Springer Verlag, 2006.

Kaminka, G., Pynadath, C., et Tambe, M. Monitoring teams by overhearing : A multi-agent plan-recognition approach. *Journal of Artificial Intelligence Research*, 17 :83–135, 2002.

Khadraoui, D. et Gateau, B. A prototype for an agent-based electronic contracting using an organizational model. Dans *CIMCA '05 : Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce Vol-2 (CIMCA-IAWTIC'06)*, pages 686–691. IEEE Computer Society, Washington, DC, USA, 2005.

Knoblock, C. A., Arens, Y., et Hsu, C.-N. Cooperating agents for information retrieval. Dans *Proceedings of the Second International Conference on Cooperative Information Systems*. University of Toronto Press, Toronto, Ontario, Canada, 1994.

Kumar, S., Huber, M. J., McGee, D., Cohen, P. R., et Levesque, H. J. Semantics of agent communication languages for group interaction. Dans *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pages 42–47. AAAI Press / The MIT Press, 2000.

Lee, K. J. et Chang, Y. S. Time-bounded negotiation framework for multi-agent coordination. Dans T. Ishida, rédacteur, *Proceedings of the first Pacific Rim International Workshop on Multiagent Platforms (PRIMA-98)*, tome 1599 de *Lecture Notes in Artificial Intelligence*, pages 61–75. Springer Verlag, Berlin, 1999.

Legras, F. *Organisation dynamique d'équipes d'engins autonomes par écoute flottante*. Thèse de doctorat, ONERA, 2003.

Legras, F. et Tessier, C. Lotto : Group formation by overhearing in large teams. Dans *Advances in Agent Communication*, tome 2922 de *Lecture Notes in Artificial Intelligence*, pages 254–270. Springer Verlag, 2004.

Lehman, T. J., Cozzi, A., Xiong, Y., Gottschalk, J., Vasudevan, V., Landis, S., Davis, P., Khavar, B., et Bowman, P. Hitting the distributed computing sweet spot with tspaces. *Comput. Networks*, 35(4) :457–472, 2001.

Malville, E. et Bourdon, F. Task allocation : A group self-design approach. Dans *ICMAS*, pages 166–173. IEEE Computer Society, 1998.

Mamei, M., Zambonelli, F., et Leonardi, L. Tuples on the air : a middleware for context-aware computing in dynamic networks. *Distributed Computing Systems Workshops, 2003. Proceedings. 23rd International Conference on*, pages 342–347, 2003.

Minsky, N. H., Minsky, Y., et Ungureanu, V. Safe tuplespace-based coordination in multiagent systems. *Applied Artificial Intelligence*, 15(1) :11–33, 2001.

Morin, E. *La méthode. Tome 1. La nature de la nature*. Éditions du Seuil, Paris, 1977.

Nwana, H. S., Ndumu, D. T., Lee, L. C., et Collis, J. C. Zeus : a toolkit and approach for building distributed multi-agent systems. Dans *AGENTS '99 : Proceedings of the third annual conference on Autonomous Agents*, pages 360–361. ACM Press, New York, NY, USA, 1999.

Odell, J., Parunak, H. V. D., et Bauer, B. Extending uml for agents. Dans *Proceedings of the Agent-Oriented Information Systems Workshop at the 17th Conference on Artificial Intelligence*. 2000.

Okuyama, F. Y., Bordini, R. H., et da Rocha Costa, A. C. Spatially distributed normative infrastructure. Dans *Proceedings of Environment for Multi-Agent Systems, Workshop held at the Fifth Joint Conference on Autonomous Agents and Multi-Agent Systems*, tome 4389 de *Lecture Notes in Artificial Intelligence*, pages 203–220. Springer Verlag, 2007.

Omicini, A. et Zambonelli, F. Tuple centres for the coordination of internet agents. Dans *SAC '99 : Proceedings of the 1999 ACM symposium on Applied computing*, pages 183–190. ACM Press, New York, NY, USA, 1999.

Osborn, S. Mandatory access control and role-based access control revisited. Dans *RBAC '97 : Proceedings of the second ACM workshop on Role-based access control*, pages 31–40. ACM, New York, NY, USA, 1997.

Paes, R., Carvalho, G., Gatti, M., Lucena, C., Briot, J.-P., et Choren, R. Enhancing the environment with a law-governed service for monitoring and enforcing behavior in open multi-agent systems. Dans *Proceedings of Environment for Multi-Agent Systems, Workshop held at the Fifthourth Joint Conference in Autonomous Agents and Multi-Agent Systems*, tome 4389 de *Lecture Notes in Artificial Intelligence*, pages 221–238. Springer Verlag, 2007.

Picco, G. P. et Buschini, M. L. Exploiting transiently shared tuple spaces for location transparent code mobility. Dans *COORDINATION '02 : Proceedings of the 5th International Conference on Coordination Models and Languages*, pages 258–273. Springer Verlag, London, UK, 2002.

Picco, G. P., Murphy, A. L., et Roman, G.-C. Lime : Linda meets mobility. Dans *ICSE '99 : Proceedings of the twenty-first international conference on Software engineering*, pages 368–377. IEEE Computer Society Press, Los Alamitos, CA, USA, 1999.

Platon, E., Mamei, M., Sabouret, N., Honiden, S., et Parunak, H. V. Mechanisms for environments in multi-agent systems : Survey and opportunities. *Autonomous Agents and Multi-Agent Systems*, 14(1) :31–47, 2007a.

Platon, E., Sabouret, N., et Honiden, S. T-compound : An agent-specific design pattern and its environment. Dans *Proceeding of the third international workshop on Agent Oriented Methodologies at OOPSLA*, pages 63–74. 2004.

Platon, E., Sabouret, N., et Honiden, S. Overhearing and direct interactions : Point of view of an active environment. Dans *Proceedings of Environment for Multi-Agent Systems, Workshop held at the Fourth Joint Conference in Autonomous Agents and Multi-Agent Systems*, tome 3830 de *Lecture Notes in Artificial Intelligence*, pages 121–138. Springer Verlag, 2005.

Platon, E., Sabouret, N., et Honiden, S. Smart environment for smarter agents in e-markets. Dans *Proceedings of the Florida Artificial Intelligence Research Society Conference'06*. 2006.

Platon, E., Sabouret, N., et Honiden, S. Tag interactions in multiagent systems : Environment support. Dans *Proceedings of Environment for Multi-Agent Systems, Workshop held at the Fifth Joint Conference in Autonomous Agents and Multi-Agent Systems*, tome 4389 de *Lecture Notes in Artificial Intelligence*, pages 106–123. Springer Verlag, 2007b.

Poslad, S., Buckle, P., et Hadingham, R. FIPA-OS : the FIPA agent platform available as open source. Dans J. Bradshaw et G. Arnold, rédacteurs, *Proceedings of the 5th International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM 2000)*, pages 355–368. The Practical Application Company Ltd., Manchester, UK, 2000.

Ricci, A., Omicini, A., Viroli, M., Gardelli, L., et Oliva, E. Cognitive stigmergy : Towards a framework based on agents and artifacts. Dans D. Weyns, H. V. D. Parunak, et F. Michel, rédacteurs, *Environments for MultiAgent Systems*, tome 4389 de *Lecture Notes in Artificial Intelligence*, pages 124–140. Springer Verlag, 2007.

Robertson, T. The public availability of actions and artefacts. *Computer Supported Cooperative Work*, 11(3-4) :299–316, 2002.

Rognin, L., Salembier, P., et Zouinar, M. Cooperation, interactions and socio-technical reliability : the case of air-traffic control. Dans T. R. G. GREEN, L. BANNON, C. P. WARREN, et J. BUCKLEY, rédacteurs, *Proceedings of the European Conference on Cognitive Ergonomics (ECCE9)*. 1998.

Russell, S. J. et Norvig, P. *Artificial Intelligence : a modern approach*. Prentice Hall, Upper Saddle River, N.J., second international edition édition, 2003.

Russell, S. J. et Zilberstein, S. Anytime sensing, planning, and action : A practical model for robot control. Dans R. Bajcsy, rédacteur, *Proceedings of the International Conference on Artificial Intelligence (IJCAI-93)*, pages 1402–1407. Morgan Kaufmann publishers Inc. : San Mateo, CA, USA, Chambéry, France, 1993.

Samarati, P. et di Vimercati, S. D. C. Access control : Policies, models, and mechanisms. Dans *FOSAD '00 : Revised versions of lectures given during the IFIP WG 1.7 International School on Foundations of Security Analysis and Design on Foundations of Security Analysis and Design*, pages 137–196. Springer Verlag, London, UK, 2001.

Sandholm, T. et Lesser, V. Issues in automated negotiation and electronic commerce : Extending the contract net framework. Dans *Proceedings of the First International Conference on Multiagent Systems.*, pages 328–335. AAAI Press / MIT Press, Menlo park, California, 1995.

Saunier, J. et Balbo, F. An environment to support multi-party communications in multi-agent systems. Dans *Multi-Agent Systems and Applications V, 5th International Central and Eastern European Conference on Multi-Agent Systems, CEEMAS 2007, Leipzig, Germany*, pages 52–61. Springer Verlag, 2007a.

Saunier, J. et Balbo, F. Vers un support des communications multi-parties pour les systèmes multi-agents. Dans *Actes des quatrièmes journées francophones Modèles Formels de l'Interaction (MFI'07)*, pages 397–404. Annales du LAMSADE, 2007b. Short paper.

Saunier, J. et Balbo, F. Regulated multi-party communications and context awareness through the environment. *International Journal on Multi-Agent and Grid Systems*, 2008. To appear.

Saunier, J., Balbo, F., et Badeig, F. Environment as active support of interaction. Dans *Proceedings of Environment for Multi-Agent Systems, Workshop held at the Fifth Joint Conference*

on Autonomous Agents and Multi-Agent Systems, tome 4389 de *Lecture Notes in Artificial Intelligence*, pages 87–105. Springer Verlag, 2007.

Schelfthout, K., Weyns, D., et Holvoet, T. Middleware for protocol-based coordination in mobile applications. *IEEE Distributed Systems Online*, 7(8), 2006.

Shannon, C. E. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3) :379–423, 1948.

Sycara, K., Paolucci, M., Velsen, M. V., et Giampapa, J. The retsina mas infrastructure. *Autonomous Agents and Multi-Agent Systems*, 7(1-2) :29–48, 2003.

Sycara, K., Widoff, S., Klusch, M., et Lu, J. Larks : Dynamic matchmaking among heterogeneous software agents in cyberspace. *Autonomous Agents and Multi-Agent Systems*, 5(2) :173–203, 2002.

Sycara, K. et Wong, H. A taxonomy of middle-agents for the internet. Dans *ICMAS '00 : Proceedings of the Fourth International Conference on MultiAgent Systems (ICMAS-2000)*, pages 465–466. IEEE Computer Society, Washington, DC, USA, 2000.

Traum, D. et Rickel, J. Embodied agents for multi-party dialogue in immersive virtual worlds. Dans *AAMAS '02 : Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 766–773. ACM Press, New York, NY, USA, 2002.

Tummolini, L., Castelfranchi, C., Ricci, A., Viroli, M., et Omicini, A. "exhibitionists" and "voyeurs" do it better : A shared environment approach for flexible coordination with tacit messages. Dans *Proceedings of Workshop on Environments for Multi-Agent Systems*, tome 3374 de *Lecture Notes in Artificial Intelligence*, pages 215–231. Springer Verlag, 2004.

Valckenaers, P., Sauter, J., Sierra, C., et Rodriguez-Aguilar, J. A. Applications and environments for multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 14(1) :61–85, 2007.

van den Besselaar, P. et Beckers, D. The life and death of the great amsterdam digital city. Dans *Digital Cities III*, tome 3081 de *Lecture Notes in Computer Science*, pages 66–96. Springer Verlag, 2005.

Viroli, M., Holvoet, T., Ricci, A., Schelfthout, K., et Zambonelli, F. Infrastructures for the environment of multiagent systems. *Autonomous Agents and Multi-Agent Systems*, 14(1) :49–60, 2007.

Warren, R. *Auditory perception. A new analysis and synthesis.* Cambridge : Cambridge University Press, 1999.

Weiss, G., rédacteur. *Multiagent Systems : A Modern Approach to Distributed Artificial Intelligence.* The MIT Press, Cambridge, MA, USA, 1999.

Weyns, D., Bouck, N., et Holvoet, T. Gradient field-based task assignment in an agv transportation system. Dans *AAMAS '06 : Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 842–849. ACM Press, New York, NY, USA, 2006.

Weyns, D., Omicini, A., et Odell, J. Environment as a first-class abstraction in multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 14(1) :5–30, 2007. Special Issue on Environments for Multi-agent Systems.

Weyns, D., Parunak, H. V. D., Michel, F., Holvoet, T., et Ferber, J. Environments for multiagent systems, state-of-the-art and research challenges. *Lecture Notes in Computer Science Series*, 3374 :2–52, 2005.

Weyns, D., Steegmans, E., et Holvoet, T. Towards active perception in situated multi-agent systems. *Special Issue of Journal on Applied Artificial Intelligence*, 18 (9-10) :867–883, 2004.

Wooldridge, M. *An Introduction to Multi-Agent Systems*. John Wiley and Sons, 2002.

Wright, I. et Marshall, J. The execution kernel of rc++ : Rete*, a faster rete with treat as a special case. *International Journal of Intelligent Games and Simulation*, 2(1) :36–48, 2003.

Zargayouna, M., Balbo, F., et Saunier, J. Agent information server : a middleware for traveler information. Dans *6th International Workshop on Engineering Societies in the Agents World (ESAW'05)*, tome 3963 de *Lecture Notes in Artificial Intelligence*, pages 3–16. Springer Verlag, 2005.

Zargayouna, M., Saunier, J., et Balbo, F. Property based coordination. Dans I. J. Euzenat et J. Domingue, rédacteurs, *Artificial Intelligence : Methodology, Systems, Applications*, tome 4183 de *Lecture Notes in Artificial Intelligence*, pages 3–12. Springer Verlag, 2006.

Zheng, W., Serban, C., et Minsky, N. Establishing global properties of multi-agent systems via local laws. Dans *Proceedings of Environment for Multi-Agent Systems, Workshop held at the Fifth Joint Conference in Autonomous Agents and Multi-Agent Systems*, tome 4389 de *Lecture Notes in Artificial Intelligence*, pages 170–183. Springer Verlag, 2007.

Zieba, B., van Sinderen, M., et Wegdam, M. Quality-constrained routing in publish/subscribe systems. Dans *MPAC '05 : Proceedings of the third international workshop on Middleware for pervasive and ad-hoc computing*, pages 1–8. ACM Press, New York, NY, USA, 2005.

Vu :
Le Président
M.

Vu :
Les Suffrageants
M.

Vu et permis d'imprimer :
Le Vice-Président du Conseil Scientifique Chargé de la Recherche de l'Université Paris IX Dauphine

Résumé

Dans cette thèse, nous proposons une modélisation des communications multi-parties et leur régulation dans les systèmes multi-agents. Cette modélisation, appelée EASI (Environnement Actif comme Support des Interactions) est celle d'un environnement actif. Les agents lui déléguent les tâches d'adressage des données et de recherche d'information. La formalisation est issue de l'analyse de données symboliques, adaptée à ce type de problèmes. Les apports de ce modèle sont (1)la prise en compte du contexte pour mener à bien les tâches de l'environnement, et (2) la standardisation des interactions directes, fondées sur la sélection dynamique des récepteurs, et des interactions indirectes, fondées sur la sélection dynamique des messages.

Nous généralisons ensuite le modèle pour réguler et contrôler les communications, en proposant le modèle EARI (Environnement Actif comme Régulateur de l'Interaction). L'environnement met en oeuvre des politiques d'interactions différenciées en fonction des comportements souhaités : services de communications standard et/ou règles impératives. Les agents peuvent également accroître ou restreindre leur attention (*awareness*) aux messages des autres agents. Les modèles sont complétés par des algorithmes de traitement et de mise à jour des informations, et par une architecture fonctionnelle de l'environnement. Ils ont été mis en oeuvre et évalués dans deux contextes de déploiement, un prototype en fonctionnement client/serveur et une extension de la plate-forme MadKit. Les tests réalisés ont montré un gain d'efficacité important de nos algorithmes sur le modèle généralement utilisé pour les communication multi-parties, la diffusion.

Mots-clés: Systèmes multi-agents, Environnement, Communication multi-parties, Médiation

Abstract

In this Ph.d thesis, we propose a model of multi-party communications and their control for multi-agent systems. This model, called EASI (Environment as Active Support of Interaction) is that of an active environment. The agents delegate to it their data addressing and information search tasks. The formalization comes from Symbolic Data Analysis, which is suitable for this kind of problems. The contributions are (1) the use of the context to carry through the tasks of the environment, and (2) the standardization of direct interactions, based on the dynamic choice of the receivers, and indirect interactions, based on the dynamic selection of data.

We extend our model to regulate and control the communications, with the EARI (Environment as Active Regulator of the Interaction) model. The environment carries out different communication policies, depending on the desired behavior : standard communication services and/or mandatory rules. The agents can also increase or decrease their awareness of other agents' messages. The models are completed with algorithms for data management and update and by an abstract architecture. They have been applied and evaluated in two different deployment contexts, a client/server prototype and an extension of the MadKit platform. The evaluation of our algorithms show a significant improvement in efficiency compared to the usual means of supporting multi-party communications, broadcast.

Keywords: Multi-agent Systems, Environment, Multi-party Communication, Mediation