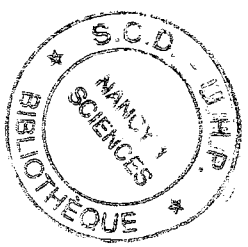
Modèle de réorganisation de systèmes multi-agents : une approche descriptive et opérationnelle

Rémy Foisel

Département de formation doctorale en Informatique
UFR STMIA

École doctorale IAE + M



Modèle de réorganisation de systèmes multi-agents : Une approche descriptive et opérationnelle

THÈSE

présentée et soutenue publiquement le 13 novembre 1998

pour l'obtention du

Doctorat de l'université Henri Poincaré - Nancy 1

(spécialité informatique)

par

Rémy FOISEL

Composition du jury

Président :	André Schaff	Professeur, Université Henri Poincaré, Nancy 1
Rapporteurs :	Jean-Pierre Müller	Professeur, Université de Neuchâtel
	Joël Quinqueton	Directeur de recherche, INRIA, Montpellier
Directeurs de thèse :	Jean-Paul Haton	Professeur, Université Henri Poincaré, Nancy 1
	Vincent Chevrier	Maître de conférences, Université Henri Poincaré, Nancy 1, ESSTIN
E		Maître de conférences, Université de Paris 6

n Informatique et ses Applications - UMR 7503

REMERCIEMENTS

Le travail de recherche présenté dans ce mémoire a été mené au sein de l'équipe RFIA et MAIA du laboratoire LORIA.

Je tiens à remercier tout particulièrement les personnes ayant permis l'aboutissement des travaux de thèse :

Je remercie vivement Jean-Paul Haton, professeur à l'université Henri Poincaré, Nancy 1, et Vincent Chevrier, maître de conférences à l'université Henri Poincaré de m'avoir proposé ce sujet de thèse et de m'avoir encadré conjointement pendant celle-ci. Je les remercie de leurs conseils, de leur disponibilité, de leur réactivité à la lecture et à la correction de mes documents, et surtout de leur soutien.

Mes remerciements s'adressent ensuite naturellement à tous ceux qui ont accepté de participer au jury de thèse :

Je remercie Jean-Pierre Müller, professeur à l'université de Neufchâtel, Joël Quinqueton, directeur de recherche INRIA au LIRMM de Montpellier, et André Schaff, professeur à l'université Henri Poincaré, Nancy 1, d'avoir accepté de rapporter avec soin ce mémoire.

Je remercie Alexis Drogoul, maître de conférences à l'université de Paris 6 de sa participation à mon jury, d'avoir manifesté un très grand intérêt à mes travaux et pour son soutien.

J'adresse mes remerciements à toutes les personnes de l'équipe RFIA, de l'équipe MAIA et du laboratoire LORIA ayant contribué de près ou de loin à ce travail de recherche. Je tiens à remercier en particulier :

François Charpillet, chargé de recherche à l'INRIA Lorraine, de sa disponibilité et de son intérêt pour mon sujet et mes travaux.

Dominique Colnet, maître de conférence à l'université Henri Poincaré, Nancy 1, de sa disponibilité et pour nos nombreuses discussions.

Olivier Festor, chargé de recherche à l'INRIA Lorraine et Laurent Andrey, maître de conférences à l'université Henri Poincaré, Nancy 1, pour leur disponibilité et les réunions de travail parfois improvisées.

Martine Kuhlmann, secrétaire de l'équipe, pour son accueil et sa disponibilité.

Je remercie vivement également tous ceux qui ont contribué à développer une ambiance de travail agréable. En particulier, un grand merci à tous mes compagnons au sein du laboratoire : Lionel et Michel pour leur amitié et leur humour, Arnaud, Jean, Marc, Olivier, et Pierre pour nos discussions caustiques, Jean-Michel pour ses blagues, Christophe pour l'ambiance dans le bureau, et à la gent féminine Céline et Iadine. A tous ceux qui n'ont pas encore terminé, je leur souhaite une agréable fin de thèse.

Enfin un grand merci à tous mes amis qui m'ont encouragé de près ou de loin pendant la fin de ma thèse.

A toute ma famille, mes remerciements les plus chaleureux, du précieux soutien qu'elle m'a toujours apporté.

Table des Matières

Liste des figures

Chapitre 1 Introduction générale

L'homme a toujours essayé d'analyser et de comprendre le fonctionnement de la pensée et de l'intelligence chez les êtres vivants. Les philosophes et les biologistes ont été les premiers à s'atteler à cette tâche. Avec l'arrivée des ordinateurs durant le XX^{ième} siècle, les chercheurs se sont alors demandé comment la machine pourrait faire ce que seuls des cerveaux faisaient jusqu'alors. C'est ainsi qu'est apparue l'*Intelligence Artificielle*, qui cherche à rendre ces ordinateurs doués, dans une certaine limite, de raison et de faculté d'action en fonction de leur raisonnement. Les recherches issues de ce domaine tendent à créer des entités intelligentes et autonomes pouvant rivaliser avec l'être humain dans des domaines précis et résolvant des problèmes complexes qui nécessitaient encore l'homme pour son savoir-faire et ses facultés intellectuelles.

Cette forme d'intelligence autonome semble très avantageuse lorsque l'entité est seule, ou que le problème est spécifique à un domaine précis. Mais elle peut devenir source de difficultés lorsque le problème fait intervenir de multiples domaines. Une solution pratique est de décomposer en modules d'expertise les différents domaines et de centraliser le contrôle de ces différentes modules. Ce type de solution est adoptée dans les *systèmes experts*. Ainsi, au lieu de se trouver devant une entité intelligente, on se trouve devant un ensemble de modules interconnectés grâce à un contrôle centralisé. Mais il faut alors se poser plusieurs questions avant de valider unanimement cette solution. Peut-on trouver l'architecture optimale quel que soit l'ensemble des domaines intervenant dans le problème? Le concepteur peut-il entrevoir les interconnexions nécessaires et idéales entre de multiples modules complexes?

Le problème principal est que, dans la solution précédente, les modules ou entités ont perdu une grande partie de leur autonomie, car leurs interconnexions sont câblées dans une architecture centralisée. Une autre solution est d'utiliser des entités autonomes (appelées agents) et non des modules. C'est de façon générale, le type d'approche qui est étudié dans l'*Intelligence Arti-*

ficielle Distribuée. La résolution de problème se fait grâce à des entités qui se coordonnent et interagissent afin de tirer le meilleur parti de leurs caractéristiques et de leurs savoir-faire individuels pour une résolution collective du problème.

Cette approche fournit une plus grande liberté en terme de résolution de problème. La résolution peut-être contrôlée de diverses manières : centralisée ou décentralisée. La résolution ne consiste pas seulement à l'exécution d'une entité informatique, mais d'un système composé de plusieurs entités informatiques. De plus, cette résolution ne se contente pas d'être la somme des résolutions partielles de chaque entité composant le système, mais de combiner ces résolutions par des influences, des lois, des interactions entre les agents. Cette liberté pose alors des problèmes pour la mise en place des interactions entre les agents : comment définir ces interactions?

La multitude d'interactions qui peuvent prendre place entre les agents d'un système, si elles permettent la résolution du problème abordé, amènent également des situations conflictuelles (conflit d'action, accès à une ressource, etc) et des situations où des actions redondantes sont entreprises par les agents. Il devient alors important que les performances collectives du système ne soient pas pénalisées voire empêchées par ces interactions «négatives». Le problème qui se pose alors est de fournir aux agents les moyens d'agir localement afin que le système globalement se comporte comme un tout cohérent en sachant que cette définition suppose que :

- les données, connaissances et heuristiques de contrôle nécessaires à la résolution sont distribuées au sein d'un ensemble d'agents. De plus, chaque agent possède ses propres buts, et ces buts locaux peuvent être différents de ceux des autres agents et de ceux du système,

- chaque agent n'a qu'une vue locale du problème. Cette vue peut être incomplète, incohérente avec celle des autres agents. En aucun cas, il n'y a de vue globale du système multi-agents qui soit explicitement représentée.

Une façon de résoudre ce problème est d'organiser les interactions des agents. Une organisation correspond à une description du système à un niveau d'abstraction supérieur à celui de l'agent. Elle fournit un cadre conceptuel pour décrire les activités et les interactions dans le système grâce à la définition de rôle, de comportement, de relation d'autorité, etc. L'organisation structure donc l'activité du système et les interactions entre ses composantes en répartissant les tâches, en limitant les conflits potentiels (relations hiérarchiques), etc.

1 MOTIVATIONS

Les recherches actuelles dans le domaine multi-agents s'intéressent particulièrement aux systèmes constitués de nombreux agents hétérogènes et ayant une très forte dynamique. Chaque agent n'a qu'une vue partielle du système dans lequel il se situe et évolue de manière autonome. Il est donc difficile, dans un tel contexte, de construire un système à partir d'une approche globale ou partielle du problème cible. En effet, la multitude des agents impose une conception globale de l'organisation du système, qui doit être gérée localement au niveau des agents pour conserver l'autonomie de ceux-ci.

D'autre part, la dynamique est importante, car il est impossible de prévoir, dès la conception, toutes les situations pouvant survenir durant le fonctionnement du système. Ces systèmes doi-

vent donc s'adapter à cette dynamique au cours de leur fonctionnement et il est aussi nécessaire de pouvoir aisément modifier cette dynamique ou l'organisation en fonction de situations non prévues lors de la conception.

L'approche multi-agents ajoute donc un aspect dynamique (comme nous le verrons dans le chapitre 5 de la première partie de ce mémoire) aux approches classiques de conception de systèmes qui repose généralement sur une organisation figée pour relier les entités composantes. Cet aspect dynamique complique alors également la conception des systèmes multi-agents, car elle doit prendre en compte la globalité du système, mais aussi se baser sur une adaptation locale aux agents pour conserver l'autonomie de ceux-ci.

Le but de ce travail est d'intégrer les concepts d'agents, d'interactions, et d'organisation des interactions dans une approche descriptive permettant une construction aisée et modulaire de systèmes multi-agents opérationnels composés d'agents hétérogènes.

L'hétérogénité des agents impose de laisser un maximum de liberté au concepteur d'un système pour la définition du comportement interne des agents. Disposer d'une construction aisée et modulaire de système nécessite de disposer d'un cadre de description ne se limitant pas à l'aspect local des agents afin de faciliter la description d'interactions et d'organisations variées, utilisables par un système.

Un tel cadre de description doit ainsi permettre de découpler les comportements spécifiques à une application des agents et l'infra-structure du système (interaction et organisation) qu'ils cherchent à adopter. C'est ici que réside le principal problème auquel se heurtent les concepteurs de systèmes multi-agents : la description d'entités intelligentes est complexe puisqu'elle constitue toute la problématique de l'Intelligence Artificielle, en particulier pour les différents modes de raisonnements [Hato91]. De plus, s'il faut décrire simultanément ces entités intelligentes, leurs interactions et les principes organisationnels, la tâche du concepteur devient très complexe. Certes, cette tâche a souvent été traitée assez facilement parce que le système était composé de peu d'agents ou parce que les agents composant le système étaient peu complexes. Mais comme l'avenir des systèmes multi-agents est d'être composés de plus en plus d'agents qui seront de plus en plus complexes, il est alors nécessaire de se pencher sur ce problème.

2 ORGANISATION DU MÉMOIRE

Ce mémoire est organisé en trois parties :

- La partie I présente les concepts généralement employés dans l'univers des systèmes multi-agents. Nous présentons ainsi le concept d'agent, d'interaction et d'organisation de ces interactions qui permettent de prendre en compte l'aspect multi-agents intervenant dans les systèmes. Cet état de l'art de ces concepts nous permet alors d'établir une rapide catégorisation des travaux du domaine et ainsi de situer notre approche qui s'intéresse principalement à l'interaction.

- La partie II présente et détaille les principes de construction que nous avons choisis pour décrire et opérationnaliser des systèmes. Ces principes consistent à utiliser différents modèles et schémas de description (que nous détaillons) pour représenter les con-

cepts d'agents, d'interactions et de sociétés organisées correspondant à un système applicatif. Nous présentons ensuite comment mettre en liaison la modélisation de ces concepts avec un niveau opérationnel, et comment exprimer des processus de réorganisation, afin d'obtenir un cadre de description de système multi-agents disposant de facultés de réorganisation qui soit opérationnalisable de manière automatique.

- La partie III présente la mise en oeuvre et l'utilisation de notre approche pour la construction de systèmes. Nous présentons tout d'abord nos premiers développements sur l'étude des aspects dynamiques de l'organisation, avant de présenter la plate-forme que nous avons conçue pour décrire des systèmes multi-agents et les rendre opérationnels. Nous présentons ensuite l'utilisation de cette plate-forme pour construire des applications multi-agents avant d'évaluer les possibilités actuelles de cette plate-forme (modularité, réutilisabilité et flexibilité) et ses perspectives futures.

- Enfin, nous terminons ce mémoire par une conclusion générale sur l'état actuel de nos travaux et une présentation des perspectives de nos recherches.

PARTIE I L'UNIVERS MULTI-AGENTS

Chapitre 2 Intelligence Artificielle Distribuée et les Systèmes Multi-Agents

Les systèmes multi-agents appartiennent à un domaine de l'IA appelé intelligence artificielle distribuée (IAD). Ce domaine a émergé il y a une vingtaine d'années et est en constante évolution [Davi80], [Davi82], [Bond88], [Gass87], [Erce93].

Les systèmes multi-agents et l'IAD, en général, permettent d'appréhender des problèmes de taille et de complexité plus importantes que ceux abordés par les approches classiques de l'IA. Dans ce mémoire, nous ne détaillerons pas les apports de cette approche; le lecteur pourra se référer aux ouvrages suivants [Ferb95], [Erce91], [Dema90], [Gass92] pour de plus amples informations. Cette première partie a seulement pour objectif de donner une vision générale du paradigme multi-agents et surtout d'introduire quelques concepts de base communément employés dans les systèmes multi-agents.

Les principes de base de l'IAD et du paradigme multi-agents consistent en la distribution des connaissances et des informations nécessaires à la résolution d'un problème parmi les différentes composantes en interaction d'un système [Chev93]. Les composants de ces systèmes sont des entités plus ou moins intelligentes désignées par le terme "agent".

Un agent est une des entités actives d'un système. Ce terme d'entité active sert à signaler le fait qu'un agent ne correspond pas à la notion de module statique, mais plutôt à une notion d'objets concurrents et autonomes ayant le propre contrôle de leur activité [Hewi77]. Ce premier concept constitue l'élément de base du paradigme multi-agents, nous y consacrons donc le premier chapitre de cette partie.

Par analogie avec les sciences sociales, le terme de société est utilisé pour désigner l'ensemble des agents constituant un système multi-agents. C'est ici que réside pour nous le principal intérêt du paradigme multi-agents : ne pas uniquement concevoir des entités intelligentes, mais

aussi mettre ces entités en relation de manière "intelligente". Cette approche transparaît également au travers de deux autres concepts communément employés dans le domaine : les interactions et l'organisation des agents.

Le terme interaction désigne l'enchaînement d'échanges d'informations ou d'influences ayant lieu entre des agents. La mise en place de ces interactions permet aux agents de résoudre ensemble un problème grâce à leur capacités respectives.

Ce concept est l'un des aspects les plus importants de l'IAD [Deck87] dans la mesure où il permet de relier les agents pour qu'ils constituent un système. L'approche des problèmes est alors telle que leur résolution est au-delà de la complexité de chacun des agents du système mais réside dans la multitude des interactions entre les constituants du système.

Mais ce concept d'interaction ne permet de mettre en relation qu'un sous-ensemble des agents composant un système. Il est alors nécessaire d'utiliser un autre concept plus général pour exprimer les relations liant tous les membres de la société : l'organisation. Ce concept a pour objectif principal de contrôler et de coordonner l'ensemble des interactions présentes entre les agents de la société, afin de structurer les activités des agents et permettre ainsi au système de se comporter comme un tout efficace et cohérent, capable de conduire à une solution du problème.

Le fondement du paradigme multi-agents est évidemment le concept d'agent qui définit les caractéristiques des composants de base d'un système. Mais, afin de permettre une meilleure approche de la conception de systèmes multi-agents, nous pensons qu'il nécessaire de prendre en compte également les concepts d'interaction et d'organisation dans une approche conceptuelle. Les différents chapitres de cette partie ont pour but de présenter les caractéristiques respectives de ces trois concepts autour desquels s'organisera toute la problématique de ce mémoire.

Chapitre 3 Les Agents

Le concept de base du paradigme multi-agents est l'agent. Il correspond à une entité active composant un système multi-agents et intervient dans la résolution de problèmes grâce à ses connaissances, ses compétences (services) et son comportement en participant à la résolution collective par le biais d'interactions avec les autres agents de ce système.

Ce terme d'entité active sert à signaler le fait qu'un agent ne correspond pas à la notion de module statique, mais plutôt à une notion d'objet concurrent et autonome ayant le propre contrôle de son activité [Hewi77]. Si le concept d'agent comme entité est très proche de celui d'objet ou d'acteur, il n'en est pas de même de l'agent comme entité active. En effet, comme le dit Ferber [Ferb97] : les objets n'ont ni but ni recherche de satisfaction. Un objet est défini par un ensemble de services qu'il offre (ses méthodes) et qu'il ne peut refuser d'exécuter si un autre objet lui demande. En revanche les agents disposent d'objectifs qui leur donnent une autonomie de décision vis-à-vis des messages qu'ils reçoivent.

D'autre part, les objets utilisent un mécanisme d'envoi de message qui se résume à un simple appel de procédure. Il n'y a pas de langage de communication à proprement parler. Les mécanismes d'interaction sont donc à la charge du programmeur. Pour les agents, les interactions sont plus complexes et font intervenir des communications de haut niveau (cf chapitre 4), où l'important est que l'agent décide par lui même comment interagir et réagir aux messages qu'il reçoit.

L'originalité et la source de toute la problématique multi-agents résident dans l'hypothèse d'autonomie de décision attribuée aux agents, c'est-à-dire dans la conservation d'un maximum de degrés de liberté pour chaque agent. Cette perspective d'autonomie introduit une dynamique qui complexifie le maintien de la cohérence et de l'efficacité dans un système d'agents, qui doivent combiner une communication plus évoluée que les objets avec une recherche de satisfaction des objectifs de chaque agent.

Dans ce chapitre, nous présentons les caractéristiques générales des agents en définissant la terminologie et une typologie possible des agents, avant de définir plus précisément le caractère autonome attribué aux agents. Nous terminons en présentant les caractéristiques essentielles permettant de définir le comportement d'un agent et en les illustrant par des architectures d'agents, avant de proposer une synthèse des divers points abordés au travers du concept d'agent.

1 TERMINOLOGIE

Les agents sont les entités actives composant un système multi-agents. Des définitions générales de ce terme ont été proposées :

> *"Les agents sont des entités physiques (capteurs, processeurs,...) ou abstraites (tâches à réaliser, déplacements,...), qui sont capables d'agir sur leur environnement et sur elles-mêmes, c'est-à-dire de modifier leur propre comportement. Elles disposent, pour ce faire, d'une représentation partielle de cet environnement et de moyens de perception et de communication."* [Erce91]

> *"On appelle agent une entité physique ou virtuelle :*
> *a. qui est capable d'agir dans un environnement,*
> *b. qui peut communiquer directement avec d'autres agents,*
> *c. qui est mue par un ensemble de tendances (sous la forme d'objectifs individuels ou d'une fonction de satisfaction, voire de survie, qu'elle cherche à optimiser),*
> *d. qui possède des ressources propres,*
> *e. qui est capable de percevoir (mais de manière limitée) son environnement,*
> *f. qui ne dispose que d'une représentation partielle de cet environnement (et éventuellement aucune),*
> *g. qui possède des compétences et offre des services,*
> *h. qui peut éventuellement se reproduire,*
> *i. dont le comportement tend à satisfaire ses objectifs, en tenant compte des ressources et des compétences dont elle dispose, et en fonction de sa perception, de ses représentations et des communications qu'elle reçoit."* [Ferb95]

Les auteurs de ces définitions distinguent trois fonctions principales dans la structure d'un agent : **percevoir, décider et agir**. Ceci induit qu'un agent a des connaissances qui lui sont propres et un comportement autonome lui permettant d'exploiter ses connaissances, de raisonner, et de décider comment et quand agir en fonction de ses interactions avec les autres agents, en fonction de ses perceptions de son environnement et selon les objectifs qu'il cherche à satisfaire.

Afin de préciser les caractéristiques énumérées dans la deuxième définition (que nous adoptons dans notre mémoire), nous allons présenter les différentes formes d'agents qui ont été catégorisées dans le paradigme multi-agents, avant de définir deux des caractéristiques qui nous paraissent essentielles dans le fonctionnement de ces entités : leur autonomie de décision et leur comportement.

1.1 Typologie des agents

Les agents peuvent être classés en deux catégories selon leur granularité. Cette notion de granularité est très subjective, elle exprime la complexité des fonctionnalités d'un agent afin de séparer les agents "intelligents" des agents moins "intelligents". Les définitions classiques de ces types d'agents sont nombreuses [Sich92], [Ferb95], [Labi93], en voici une synthèse :

- **Agents réactifs :**

> *Ce sont des agents qui réagissent uniquement à leur perception de l'environnement et qui agissent en fonction de cette perception. C'est le niveau de complexité le plus bas des agents. Ce type d'agents ne peut pas vraiment être qualifié d'intelligent, étant donné que leur fonctionnement est principalement basé sur le principe du stimulus/action. Ce principe permet aux agents d'agir grâce à des réflexes conditionnés. Néanmoins, ils présentent des caractéristiques intéressantes* [Drog93], [Lena94] *: simplicité de la description du comportement local et émergence de comportements globaux au delà de la simplicité de l'agent.*

- **Agents cognitifs :**

> *C'est le type d'agents le plus complexe* [Erce91]. *De tels agents sont non seulement capables de percevoir et d'agir sur leur environnement (s'il en utilise un), mais en plus ils ont des capacités de cognition leur permettant de raisonner sur les autres ou sur l'avancement de la résolution. Ils font souvent appel à des modes de communication plus complexes qu'une simple perception. Ils communiquent généralement grâce à des structures de données partagées ou par des communications directes.*

La différence entre ces deux catégories se situe donc non seulement dans la complexité des représentations au sein des agents, mais aussi dans la complexité des informations échangées. En effet, les agents cognitifs échangent des données plus riches que les agents réactifs et sont généralement pourvus de connaissances et de savoir-faire leur permettant des raisonnements plus complexes (inférences, filtrage des informations, accointances,...) pour la gestion de leurs interactions [Trou93], [Chev93]. En général, les agents cognitifs disposent d'une représentation explicite des autres agents et de la manière d'interagir avec eux. Cette représentation est désignée par le terme d'accointance [Riva92]. Nos recherches s'intéressent particulièrement à ces aspects de l'interaction, mais sans pour autant négliger la facilité de description et l'émergence de comportements globaux des agents de type réactif.

2 LE CARACTÈRE AUTONOME DES AGENTS

Un agent peut être qualifié d'autonome par rapport à ses caractéristiques physiques ou conceptuelles. Ces deux aspects de l'autonomie correspondent aux définitions suivantes :

> *Un agent est défini comme autonome physiquement s'il s'agit d'un processus intelligent avec une mémoire privée (savoir et données) et une aptitude à interagir* [Glei94]. *Il est donc une entité active différente et indépendante des autres entités du système pour son exécution.*

> *La notion d'autonomie, d'un point de vue conceptuel, fait référence à l'autonomie de l'agent au sens où il possède ses propres buts, ses motivations ou des capacités pour résoudre seul une (ou plusieurs) tâche(s) précise(s)* [Sich95].

A notre avis, ces définitions sont complémentaires. Nous ne voulons pas proposer une autre définition, mais seulement mettre en valeur deux aspects de la seconde définition qui nous paraissent essentiels [Sich95], [Gass92] :

- **L'autonomie par rapport à ses propres buts :**

  Un agent autonome est un agent qui peut atteindre seul ses buts. Il n'a pas besoin de coopérer avec d'autres agents pour y aboutir, et s'il décide de le faire, c'est parce qu'il espère une amélioration de performance pour réaliser ses buts. Cette notion d'autonomie, de même que la notion de localité de la tâche à exécuter (une tâche est locale si un seul agent est concerné par celle-ci, et elle est globale si plusieurs agents sont concernés), a été utilisée pour classer les comportements possibles d'un agent (cohabitation, collaboration, coopération et distribution) au sein d'une société [Dema90].

- **L'autonomie par rapport à ses motivations :**

  Un agent autonome est un agent qui a la liberté de ses choix pour interagir [Gass92]. C'est à partir de son état interne, de ses buts, de sa représentation du contexte de la résolution qu'il décide de coopérer ou non [Cast90]. Un agent n'est donc pas considéré ici comme une entité forcément bienveillante et asservie par ses relations avec les autres entités du système. Il peut décider par lui même des motifs le poussant à interagir et avoir ses propres préférences.

De ces différents aspects de la notion d'autonomie, nous retenons la capacité des agents à décider par eux-même de leurs choix. Un agent est certes un composant dans un système, mais c'est aussi un composant libre, capable de choisir et de décider par lui-même de ce qu'il juge devoir faire à chaque instant. Nous axons nos travaux sur cette autonomie de décision des agents vis-à-vis de leurs motivations.

3 L'AGENT ET SON COMPORTEMENT

Le comportement d'un agent peut être considéré d'un point de vue externe ou interne à l'agent. Le comportement externe d'un agent correspond à l'observation d'une suite d'actions entreprises par celui-ci, alors que son comportement interne correspond à l'expression de ses capacités de perception, de décision, et d'action. Nous considérons dans ce mémoire le comportement des agents uniquement selon ce point de vue interne.

Pour nous, le comportement interne d'un agent exprime quand et comment un agent va utiliser ses connaissances, ses savoir-faire et ses facultés de perception de l'environnement, ou de communication pour décider de ses actions. Pour un concepteur, la définition du comportement peut alors se résumer ainsi : *"comment assembler les différentes parties d'un agent de manière qu'il accomplisse les actions que l'on attend de lui ? "* [Ferb95]. Il ne s'agit donc pas de décrire

uniquement les buts, les plans, et les actions dont un agent peut disposer et les décisions qu'il peut prendre, mais aussi de décrire comment et quand un agent va communiquer et échanger des informations avec les autres.

Différents travaux ont proposé des modèles intégrant toutes les capacités de l'agent : perception, communication, décision et action [Gass88], [Mart92], [Glei94], [Sich95]. Ces travaux proposent des techniques de raisonnement et de prise de décision (introspection, croyances, intentions, engagements, etc). Ces techniques considèrent généralement qu'un agent possède des connaissances et un savoir et qu'il interagit ou dépend d'autres agents du système pour exploiter ce savoir. Le fonctionnement d'un agent est alors centré sur ses facultés de décision et de raisonnement qui prennent place dans un contexte social d'interactions et de dépendances.

3.1 Exemples de modélisation d'agents

Afin d'illustrer l'assemblage des différentes parties permettant de modéliser le fonctionnement interne d'un agent, nous présentons dans ce paragraphe deux approches de modélisation d'agents : celle proposée par Z. Guessoum dans sa plate-forme DIMA [Gues97], [Gues96] et celle proposée dans LALO [Gauv97].

3.1.1 La modélisation des agents de DIMA

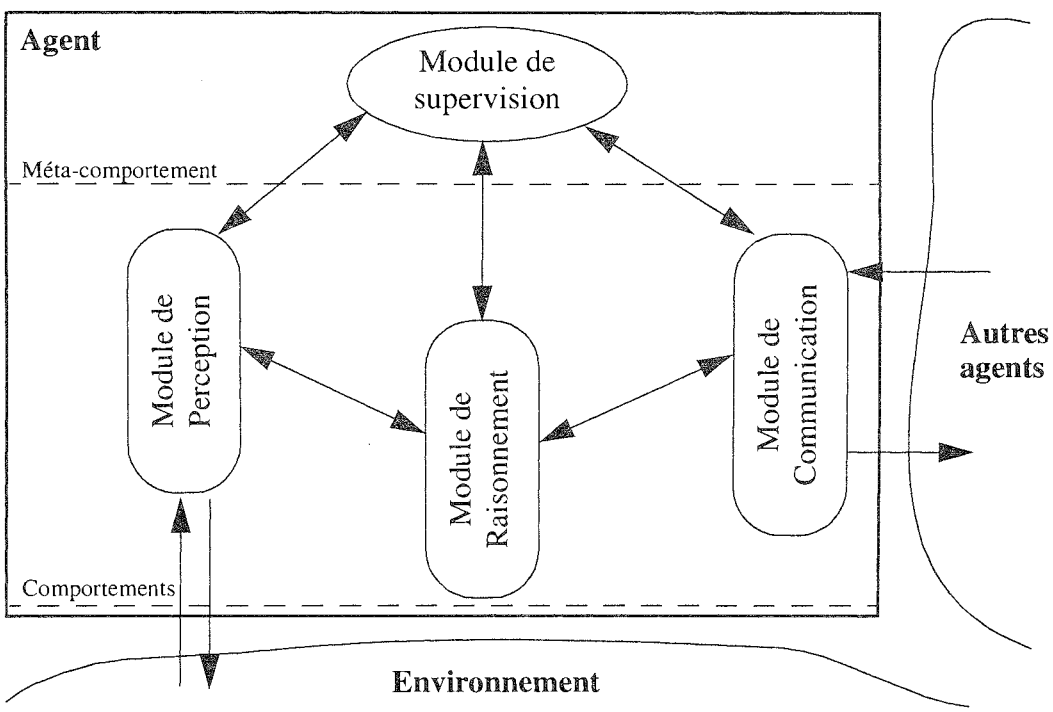


FIGURE 3-1 : Exemple de différents modules d'un modèle d'agent pour DIMA.

Le modèle de DIMA propose une architecture de méta-niveau pour décomposer le comportement d'un agent en une structure de comportements. Cette architecture comprend une première strate composée de modules décrivant les différents comportements d'un agent et une deuxième strate représentant le méta-comportement de l'agent. Elle permet à l'agent de raisonner et d'agir sur ses différents comportements.

Les modules de la première strate correspondent à différents types de comportements : procé-

duraux (stimuli-réponses) et à base de connaissances (cognitifs). Par exemple, les trois exemples de modules présentés dans la figure 3-1 : le module de perception (comportement procédural), le module de raisonnement (comportement à base de connaissances) et le module communication (comportement procédural ou à base de connaissances en utilisant les actes de langages).

Le module de supervision de la deuxième strate définit les mécanismes et les stratégies nécessaires pour gérer les interactions entre les modules de l'autre strate. Il permet à chaque agent d'adapter les comportements à base de connaissances aux comportements procéduraux. Ce module de supervision correspond à une représentation synthétique et déterministe du comportement de l'agent en précisant un séquencement temporel global interne à l'agent au moyen d'états et de transitions. Les états marquent les étapes importantes de la vie d'un agent : ils en représentent une phase temporaire. Ils sont reliés par des transitions qui représentent les changements d'états. Un agent ne change d'état qu'après l'apparition d'un événement (réception d'un nouveau message, perception d'alarme,...).

3.1.2 Les modèles d'agent de LALO

Les modèles d'agent de LALO sont définis par des classes de base pour la construction d'agents dont trois ont été réalisées : BasicAgent, LaLoAgent et RBasedAgent. Chacune de ces classes contient un comportement par défaut de l'agent ainsi qu'une architecture particulière. Par exemple, la classe BasicAgent est basée sur une architecture réactive. La classe LaLoAgent hérite de BasicAgent et représente une architecture hybride dans laquelle l'état mental de l'agent est exprimé par des croyances, des décisions, des engagements et des capacités, mais dans laquelle le comportement est défini de manière réactive. Quand à la classe RBasedAgent, elle utilise la représentation de l'état mental héritée de la classe LaloAgent, mais avec un comportement spécifié avec une base de règles, réalisant ainsi une architecture de type délibérative.

La classe BasicAgent possède les mécanismes de communication dont héritent toutes les autres classes d'agents. Elle n'a pas de représentation explicite du monde et son comportement par défaut est simplement de réagir aux messages reçus ou d'exécuter certaines actions à des temps spécifiés. La structure de cette classe est présentée dans la figure 3-2 selon les deux niveaux qu'elle comporte : communication et comportement.

Le niveau de communication se charge des communications avec les autres agents. Il est composé de quatre modules : réception, expédition, carnet d'adresses et boîte aux lettres. Le module réception traite les messages qui arrive en les plaçant dans la boîte aux lettres. Le module expédition se charge de l'envoi des messages. Le module carnet d'adresses contient les associations entre nom symbolique d'agent et adresse physique. Ces modules correspondent aux mécanismes de communication hérités par toutes les classes d'agents.

Le second niveau réalise le module de comportement réactif de l'agent. Les messages contenus dans la boîte aux lettres sont traités par l'appel de la méthode TraiterMessage : les messages traités sont par défaut uniquement de type transport-address et achieve de KQML. Pour les messages de type transport-address, le module carnet d'adresses est mis à jour. Pour les messages de type achieve, l'agent tentera d'expédier le message inclus si celui-ci est lui-même un message KQML. Ce comportement par défaut peut être changé en créant des sous-classes de la classe BasicAgent où seront redéfinies certaines méthodes de celle-ci.

L'horloge est utilisée afin de déclencher certaines actions à des temps donnés. Il existe deux catégories d'actions : les actions de communication, qui sont communes à tous les agents et les actions locales, spécifiques à chaque agent. L'exécution d'une action du premier type a pour

effet l'envoi d'un message. Les actions du second type constituent le moyen par lequel l'agent interagit avec son environnement. La seule action locale prédéfinie est l'action stop permettant de mettre fin à l'exécution d'un agent.

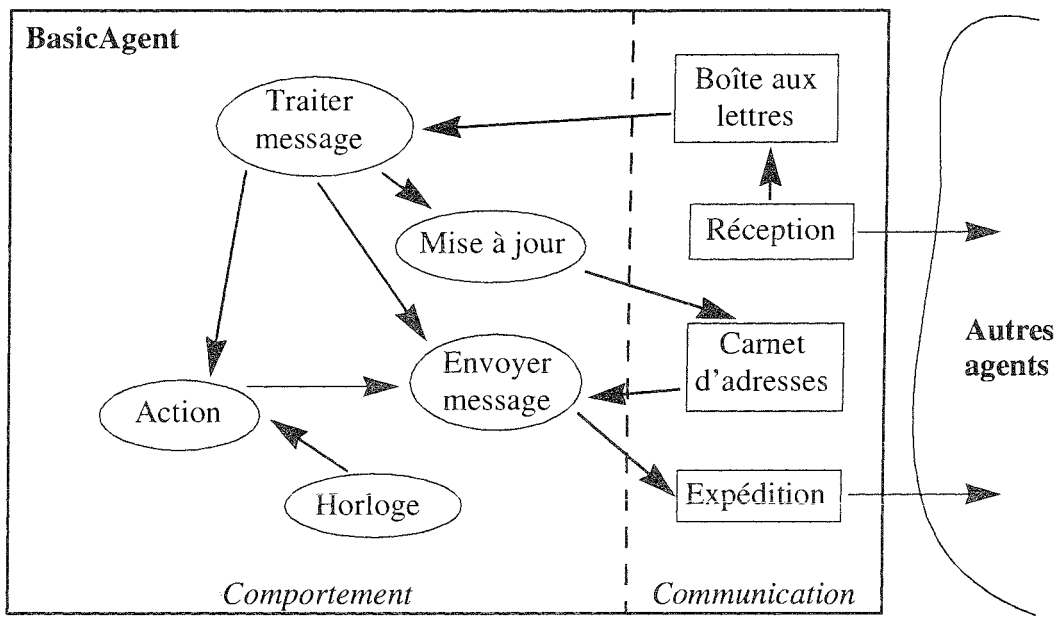


FIGURE 3-2 : Structure de la classe BasicAgent.

3.1.3 Discussion

L'avantage de ces deux approches de la modélisation d'agents est l'utilisation de modèles d'agent hétérogènes au sein d'un même système. Le concepteur peut ainsi choisir ou composer l'architecture de chacun de ses agents en fonction de son application.

Pour DIMA, ce choix se réalise en assemblant des modules de comportements spécifiques (perception, raisonnement,...) et en définissant le méta-comportement de l'agent explicitant les interactions entre les autres modules. L'architecture d'agent proposée est par conséquent modulaire et générique et permet aussi de réutiliser des modèles d'agent dans la bibliothèque de DIMA.

Pour LALO, le choix se réalise à partir d'une bibliothèque de classes correspondant à des modèles d'agent. Ces classes héritent toutes de la classe BasicAgent et l'utilisateur peut créer ses propres modèles. La définition du comportement interne de l'agent n'impose aucune structure, même si des comportements par défaut sont fournis pour les classes de la bibliothèque. Aucune restriction n'est donc imposée quant à la spécification du comportement de l'agent. Le programmeur peut donc choisir la méthode de programmation la plus appropriée en fonction du besoin et de l'application.

Ces deux approches proposent de décomposer chaque agent en différents modules avec un module spécifique qui contrôle l'activation des autres modules. Ils combinent des modules réactifs et des modules cognitifs. Le module spécifique qui gère le contrôle sert à imposer un séquencement temporel interne à l'agent qui définit son comportement.

4 CONCLUSION

Un agent est une entité autonome ayant des facultés de perception, de communication, de décision et d'action. Nous avons montré qu'un aspect important définissant le concept d'agent est son autonomie de décision. Cette autonomie dépend du modèle interne sur lequel repose le fonctionnement des agents (leurs raisonnements et leurs comportements). Elle peut être une autonomie pour la réalisation des buts qui consiste à choisir, en fonction des plans, et des actions relatifs à ces buts, des motifs pour interagir. Mais elle peut aussi être une autonomie de décision vis-à-vis des motivations se préoccupant de permettre aux agents de choisir par eux-même quand et comment interagir avec les autres. Or, si un agent doit interagir avec d'autres agents, c'est généralement parce qu'il dépend d'eux [Sich95] pour réaliser ses buts, il n'est plus complètement autonome, même s'il prend seul ses décisions.

Les interactions entre les agents et la résolution distribuée qu'elles mettent en place induisent alors une contradiction avec ce principe d'autonomie de décision propre au concept d'agent. Les agents peuvent avoir conscience de cette résolution collective (agents cognitifs) ou non (agents réactifs), mais de toute façon, ils utiliseront ou subiront à un moment ou à un autre des contraintes issues de leurs interactions avec les autres agents du système.

Toute la difficulté de concevoir des systèmes multi-agents consiste alors à gérer cette contradiction entre le principe d'autonomie des agents et la résolution collective par le système qu'ils composent. Tous les agents doivent s'intégrer au même système pour former un tout cohérent résolvant le problème à traiter, même si les formalismes qui modélisent les différents agents du système sont hétérogènes. Cette nécessité d'intégration et d'interaction avec les autres fait intervenir des mécanismes et des notions permettant la coordination de la résolution distribuée du problème pour obtenir un comportement global cohérent et efficace du système. Les notions d'interactions et d'organisation permettent d'appréhender cette nécessité et nous allons respectivement les aborder dans les deux chapitres suivants.

Chapitre 4 Les interactions et les agents

La notion d'interaction est une notion qui prend place à l'intérieur des agents, via leur faculté de perception, d'action et de communication, mais qui prend corps entre les agents. Elle peut être définie comme un processus, une dynamique qui consiste en l'enchaînement d'actions réciproques de la part des agents [Bras94]. Ce concept est l'un des aspects les plus importants de l'IAD [Deck87] dans la mesure où il permet de relier les agents qui peuvent échanger des informations et s'influencer afin de constituer un système.

Cette notion d'interaction peut faire intervenir différents modes de communication dont les agents peuvent être dotés. Mais l'interaction ne se résume pas uniquement à une action locale de communication d'un agent vers un autre, c'est surtout un enchaînement d'actions et de communications qu'il faut mettre en place et contrôler entre les agents.

Dans ce chapitre, nous abordons les principes de l'interaction et de sa mise en oeuvre. Nous commençons en présentant le concept de communication sur lequel repose toute interaction. Nous définissons à partir de ce concept de communication, les caractéristiques essentielles de l'interaction dans la section 2. Puis en section 3, nous présentons différentes techniques et modèles, qui nous intéressent plus particulièrement, proposés dans le cadre conversationnel des interactions. Nous terminons en réalisant une synthèse des différents points importants devant être pris en compte dans la mise en place et la modélisation de l'interaction.

1 LA COMMUNICATION

La communication est le concept de base qui permet de relier les agents d'un système pour qu'ils constituent un tout. C'est à ce titre que Decker le considère comme un concept fondamental du multi-agents [Deck87].

Nous adoptons dans ce mémoire la définition suivante pour ce concept :

> *La communication est* ***une action locale d'un agent vers les autres*** *qui peut prendre deux formes : elle peut être indirecte ou directe.*

Dans cette définition, les communications indirectes (ou implicites) supposent l'existence d'un support de communication commun aux agents (un environnement ou une structure de données partagée) qui mémorise et propage (en déformant éventuellement) l'information associée aux signaux déclenchés par la réalisation de l'action. Les communications directes (ou explicites) utilisent comme support de communication des liens directs entre les agents (envoi de message, diffusion totale ou diffusion restreinte) [Chev95], [Ludw93].

1.1 Communication indirecte

Ce type de communication distingue deux techniques différentes selon que le support de communication est passif ou actif vis à vis de l'information :

- **Communication par tableau noir :**

  Cette première technique [Nii86], [Hato91] utilise une structure de données commune aux agents dans laquelle les agents déposent, consultent et modifient des informations. Cet espace est partitionné en différents niveaux d'abstraction du problème et/ou différentes catégories d'information décrivant le problème à résoudre et l'avancée de sa résolution. L'information n'est pas modifiée par le tableau noir. Ce support est dit passif ou neutre vis à vis des informations, bien qu'il signale les événements de modification et de dépôt d'information aux agents. Les agents qui utilisent cette technique sont usuellement appelés *agents spécialistes* [Erce91].

- **Communication via l'environnement :**

  L'environnement est une structure commune aux agents qui possède des lois de comportement. Cet espace dispose généralement d'une métrique et contient un ensemble d'objets O. Ces objets sont généralement passifs, c'est-à-dire qu'ils peuvent être perçus, créés, détruits ou modifiés par l'ensemble des agents (noté A). Ces agents sont des objets particuliers ($A \subseteq O$) qui réagissent à leur perception partielle de l'état de l'environnement et qui combinent leurs actions dans celui-ci avec les lois qui le régissent, afin de modifier son état et ainsi influencer la résolution collective. Par exemple, le passage d'un agent-fourmi [Drog93], [Lena94] va laisser une trace (un objet phéromone) dans l'environnement, qui pourra être perçue par d'autres fourmis pendant un certain temps (loi de persistance) et influer sur leur comportement. Ce support est dit actif, car les lois qui le régissent influent sur les actions des agents qui sont généralement des *agents réactifs*.

Une caractéristique importante de ces techniques de communication est que l'information n'est pas privée, dans le sens où l'agent réalisant l'action de communication ne désigne pas le ou les agents concernés par celle-ci. La structure commune est accessible par tous les agents et les informations sont par conséquent publiques.

1.2 Communication directe

Ce type de communication utilise une technique d'envoi de messages pour transmettre directement les informations. L'utilisation de cette technique dans le domaine des systèmes multi-agents est généralement faite sur la base d'une hypothèse de fiabilité des communications : aucun message n'est perdu ou modifié par le mécanisme d'envoi.

Cette technique de communication peut prendre plusieurs formes :

- **Communication par envoi de messages point à point :**

  Dans ce cas, l'agent émetteur (celui qui réalise l'action de communication) connaît nominativement l'agent destinataire du message et lui adresse alors directement l'information. Cet agent destinataire est le seul à recevoir l'information qui est alors privée.

- **Communication par diffusion généralisée**

  Il s'agit, dans ce cas, d'un envoi simultané d'un même message à tous les agents du système. L'agent émetteur ne connaît pas nécessairement nominativement les destinataires du message, ou du moins ne les désigne pas explicitement lors de l'envoi. L'agent suppose également que parmi tous les agents du système, il en existera bien un capable de traiter ce message.

- **Communication par diffusion restreinte**

  Il s'agit là aussi d'un envoi simultané d'un même message mais seulement à un sous-ensemble des agents du système. L'agent émetteur ne connaît pas nécessairement nominativement tous les agents destinataires, mais il doit être capable de les décrire par des caractéristiques discriminantes, ou de les atteindre grâce à une notion de groupe présente dans le système.

Nous appellerons les agents utilisant ces techniques de communication directe, les *agents communicants* [Erce91], afin d'éviter toutes confusions vis à vis de leurs capacités cognitives, qui peuvent être plus ou moins élevées.

1.3 Conclusion

Le concept de communication permet de relier les agents pour qu'ils constituent un système et c'est à ce titre qu'il est un concept fondamental du paradigme multi-agents [Deck87]. Il permet de réaliser l'échange des informations, la formulation de requêtes au sein du système et par là même, il est le support de l'interaction.

Le problème est alors de combiner ces actions particulières avec les autres actions (compétences, savoir-faire ou services) des agents, afin de coordonner et de contrôler les échanges liant plusieurs agents pour obtenir un comportement collectif entre ces agents.

Cette problématique de combinaison d'actions locales est étudiée au travers de l'interaction, qui constitue par conséquent un niveau d'abstraction supérieur à la notion de communication. De manière générale, une interaction correspond alors non pas à une simple action locale d'un agent vers les autres, mais à un enchaînement d'actions locales entre des agents.

2 L'INTERACTION

Pour un agent, interagir constitue à la fois la source de sa puissance et l'origine de ses problèmes [Ferb95]. En effet, c'est parce qu'ils coopèrent (plus généralement parce qu'ils interagissent) que les agents peuvent accomplir plus que la somme de leurs actions locales, mais c'est aussi à cause de leurs interactions avec les autres agents qu'ils doivent coordonner leurs actions et résoudre des conflits.

Les interactions sont à la base du fonctionnement des systèmes informatiques constitués de plusieurs entités et notamment des systèmes multi-agents et systèmes répartis. La notion d'interaction est néanmoins difficile à cerner, car elle est définie par les entités qui interagissent (leurs objectifs, leurs intentions...), mais elle est aussi liée à l'organisation existant entre les différents composants du système, à l'environnement qui supporte l'interaction, et aux moyens de communication pour la mise en oeuvre des interactions. Plus généralement :

> *L'interaction permet de trouver l'équilibre entre l'autonomie de chaque agent et la coordination des agents vers un objectif commun (donné ou non).*

Selon [Bras96b], le terme d'interaction peut être décliné de deux façons différentes. L'interaction peut être faible ou forte :

- Elle est *faible*, si elle n'est composée que d'un ensemble d'*"action-influence unidirectionnelle"*, c'est-à-dire que les agents agissent *"en direction"* des autres.
- Elle est *forte*, si elle est composée d'un ensemble de *"co-action-influence mutuelle"*.

Le second aspect de cette définition basée sur des actions co-construites est la vision classique de la physique et de la psychologie qui insistent sur l'importance de l'aspect mutuel de ce concept. Brassac et Chevrier insistent alors sur l'intérêt d'envisager des interactions fortes :

> *L'interaction est empreinte de réciprocité ou elle n'a d'"inter" que le nom* [Bras96b].

Malgré l'intelligence faible des agents réactifs, ceux-ci utilisent généralement des interactions fortes où l'enchaînement des actions de communication émerge dans l'environnement [Ferb95], [Lena94]. Le contrôle de cet enchaînement est basé sur des comportements réflexes de type stimuli/actions stipulant le comportement de l'agent. Les agents produisent alors localement des actions dans l'environnement en fonction de leur perception selon des principes réflexes. Dans ce cadre, la problématique de l'interaction consiste à spécifier ces réflexes dans chaque agent pour obtenir un enchaînement viable de ces actions qui émerge de la multitude des agents et des réflexes associés. La description de cet enchaînement est donc réalisée de manière implicite.

Dans le cadre des systèmes composés d'agents communicants, une description de cet enchaînement en terme de perception/action est généralement longue, fastidieuse, et surtout la viabilité des enchaînements est difficile à maîtriser. En effet, pour les agents communicants, une perception ne consiste plus uniquement en un signal dans un environnement, mais à une combinaison de signaux formant un message. La sémantique des communications est alors beaucoup plus riche et complexe. La notion d'interaction reliant les agents s'apparente alors plus à la notion de conversation qu'à une notion d'action.

Les agents communicants ne permettent généralement pas de faire émerger aisément des enchaînements en raison de la complexité des informations véhiculées par les messages et du caractère privé de ces messages. Le principe adopté est plutôt un principe de description explicite de l'enchaînement en utilisant la notion de conversation [Ferb95], [Rous94]. L'enchaînement n'est pas alors uniquement basé sur une continuation locale des messages dans les agents, mais est décrit par des règles conversationnelles [Gric75] et un véritable langage intégrant la syntaxe et la sémantique associée aux signaux constituant les messages [Dema95], [Fini94]. Ceci implique alors une perte d'autonomie des agents puisqu'il leur est imposé des modèles d'enchaînement, ou règles de conversation, qu'ils doivent suivre. Ce type d'interaction peut alors être qualifié d'interaction faible, puisque les règles de conversation sont imposées aux agents. Mais cette approche présente l'avantage de permettre une description plus générique des interactions et par conséquent une réutilisation de la description [Barb95], [Burm95].

Les deux types d'interaction sont néanmoins complémentaires dans le paradigme multi-agents. L'interaction forte permet de conserver le caractère autonome des agents et une grande dynamique dans le système en se basant sur des actions réciproques. Cette approche de l'interaction permet alors aux agents de définir ensemble leurs interactions par un processus en évolution continue [Bour93]. L'interaction faible se base sur une description explicite de l'enchaînement des actions de communication, ce qui permet de mieux maîtriser les phénomènes conversationnels qui sont simulés dans les systèmes. Les modèles, ou règles de conversation, introduits permettent d'envisager une description et une utilisation plus générique de la notion d'interaction [Barb95], [Burm95], c'est-à-dire à un niveau inter-agent et non uniquement par de actions locales à l'agent. Ces modèles constituent alors des éléments de conception permettant la réutilisation et l'évolution des règles de conversation définies dans un système applicatif pour une autre application.

Nous étudions ici essentiellement l'interaction dans le cadre des agents communicants en nous focalisant sur la description de l'enchaînement des communications, mais en envisageant également une dynamique dans l'utilisation et la mise en place des interactions. Nous adopterons la définition suivante de l'interaction dans ce mémoire :

> *Une interaction est un processus, dynamique, basé sur un ensemble d'actions entreprises localement par les agent, qui nécessite un contrôle et une coordination de leur enchaînement entre les agents.*

L'interaction est alors envisagée au niveau des agents comme un processus local et autonome, mais ce processus est régi par des règles de conversation permettant un contrôle à un niveau inter-agent. La problématique essentielle de l'interaction est à notre avis d'envisager l'expression de phénomènes interactionnels, lors de la conception des systèmes, entre et dans les agents, tout en préservant l'autonomie de ceux-ci.

Nous allons poursuivre cette étude de l'interaction en présentant les modèles et techniques généralement utilisés dans le cadre des communications directes pour contrôler les interactions.

3 TECHNIQUES ET MODÈLES D'INTERACTION

Les modèles et techniques utilisés pour gérer l'interaction des agents dans le cadre des systè-

mes composés d'agents communicants font généralement appel aux notions de langage d'interaction et de protocole d'interaction. Nous allons présenter ces deux notions et montrer à quel niveau elles prennent place dans la conception des systèmes multi-agents [Durf87], [Deck89], [Ferb95] : au niveau du système (global), au niveau de l'agent (local), ou à un niveau inter-agent (semi-local) pour contrôler l'enchaînement des actions locales.

3.1 Le langage d'interaction

En général, les approches classiques [Durf87], [Bond88], [Dema95] du paradigme multi-agents considèrent que la notion de langage d'interaction est nécessaire à un niveau global pour que les agents se comprennent entre eux. Ce postulat est généralement appliqué par la mise en place d'un langage commun à l'ensemble des agents, définissant pour le système entier une syntaxe et une sémantique des échanges.

Cette définition d'un langage d'interaction a conduit les chercheurs à faire appel à une théorie qui semble être une bonne candidate à la modélisation des échanges entre agents : la théorie des actes de langage [Sear72], [Sear79], [Sear85], [Vand88], [Bras92]. De par sa nature, il est naturel de considérer la communication, comme un acte lié au langage. Nous allons présenter brièvement cette théorie, puis son utilisation dans le paradigme multi-agents.

3.1.1 Les actes de langage

La théorie des actes de langage est basée sur l'étude des dialogues humains [Sear72], [Sear79]. Un agent qui utilise un langage agit et produit de ce fait une action vers les autres agents. Cette action a une composante propositionnelle (le contenu propositionnel) qui spécifie l'action et tente de la réaliser, et une composante intentionnelle (la force illocutoire) appliquée à cette action par le locuteur. La réunion de ces deux composantes forme un acte de langage ou acte illocutoire.

La force illocutoire peut être exprimée grâce à un verbe qualifié de performatif. La théorie des actes de langage distingue différentes catégories dans l'ensemble des verbes performatifs d'une langue : les assertifs (informer), les directifs (demander), les commissifs[1] (proposer), les déclaratifs (déclarer) et les expressifs (exprimer) [Bras96a]. L'exemple simple des deux phrases utilisées par J. A. Campbell et M. P. D'Inverno [Camp90] illustre cette notion de force illocutoire. Les deux phrases "Ferme la porte" et "Tu vas fermer la porte" ont le même contenu propositionnel, mais le premier à la force illocutoire d'un ordre (un directif particulier) et l'autre d'une prédiction (un assertif particulier).

L'hypothèse sur laquelle repose l'étude de J.R. Searle est que parler une langue c'est accomplir des actes respectant des règles [Sear72]. L'accomplissement d'un acte est réussi sous certaines conditions (appelées conditions de succès de l'acte de langage). La logique illocutoire est une axiomatisation de la théorie de l'accomplissement, réussi ou non, des actes de langage réalisés à l'aide de la profération d'énoncés en contexte par un locuteur [Bras96a]. Il s'agit d'une logique des conditions de succès des actes de langages [Sear85].

La théorie des actes de langage, dans sa version la plus récente, intègre ces conditions de succès et des conditions de satisfaction [Vand88]. Ces dernières prennent en compte les aspects perlocutoires des actes théorisés par Vanderveken. Ces aspects ne sont pas codés dans les actes de communication et dépendent du contexte dans lequel s'effectue la communication. Il s'agit par

1. Aussi désignés par le terme promissifs

exemple de l'effet de persuasion, qui par exemple dans une salle des ventes associe à l'affirmation "c'est une oeuvre rare", la tentative de persuader les acheteurs potentiels d'acquérir l'oeuvre. L'intention du locuteur intervient dans l'échange de manière implicite.

Une autre logique semble prometteuse, la logique interlocutoire [Bras92] qui est définie par l'auteur comme une dialogisation de la logique des conditions de succès et des conditions de satisfaction des actes de langage. Celle-ci se base sur l'analyse des enchaînements conversationnels qui révèle que la profération d'un énoncé en contexte ne supporte pas une unique intention de son locuteur. L'intention se négocie, se co-construit au long du développement temporel de l'interaction.

3.1.2 Les techniques multi-agents et les actes de langage

Les actes de langage ont inspirés différents travaux en IAD [Cohe90c], [Cohe90a], [Fini94], [Dema95], [Rous94]. Nous n'allons pas tous les présenter en détail, mais seulement les décrire brièvement et commenter [Cohe95], [Bras96a], [Ferb95] leurs choix vis-à-vis de l'utilisation des actes de langage. Dans un premier temps, nous présentons les travaux de Cohen et Levesque, ainsi que KQML [Fini94], avant de présenter les approches de Demazeau [Dema95] et Rousseau [Rous94].

Les travaux de Cohen et Levesque sont basés sur le concept d'intention dans la communication [Cohe90b]. Pour les auteurs, la communication dépend de la capacité des agents à reconnaître les intentions et les plans des autres agents, mais ils n'utilisent pas la force illocutoire des actes. Dans ce cadre, ils élaborent un modèle de l'intention fondé sur une théorie de l'interaction rationnelle entre agents et déploient une logique permettant à la fois de décrire et de raisonner sur les états mentaux des agents à partir de quatre opérateurs modaux : la croyance, les buts, et deux opérateurs temporels pour ce qui vient de et ce qui va se produire. Cette logique est basée sur des inférences contextuelles sur les états mentaux.

KQML (Knowledge Query and Manipulation Language) [Fini94] est une approche basée sur les actes de langage visant à résoudre l'interopération entre des agents en prenant en compte une diversité des langages de communication. La communication est considérée comme un ensemble d'échanges de messages de type déclaratif tels que les assertions. Une expression KQML est composée d'un message interne spécifié en KIF (Knowledge Interchange Format) sur lequel s'applique un verbe performatif [Fini94], [Mayf96]. La force illocutoire appliquée au contenu du message est prise en compte dans cette approche. Néanmoins, KQML n'adopte que les performatifs de type assertif et directif et certains des performatifs utilisés ont une signification floue et ambiguë, c'est le cas du performatif "dénier" comme le montre Cohen et Levesque [Cohe95]. Ferber critique aussi KQML [Ferb95], lui reprochant entre autre qu'il ne prend pas en compte les promissifs. Il n'est alors pas possible en KQML de dire que l'on s'engage, auprès d'un tiers, à accomplir une action.

L'approche de MAGMA [Dema95] est basée sur une formalisation des messages reposant sur un langage d'interaction inspiré de la théorie des actes de langage. La capacité d'interaction repose sur l'implémentation de protocoles d'interaction plus ou moins complexes, qui sont associés au langage d'interaction. Le protocole d'interaction est un graphe d'états prédéterminés dans lequel l'agent évolue en fonction des messages qu'il reçoit [Popu93]. Il conditionne le comportement de l'agent en le renseignant sur le type de message qu'il doit émettre à son tour. Dans cette approche, il faut noter que les protocoles utilisés sont transmis dans les messages pour éviter tout indéterminisme local aux agents. Les agents ne sont donc pas autonomes pour l'interprétation d'un message puisqu'ils sont soumis au protocole transmis et aux règles compor-

tementales prévues pour les quatre types pouvant caractériser un message : *present*, *request*, *answer* et *inform*. Ces règles sont fournies par le langage d'interaction et lèvent toute ambiguïté vis-à-vis de la reconnaissance des intentions dans les actes de communication.

Les travaux de Rousseau, Moulin et Lapalme [Rous94] considèrent les actes de discours comme des événements dépendants les uns des autres. Ils considèrent alors l'existence de structures conversationnelles qui prennent une forme prédéterminée par les usages et conventions de la société. Ils introduisent la notion d'objets conversationnels qui peuvent correspondre à des intentions, des croyances, mais aussi des faits, des plans, des règles, des émotions, etc. Ils proposent de modéliser les conversations par l'échange d'objets conversationnels qui sont consultés et mis à jour par les agents participants à une conversation par l'intermédiaire d'un agent spécialiste : l'agent de conversation. Cette solution, si elle permet de placer l'interaction à un niveau inter-agent pose le problème d'introduire un goulot d'étranglement, car un agent gère toutes les conversations prenant corps dans le système.

Les deux premières approches respectent partiellement les bases de la théorie des actes de langage [Sear72], [Sear79]. Elles considèrent toutes deux que l'interaction est basée sur des actions empreintes d'intentions, ce qui justifie l'utilisation d'une logique pour raisonner sur les intentions (explicites ou implicites des actions de communication) localement au niveau de l'agent. Le problème, comme le soulève Ferber, est alors que :

> *"Tout standard de communication, qui repose sur la manière dont un agent se comporte, limite les possibilités d'intégration d'agents véritablement hétérogènes. Il apparaît donc qu'une spécification des communications, en termes de protocoles, présente l'avantage de pouvoir faire abstraction de la nature propre des agents en se focalisant sur les relations qui existent entre les communications" (page 360* [Ferb95]*).*

Les deux dernières approches vont dans ce sens et se focalisent plutôt sur la problématique de l'enchaînement en utilisant ou non les catégories de verbes performatifs de la théorie. La logique et la sémantique des actes sont alors incluses dans la problématique de l'enchaînement par la notion de protocole.

Tous ces travaux soulèvent encore le problème de l'enchaînement des actes durant l'interaction entre les individus. Brassac insiste sur une sémantique et un enchaînement co-construit par deux agents pendant leur dialogue [Bras96a], mais comment contrôler la cohérence de l'enchaînement localement au niveau des agents? Ce problème n'est pas complètement résolu par la théorie des actes de langage de Brassac et Trognon :

> *"L'enchaînement conversationnel est imprévisible, s'opère de façon rétroactive, par chaînage arrière, est marqué par une indécidabilité fondamentale et est le fait des deux conversants qui agissent de façon collaborative et qui négocient continûment le sens des énoncés"* [Bras96b].

A partir de cette citation, il faut peut-être se demander si l'utilisation des actes de langages, qui sont empreints des défauts propres à l'homme, doit être nécessairement reproduite en l'état par des machines? [Bras96a]. Nous avons alors choisi d'envisager l'interaction à partir de protocoles d'interaction [Dema95], [Popu93], [Rous94] précisant les usages et les conventions des interactions. En sachant qu'un problème demeure dans l'utilisation des protocoles d'interaction, ils aboutissent généralement toujours sur des interactions faibles, qui privent les systèmes de dynamique.

3.2 Protocoles d'interactions

L'expression de l'enchaînement d'actions de communication et son utilisation sont des problèmes classiques étudiés dans le domaine des systèmes répartis [Juan95]. Nous allons commencer par présenter la notion de protocole des systèmes répartis, avant de préciser la notion de protocole d'interaction utilisée dans le cadre des systèmes multi-agents.

Dans les systèmes répartis, les protocoles décrivent des enchaînement d'actions à partir des formalismes des réseaux de Pétri et des automates d'états finis. Le problème étudié vis à vis de cet enchaînement est lié à la validation et à la fiabilité des protocoles physiques implantés. L'approche est fortement liée au support physique de la communication sur lequel aucune hypothèse de fiabilité de la transmission n'est généralement faite. Le but est alors de décrire l'enchaînement, mais surtout de valider celui-ci généralement dans le cadre du modèle standard ISO/OSI (International Standard Organization's Open System Interconnect).

Les techniques de base (réseaux de Pétri ou automates d'états finis) employées dans ce domaine ont été reprises pour les systèmes multi-agents dans de nombreux travaux sur les protocoles d'interactions [Gasp91], [Burm95], [Camp90], [Dema95]. Signalons néanmoins que la notion de protocole utilisée dans les systèmes multi-agents n'est pas du même niveau conceptuel que celle des systèmes répartis. Dans les systèmes multi-agents, les protocoles d'interaction sont plutôt liés à la gestion des communications en posant l'hypothèse que les communications sont fiables. L'approche multi-agents étudie, entre autre, la description générique et la gestion en simultané de ces protocoles d'interaction [Burm95], [Bela96] en tant que connaissances. Les protocoles d'interactions forment un niveau de connaissance et non uniquement un niveau de routage des communications.

Par exemple, l'approche du groupe MAGMA [Dema95] est de décrire des protocoles d'interaction comme des connaissances fournies aux agents d'un système. L'ensemble de ces connaissances définit alors un langage d'interaction pour le système. Nous allons brièvement présenter cette approche afin d'illustrer notre propos.

3.2.1 L'approche de MAGMA

Le groupe MAGMA propose un langage de description de protocole d'interaction, qui correspond à un enchaînement d'actions de communications. Ces actions de communication sont décrites selon la grammaire suivante [Dema95] :

<interaction> = <communication> <représentation de connaissance>

Le champ <communication> décrit les différents paramètres utilisés pour le routage des messages selon le point de vue des systèmes distribués. Ces paramètres sont : le receveur <à> et l'expéditeur du message <de> (le contenu de ces champs sont des identifiants d'agents ou le mot-clé broadcast pour une diffusion), l'identifiant du message <id>, le type de canal de communication utilisé <via> (transfert de données à haut débit, passage de message par boîte aux lettres, ...), et le mode de communication <mode> (qui peut être synchrone ou asynchrone).

<communication> = <de> <à> <id> <via> <mode>

Le champ <représentation de connaissance> décrit le routage des messages selon la connaissance du domaine multi-agents (qui inclut l'intention de l'expéditeur et ses espérances) et le domaine d'application (qui est une décomposition du contenu du message spécifique à chaque domaine d'application).

<représentation de connaissance> = <multi-agents><application>

La connaissance du domaine multi-agents est exprimée selon cinq champs qui sont : le <type> (exprimé selon quatre types primitifs : *present*, *request*, *answer* et *inform*), la <force> illocutoire (qui définit la priorité de l'acte de communication pour le receveur) et la <nature> du message (qui peut être des échanges d'hypothèses, de buts, de plans ou d'actions choisies), mais aussi le <protocole> qui correspond au contexte dans lequel l'acte de communication est émis et la <position> dans laquelle le receveur est quand il reçoit le message.

<multi-agents> = <type> <force> <nature> <protocole> <position>

Un protocole d'interaction correspond alors à restreindre les différentes actions de communication qui peuvent lier un agent à d'autres pour résoudre un problème donné. Il permet donc de structurer les communications pour définir une interaction et correspond à un réseau de transitions qui définissent, a priori, l'ensemble des transitions possibles qui relient un ensemble d'états que l'agent peut occuper alternativement selon les effets des actions de communication échangées. Les transitions sont représentées par le langage de description présenté pour les actions de communication et l'état source d'une transition est représenté par la <position>. Un exemple de protocole, qui peut être ainsi décrit, est celui des réseaux de contrat [Smit80].

L'exécution d'une interaction entre des agents correspond alors à l'exécution d'un protocole d'interaction. Pour cela, un agent, qui reçoit un message, doit être capable de décoder l'interaction à laquelle se rapporte ce message en utilisant différents critères : l'expéditeur, l'intention de celui-ci, et ses espérances. A partir de ces éléments, un agent est capable d'évaluer la situation et réagit à celle-ci en respectant les actions de communication possibles qui sont décrites par le protocole d'interaction.

3.2.2 Discussion

Dans le cadre des systèmes multi-agents, le concept d'interaction fait donc intervenir différents niveaux pour la description et l'opérationnalisation des interactions. Nous allons adopter différents niveaux terminologiques pour manipuler ce concept :

A un niveau descriptif, nous parlerons de **protocole d'interaction** pour désigner la connaissance semi-globale représentant un plan à un niveau inter-agent qui permet à chacun des agents concernés de savoir comment et avec qui interagir. En outre, cette connaissance contient explicitement la syntaxe et la sémantique des échanges qui est définie par le langage d'interaction.

L'utilisation de ce type de connaissance est alors de fournir aux agents différents schémas pour interagir. Nous parlerons de **schémas de conversation** pour désigner la représentation d'un protocole en terme de connaissance locale à un agent.

Enfin, nous appellerons **interaction** l'instanciation et l'utilisation d'un tel schéma par un agent pour réaliser les actions ainsi décrites.

4 CONCLUSION

Ce chapitre vient de présenter les principes et concepts de base permettant de mettre en place l'échange d'information, d'induire des comportements spécifiques et donc de donner aux agents

la possibilité d'interagir.

La communication qui constitue un concept fondamental du multi-agents [Deck87] est la brique de base de cette notion d'interaction. Nous avons montré que cette communication pouvait prendre différentes formes, mais que quelle que soit la forme prise, il s'agit d'actions locales fiables entreprises par les agents.

La notion d'interaction englobe cette notion de communication, une interaction combinant plusieurs actions de communication. Cette combinaison d'actions pose alors un problème qui est de prendre en compte le caractère autonome des agents en leur permettant de co-construire leurs interactions et de contrôler cette co-construction.

Pour prendre en compte l'autonomie des agents et permettre une description aisée de l'enchaînement des actions de communication, l'interaction doit alors être mise en place à partir de deux niveaux : à un niveau local, qui est le niveau de base permettant la mise en place des interactions dans les agents à partir de communication, et un niveau inter-agent permettant une description plus aisée.

Le premier niveau est celui qui assure l'autonomie des agents pour interagir en leur conférant la possibilité de choisir par eux-même les actions de communications à entreprendre. Ce niveau local existe dans les différentes techniques multi-agents que nous avons présentées selon des modèles ou des logiques plus ou moins complexes.

Le second niveau fait abstraction du comportement et de l'autonomie des agents. Il permet de spécifier les communications en se focalisant sur les relations qui existent entre elles. Il prend corps explicitement au travers de l'agent de conversation pour Rousseau [Rous94], ou bien il est un niveau de référence pour la conception des interactions du système grâce aux protocoles d'interactions [Dema95], [Ferb95]. Ce niveau peut être qualifié de semi-global, car il relie des sous-ensembles d'agents du système en décrivant l'enchaînement des actions lors des interactions.

Selon ces deux niveaux, l'interaction intervient à la fois entre et dans les agents d'un système. Entre les agents, elle prend la forme de protocoles d'interaction qui concernent potentiellement tous les agents du système. Dans les agents, cette description prend place sous la forme de schémas de conversation qui permettent de contrôler les interactions à un niveau inter-agent à partir de l'agent.

Néanmoins, les techniques multi-agents considèrent généralement explicitement ou implicitement que l'interaction nécessite l'adoption d'un langage d'interactions global, afin d'assurer une compréhension mutuelle des agents composants le système [Dema95], [Smit80]. Cette nécessité ajoute alors un troisième niveau, global, dans la conception des interactions.

Le problème est alors de concevoir des interactions et de les utiliser selon ces trois niveaux. Les niveaux global et semi-global interviennent dans la description des phénomènes interactionnels au niveau du système, alors que le niveau local correspond à la mise en place de ces phénomènes et à leur gestion par les agents.

Nous avons montré la nécessité du contrôle et de la coordination pour permettre aux agents d'interagir. Mais, la multitude d'interactions potentielles dans un système, si elle permet la résolution du problème abordé, amène également des situations conflictuelles (conflit d'actions, accès à une ressource, etc) et des situations où des actions redondantes sont entreprises par les agents. Il devient alors nécessaire de structurer et de coordonner l'ensemble des interactions des

agents d'un système pour maintenir la cohérence du tout durant la résolution.

La notion d'interaction est donc fortement liée à la notion d'organisation, car ce sont les possibilités d'échanges et d'influences que les interactions créent entre les agents qui sont les bases permettant à un système de s'organiser et de résoudre un problème. Nous allons étudier cet aspect dans le chapitre suivant.

Chapitre 5 L'organisation et les interactions

L'approche multi-agents est telle que la résolution d'un problème se situe au-delà de la complexité de chacun des agents du système et réside dans la multitude des interactions entre les constituants d'un système.

Cette multitude d'interactions, si elle permet la résolution de problèmes complexes, amène également des situations conflictuelles (conflit d'actions, accès à une ressource, etc) et des situations où des actions redondantes sont entreprises par les agents. Il devient alors nécessaire de coordonner les interactions existant entre les agents, afin de structurer leurs activités.

Ce problème est abordé au travers de la notion d'organisation qui est un autre aspect fondamental du paradigme multi-agents. Cette notion se préoccupe de définir, de gérer et de modifier les relations entre les individus. Elle correspond donc à un concept liant la description d'une société d'agents et celle des activités de contrôle et de coordination. Cette description se fait en termes de définition, de rôles, d'assignation de tâches, de comportements, de relations d'autorité, etc [Gass92]. Elle permet de contrôler l'ensemble des agents pour qu'ils résolvent ensemble un problème, ceci en restreignant les comportements individuels pour limiter les conflits potentiels et pour gérer le problème de cohérence globale dans le système. Dans notre travail, nous considérons qu'une relation se concrétise par une interaction.

Nous débutons ce chapitre en présentant diverses définitions possibles pour l'organisation, afin de situer la problématique de cette notion dans la section 1.1. Dans la section 1.2, nous présentons l'organisation sous son aspect statique (la structure organisationnelle), puis dans la section 1.3, nous abordons son aspect dynamique (le processus qui permet d'aboutir à cette structure). Enfin dans la section 2, nous détaillons les rapports existant entre la structure organisationnelle et le processus d'organisation dans le cadre de systèmes composés d'agents de type communicants, avant de terminer ce chapitre par une synthèse des concepts abordés.

1 L'ORGANISATION

1.1 Quelques définitions

Le concept d'organisation est un thème majeur du paradigme multi-agents. Il a fait l'objet de nombreux travaux [Stru95], [Chev93], [Bour93], [Trou94], [Carl94], [Sich95], [MARC96], [Baei96]. Ces travaux se sont souvent inspirés des ouvrages de sociologie [Dupu92], psychosociologie [Peti84], économie [Mint79], éthologie [Lena94] ou encore sciences cognitives.

Ces disciplines fournissent de nombreuses définitions de l'organisation. Par exemple, la sociologie ou l'économie propose les définitions générales suivantes :

> *Les organisations doivent être vues plutôt comme des coalitions dynamiques d'intérêts divergents et souvent conflictuels que comme des systèmes de collaboration parfaite* [Schm91].

> *"L'organisation peut être définie comme un agencement de relations entre composants ou individus qui produit une unité, ou système, dotée de qualités inconnues au niveau des composants ou individus. L'organisation lie de façon inter-relationnelle des éléments ou événements ou individus divers qui dès lors deviennent les composants d'un tout. Elle assure solidarité et solidité relative, donc assure au système une certaine possibilité de durée en dépit de perturbations aléatoires"* [Mori77].

> *La conception d'une structure d'organisation est fondée sur la conception de paramètres essentiels qui influent la division du travail et les mécanismes de coordination, affectant ainsi le fonctionnement de l'organisation, l'autorité, l'information véhiculée et la prise de décision* [Mint79].

Le domaine des systèmes multi-agents, en cherchant à mettre en oeuvre ce concept, a donné naissance à plusieurs définitions similaires. Nous n'en citons que deux, sur lesquelles nous nous basons :

> *M.S. Fox définit une organisation comme un gabarit ("pattern") décrivant comment les membres de l'organisation sont en relation et interagissent afin d'atteindre un but commun* [Fox81].

> *"les organisations constituent à la fois le support et la manière dont se passent les interrelations.... c'est ce qui rend ce terme si difficile à cerner, l'organisation désigne à la fois le processus d'élaboration d'une structure et le résultat même de ce processus. Cette dualité du terme montre qu'il n'existe, à proprement parler, pas d'organisations statiques. L'organisation est nécessairement dynamique et toujours en réorganisation de l'ensemble des entités et des liens qui unissent ces entités" (page 98* [Ferb95]*).*

Après ce bref aperçu des différentes définitions existantes, il apparaît difficile de donner une définition unanime de ce concept. Néanmoins l'analyse de Ferber sur les organisations montre qu'il existe une dualité du terme entre l'aspect statique et l'aspect dynamique. Ce dernier distingue la structure organisationnelle pour désigner l'invariant momentané observable sur le système et le terme "organisation" correspondant au processus de constitution d'une telle structure. Nous adoptons cette dernière définition et l'étudions plus en détail, afin de mieux comprendre

les mécanismes essentiels de l'organisation. Nous poursuivons tout d'abord la présentation du concept d'organisation selon l'aspect statique de Ferber. Puis, nous présentons l'aspect dynamique et les différentes approches possibles de celui-ci.

1.2 La structure organisationnelle

Une structure organisationnelle est un gabarit [Fox81] décrivant les relations entre les membres d'un système afin d'atteindre un but commun. Cette structure correspond aussi d'un point de vue sociologique [Dupu92] et éthologique [Lena94] à un schéma qui peut être observé sur un système. La structure d'organisation peut donc être vue comme la description d'un gabarit ou son observation.

Qu'elle soit décrite ou observée, cette structure traduit la mise en place de contraintes sociales qui restreignent l'activité individuelle pour assurer une cohérence globale de la résolution du système. Cette structuration correspond à un ensemble de relations entre les agents de la société et à un ensemble de rôles affectés à ces agents. Elle décrit ou traduit le fonctionnement du système en terme d'activités et de relations dans le système grâce à la définition de rôles, de comportements, de relations d'autorités, etc [Gass92].

L'observation d'une structuration impose que celle-ci soit adoptée par les agents composants le système. Il faut alors que chaque agent dispose de connaissances (partielles ou totales) relatives à cette structure : on parle alors de connaissances relationnelles ou encore d'accointances. La structuration de l'activité de la société se traduit par l'observation des interactions liant les agents du système durant une période d'observation. L'observation d'une structure ne correspond donc pas forcément à une représentation globale et concrète de celle-ci dans le système ou dans les agents, mais est fonction des relations ou, plus concrètement, des interactions utilisées par les agents composant le système durant leurs activités.

Les agents d'un système adoptent une structuration en fonction de leurs connaissances individuelles des interactions qui caractérisent cette structuration. L'organisation d'un système consiste alors à décrire ces connaissances d'une manière globale et à doter les agents de ces connaissances [Durf87]. Cette description correspond à l'adoption de rôles et de conventions qui conduisent à un comportement caractéristique de la structure en question.

L'adoption de rôles et de conventions est la base de toute structure organisationnelle [Ferb95] :

> *"Les rôles décrivent la position des agents au sein d'une organisation ainsi que l'ensemble des activités qu'ils sont censés exercer afin que l'organisation puisse accomplir ses objectifs. Ils caractérisent ainsi les fonctions que les agents remplissent, indépendamment de leur structure interne et des mécanismes mis en oeuvre."*

Les rôles permettent ainsi d'exprimer une structure organisationnelle qui caractérise, sur un plan abstrait, une classe d'organisation, d'après Ferber [Ferb95]. Il définit alors l'organisation concrète (nous parlerons plutôt d'une structure concrète) comme une instanciation (au sens des langages objets) possible d'une structure organisationnelle. Une telle structure organisationnelle est définie par la double donnée :

- d'un ensemble d'agents caractérisés par les rôles (ou fonctions) qui leur sont affectés,
- de l'ensemble des relations abstraites existant entre ces rôles, pouvant définir des modes

de coopération variés [Monc94], ou conventions.

Cette définition s'illustre encore dans le modèle AALAADIN [Gutk98] qui se base sur les trois concepts centraux d'agent, de groupe et de rôle. La figure 5-1 présente un diagramme de ce modèle où l'agent est simplement décrit comme une entité autonome communicante qui joue des rôles au sein de différents groupes. Le groupe est alors défini comme la notion primitive de regroupement d'agents. Chaque agent peut être membre d'un ou plusieurs groupes. Un groupe est une instance de structure de groupe, cette structure étant une description abstraite d'un groupe. Une structure de groupe est définie comme un ensemble fini d'identifiants de rôles et un graphe des interactions dans ce groupe. Les interactions internes à un groupe sont donc définies par rapport à la notion de rôle, qui est une représentation abstraite d'une fonction, d'un service ou d'une identification d'un agent au sein d'un groupe particulier. Chaque agent peut avoir plusieurs rôles, un même rôle peut être tenu par plusieurs agents, et les rôles sont locaux aux groupes.

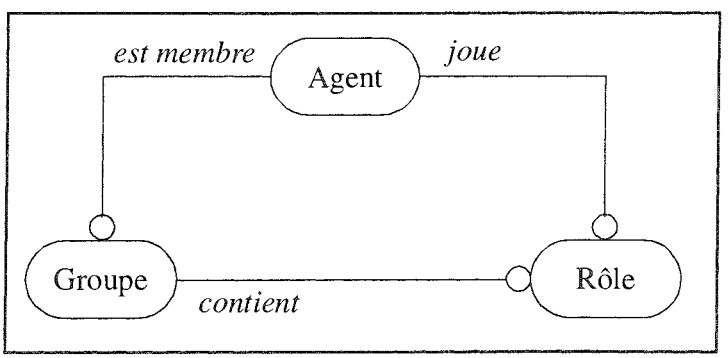


FIGURE 5-1 : Les trois concepts centraux d'AALAADIN.

Ces notions de rôles et de conventions (associées à la notion de groupe pour Gutknecht et Ferber) servent alors à exprimer les différentes phases de négociation ou de coordination que doivent mettre en oeuvre les agents pour suivre le gabarit d'un ensemble relations. Les rôles joués par les agents ne sont pas nécessairement fixes, mais peuvent évoluer et ainsi enchaîner d'autres relations (une autre structure concrète) de la structure organisationnelle [Quin92].

Les agents s'organisent en adoptant des rôles, et en influençant les autres agents pour qu'ils jouent d'autres rôles, afin de participer ensemble à la résolution du problème selon les conventions définies pour les relations potentielles. Ces rôles et ces conventions régissent le comportement des individus, en spécifiant la réalisation de leurs relations afin de contrôler le comportement du collectif.

1.3 Le processus d'organisation

Le processus d'organisation est la deuxième composante d'une organisation. Il correspond à un savoir permettant d'aboutir à une structure organisationnelle ou permettant de changer de structure organisationnelle.

Selon la perspective multi-agents, le processus d'organisation d'un système varie en fonction du type d'agents qui le composent [Stru95]. Les agents de type cognitif ou communicant composent généralement des organisations à structure imposée dès la conception du système et relativement stables (au sens de la théorie des systèmes utilisée en physique). Des ensembles d'agents de type réactif forment diverses organisations dites émergentes (on parle aussi dans ce cas

d'auto-organisation des agents).

Dans le cadre des agents communicants, la notion d'organisation généralement employée limite l'autonomie d'action et de décision des agents. Elle agit comme une contrainte, invariante et extérieure, imposée [Stru95] à l'agent qui lui indique avec qui interagir, dans quel contexte et selon quel mode. De ce fait, lorsqu'un agent est amené à interagir, il utilise uniquement une structure organisationnelle qui lui a été imposée au départ et qui précise avec quel(s) autre(s) agent(s) il doit le faire, lui évitant ainsi de chercher et décider par lui même. L'emploi d'une structure d'organisation imposée restreint alors le nombre de conflits, et coordonne l'activité de la société, mais au détriment de l'autonomie des agents et du processus d'organisation qui est alors inexistant dans le système.

Le modèle du réseau de contrat [Smit80], [Davi81] adopte une gestion plus dynamique de l'organisation en construisant des relations hiérarchiques [Camp95]. La structure organisationnelle n'est donc pas imposée aux agents, mais construite par eux, grâce à des diffusions d'appels d'offres et à des mécanismes de raisonnement sur leur intérêt ou sur leur qualification pour une tâche. Un agent n'a pas de connaissance a priori lui permettant de décider par lui-même avec quel(s) autre(s) agent(s) il doit interagir. Pour lui, chaque agent du système forme un candidat potentiellement satisfaisant qu'il doit solliciter pour pouvoir prendre une décision puisqu'il n'a aucune connaissance sur l'intérêt des autres agents. Le mode de construction de l'organisation est alors imposé au départ. Les agents ont un seul moyen de coordonner leurs tâches et de s'organiser : les diffusions globales (ou partielles). L'application de ce modèle sur des systèmes comprenant de nombreux agents peut par conséquent engendrer un grand flux de messages pénalisant l'efficacité de la résolution.

Une autre gestion possible de la dynamique d'organisation a été proposée par Sichman [Sich95]. Elle se base sur l'utilisation de trois protocoles d'interactions types : un protocole de présentation d'un agent entrant dans la société, un protocole de sortie et un protocole de formation de coalitions. Ce dernier protocole permet de gérer une organisation dynamique qui est proposée par un agent en fonction de ses buts, plans et actions propres. Cette proposition peut alors provoquer une révision de croyances, être refusée, ou acceptée (dans ce dernier cas un quatrième protocole d'interaction servirait à coordonner l'action conjointe des agents réunis dans la coalition) par les autres agents. Les deux premiers protocoles correspondent au cadre ouvert dans lequel se situe les travaux de Sichman sur les système multi-agents. Le troisième est lié aux mécanismes de raisonnement qu'il propose pour gérer le processus d'organisation du système.

Le fonctionnement du processus d'organisation proposé se base sur deux types de comportements possibles pour un agent : être actif, ou être passif, selon le choix de l'utilisateur. Dans le premier cas, l'agent a l'initiative de proposer une coalition aux autres, en utilisant un mécanisme de raisonnement social pour le choix de buts, de plans et de partenaires. Dans le second cas, l'agent active les mécanismes de perception et d'inférences sur le domaine, et il se limite à répondre aux propositions de coalition envoyées par les autres.

Le mécanisme de raisonnement social des agents est basé sur la théorie de dépendance [Cast90]. Un agent est dépendant de plans pour atteindre un certain but. Après avoir calculé ses relations de dépendances, l'agent utilise ces dernières pour raisonner sur les autres en fonction de ses croyances sur ceux-ci. Il calcule ainsi ses situations de dépendances envers chaque agent et utilise ces situations pour choisir les partenaires à qui envoyer une proposition de formation de coalition lorsqu'il est incapable d'atteindre un de ses buts seul. C'est ainsi que se forment des organisations dynamiques dans le système DEPINT. Le calcul des relations et des situations de

dépendances nécessite de manipuler des croyances et de maintenir la cohérence de ces croyances pendant l'activité du système. Ce calcul se base sur les plans de chaque agent. Les croyances sont alors des connaissances très fines des buts et des plans des autres agents du système qui permettent aux agents de calculer dynamiquement la structure organisationnelle et de l'adapter au contexte que forme l'ensemble des agents du système.

Nous retirons de tout ceci que le processus d'organisation peut être présent à différentes phases dans un système multi-agents. Il peut être **réalisé par le concepteur** [Carl94] qui définit la structure imposée au système, ou il peut être **réalisé par l'ensemble des agents du système**. Dans ce second cas, il peut intervenir tout au long de la vie du système ou seulement après le démarrage du système.

Le modèle du réseau de contrat [Smit80], [Davi81] et le mécanisme de raisonnement social [Sich95] construisent de manière dynamique la structure organisationnelle tout au long de la vie d'un système, aucune structure organisationnelle n'étant réellement fournie au système au départ. Mais, il est aussi possible d'envisager de définir un système ayant une dynamique d'organisation à partir d'une structure de départ. Cette structure de départ fournie par le concepteur, n'implique pas que celle-ci reste la même quoi qu'il arrive. Dans ce cadre, le processus d'organisation peut être envisagé comme un processus qui permet de passer d'une structure composée de relations utilisées à une autre structure mieux adaptée. Nous appellerons cette évolution dynamique de la structure : **la réorganisation**.

2 Organisation statique ou dynamique pour un SMA

La mise en oeuvre du paradigme multi-agents pour une application amène un certain nombre de difficultés. Il faut qu'un système multi-agents se comporte comme un tout et possède un comportement globalement cohérent. L'aspect statique de l'organisation permet d'exprimer un comportement cohérent et global pour l'ensemble des agents d'un système, mais le problème est alors de choisir le type d'organisation à mettre en place pour l'application.

L'étude des types d'organisation stable proposée par V. Chevrier [Chev93] a montré que l'existence d'une structure optimale d'organisation est peu probable. Une structure d'organisation efficace face à un type de problème peut être inefficace et inopérante face à un autre. Il faut alors que les systèmes multi-agents soient dotés d'une structure adaptée au problème à résoudre. Trouver cette structure optimale n'est pas un problème simple à résoudre, et surtout dans le cas où les composants du système évoluent, cette structure optimale et stable n'existe pas. C'est pourquoi nous pensons que l'aspect statique est insuffisant.

Réduire la notion d'organisation à celle d'une structure dans laquelle l'activité des agents prend place est donc trop limitatif en terme d'applications possibles, même si cette approche permet de résoudre de manière satisfaisante certains aspects du problème de cohérence globale. Il faut également considérer l'organisation comme le processus qui permet d'aboutir à une structure, et comme les différentes interactions qui conduisent à la constitution ou à l'évolution de celle-ci.

Les travaux de Lestrugeon [Stru95] ont développé cet aspect dynamique à partir de l'étude de critères permettant de décider quand appliquer un nouveau modèle de structure à un système. Le

processus d'organisation correspond alors à une réorganisation globale du système permettant d'adapter le système au contexte de la résolution du problème.

Un autre solution est d'envisager la réorganisation, à un niveau plus local, dans les agents. Ceci afin de conserver le caractère autonome des agents et de permettre une évolution moins brutale de la structure. Dans ce cadre local, nous parlerons d'**adaptation dynamique** pour désigner la réorganisation.

Réorganiser, c'est modifier les composants de la société ou leur structuration. L'adaptation peut donc concerner les agents ou leurs relations. Nous développerons pour notre part uniquement l'adaptation des relations. Cette capacité d'adaptation permet aux agents de percevoir la nécessité d'un changement de leurs liens relationnels avec les autres et de procéder aux évolutions nécessaires de la structure.

L'étude des **situations relationnelles** pouvant exister entre les agents [Sich95], [Mart92] distingue deux types de relations : des relations positives et des relations négatives. Les relations positives sont le support essentiel de l'étude de Sichman proposant une théorie de dépendance permettant de raisonner et d'élaborer des relations positives telles que la coopération, la distribution, la collaboration et la cohabitation [Dema90].

Les relations négatives sont généralement moins développées, car la plupart des approches multi-agents visent essentiellement à les éviter. Néanmoins, les situations de conflit (incompatibilité d'actions,...) ou de concurrence (ressources indisponibles,...), qu'elles amènent, peuvent être finalement bénéfiques à l'activité du système [Gall90]. Le processus d'organisation correspond alors à fournir aux agents la capacité d'adapter la structuration du système aux différentes situations relationnelles qui peuvent se succéder durant la résolution.

Nous retiendrons donc qu'une application multi-agents ne subissant aucune évolution de ses composants peut fonctionner à partir d'une structure optimale, si le concepteur est capable de la trouver [Chev93]. Cette structure décrit alors des relations stables entre les agents et indique de manière préférentielle avec qui elles se déroulent. L'utilisation d'un processus d'organisation au niveau des agents est inutile dans ce cadre.

Nous supposons d'une manière générale, que le concepteur d'une application, quelle qu'elle soit, s'il n'est pas capable trouver la structure optimale pour celle-ci, reste toujours capable de fournir une structure de départ pour le système.

De plus, dans le cadre des systèmes composés d'agents communicants (en mode direct), le mode de communication des agents pose un problème : un agent ne peut pas communiquer s'il ne dispose d'aucun moyen pour entrer en relation avec les autres membres du système. L'attribution d'une structure de départ au système est donc nécessaire pour de tels cas ou alors il faut que les agents puissent communiquer de manière indirecte [Smit80], ce qui consiste à fournir une structure non hiérarchique. Un tel système doit donc disposer d'une structure de départ, puisque *"de rien, il ne sort rien"* [Bate84]. Il est alors nécessaire de fournir aux agents la capacité d'adapter leur structuration aux situations relationnelles rencontrées pour aborder la problématique des systèmes dynamiques.

D'une manière générale, que l'organisation soit statique ou dynamique, il est de toute façon nécessaire de fournir aux agents une représentation de la structure de celle-ci qui reste stable ou qui est modifier par les agents. L'organisation peut donc être considérer dans le cadre de la conception de systèmes multi-agents, comme statique et dynamique à la fois.

3 Conclusion

Ce chapitre vient de présenter les principes et concepts de base permettant de mettre en place et de maintenir des relations cohérentes entre des agents, ceci afin de gérer l'activité de l'ensemble des agents participant à la résolution d'un problème.

Les théories de l'organisation et l'étude de ce concept dans le cadre des systèmes multi-agents ont montré la dualité entre la structure organisationnelle et le processus d'organisation pour définir une organisation. Cette approche duale de l'organisation permet d'assurer et de maintenir la cohérence globale, ainsi que l'efficacité dans les SMA.

Une structure organisationnelle est un concept abstrait qui prend concrètement place au travers d'un ensemble de relations (des structures concrètes potentielles définies à partir de rôles et de conventions) permettant aux agents de structurer leurs activités. Cette structure permet la mise en place d'une coordination globale des relations entre les individus. Elle se décrit et s'observe à un niveau global, mais selon le principe d'autonomie défini pour le fonctionnement des agents, elle doit être pilotée à un niveau local par les agents.

Le processus d'organisation peut être entrepris par le concepteur au niveau de la conception du système, par le système lui-même lors de son démarrage ou tout au long de la vie de celui-ci. Nous avons explicité comment nous interprétions ces trois niveaux de mise en place et défini par la même occasion ce que nous désignons par les termes de réorganisation et d'adaptation.

Nous avons montré qu'une structure de départ, même si elle n'est pas optimale, est nécessaire pour permettre aux agents d'utiliser les paradigmes de la communication. L'aspect dynamique correspond alors à l'adaptation de l'organisation de la société par les agents en remettant en cause la structure dans laquelle ils ont été placés ; ceci en fonction des situations relationnelles qu'ils rencontrent et afin de maintenir la cohérence et l'efficacité de fonctionnement du système.

L'organisation est par conséquent un concept complexe qui est associé globalement à un système et qui intervient au niveau des agents (autonomie), comprenant un aspect statique et un aspect dynamique, qui doivent assurer et maintenir la cohérence globale, et contribuer à une amélioration des performances collectives.

Chapitre 6 Conclusion

Dans cette partie, nous avons présenté les différents concepts de base du paradigme multi-agents à la fois du point de vue de leur description et de leur opérationnalisation. Ces concepts sont l'agent, l'interaction et l'organisation. Ils correspondent respectivement à trois niveaux distincts de description d'un système : local, semi-global et global (nous parlerons aussi pour ce dernier d'un niveau de société).

L'agent, quel que soit son type, est l'entité de base de toute approche multi-agents. Il est caractérisé par son autonomie de décision et sa recherche de satisfaction [Ferb97]. Il constitue le niveau local de la description d'un système et le support du fonctionnement du système.

L'interaction est le concept permettant de relier les agents d'un système. Nous l'avons présenté comme un niveau de description semi-global de ces systèmes exprimant les échanges devant être réalisés entre des agents du système. Mais, pour respecter le caractère autonome des agents, ce concept doit être rendu opératoire au niveau des agents eux-mêmes.

Enfin, l'organisation est le concept permettant de désigner l'activité de l'ensemble des agents constituant le système qui forme alors une société. Nous l'avons présenté selon un point de vue global comme la description de cette société et nous avons alors parlé de la structure organisationnelle de celle-ci. Mais, toujours pour respecter le caractère autonome des agents, ce concept doit être rendu opératoire au niveau des agents qui décident et réalisent ensemble l'adaptation de la structure du système (que nous désignons globalement par le terme de réorganisation).

Il est alors possible de concevoir des systèmes multi-agents selon différentes approches [Ferb97] : l'agentification, la résolution distribuée de problèmes, ou l'interactionnisme. La première de ces approches considère que toute la conception et le fonctionnement d'un système se résume à une bonne description du fonctionnement des agents. Tout un courant de chercheurs

qui se réclament de cette approche travaille dans ce sens et sont convaincus que la solution passe directement, et uniquement, par la réalisation formelle d'un modèle d'agent et par son implantation [Shoh93], [Geor84], [Cohe90].

Au contraire, d'autres chercheurs, se reconnaissent dans la notion de "système multi-agents" et insistent sur l'aspect "multi". Ils considérent qu'il faut d'abord penser l'interaction et ensuite en déduire la structure intentionnelle des agents et non l'inverse. L'intérêt des systèmes mutli-agents réside essentiellement dans la notion d'action collective et dans la capacité à articuler les actions individuelles par rapport au collectif à partir des agents. On peut alors sub-diviser cette approche des systèmes multi-agents en deux sous-approches.

La première que l'on appelle "résolution distribuée de problèmes" suppose qu'il existe une fonction de satisfaction globale [Ferb97] pour le système. La conception du système se base alors sur cette fonction globale qui influence alors la mise en place et le choix d'une structure d'organisation exprimant les règles de gestion des interactions et maximisant la satisfaction. L'accent est donc surtout mis sur l'aspect statique de l'organisation.

La seconde approche suppose qu'il n'existe pas de fonction de satisfaction globale permettant de maximiser la résolution. Si cette hypothèse est adoptée, le principe consiste à rechercher alors à maximiser non pas une fonction globale, mais des fonctions de satisfaction locales propres aux agents. Nous désignons ce type d'approche des problèmes en parlant de "résolution multi-agents au sens strict". Il s'agit ici d'obtenir, à partir des interactions et des fonctions de satisfaction locales, l'organisation ou la réorganisation de l'activité de résolution du système. L'accent est donc mis ici à la fois sur l'interaction, l'autonomie de décision et l'aspect dynamique de l'organisation.

Nous adoptons cette dernière approche pour notre travail. Dans ce cadre, nous focalisons nos travaux sur la construction de systèmes multi-agents permettant de spécifier une organisation de départ (qui n'est pas forcément la plus adaptée), ainsi que l'adaptation de cette organisation, ceci en adoptant la position suivante :

les concepts d'interactions et d'organisation sont généralement utilisés pour présenter l'analyse multi-agents des problèmes et expliquer le fonctionnement des systèmes les résolvant. Mais l'implantation de ces systèmes se base généralement sur un seul niveau de description, l'agent, qui comprend éventuellement des modules spécifiques à la gestion de l'organisation et des interactions. Il nous semble alors que cette approche présente un problème majeur vis-à-vis des perspectives d'utilisation du paradigme multi-agents. En effet, le paradigme multi-agents, par sa nature, présente un grand intérêt quand le nombre des agents est élevé et qu'ils sont de nature hétérogène (de par l'expertise, le mode de raisonnement et le comportement interne qu'ils utilisent). Or, dans ce cas, il est de plus en plus difficile d'implanter localement dans les agents, en même temps que l'expertise propre à chaque d'eux, les concepts d'interactions et d'organisation qui ne concernent pas un seul agent. Il nous semble alors important de s'intéresser de plus près à l'utilisation des concepts d'interactions et d'organisation pour la description de systèmes multi-agents, ainsi qu'aux mécanismes permettant à une telle description d'être exploiter localement dans les agents pour conserver le caractère autonome de ceux-ci.

Nous allons présenter dans la partie suivante la démarche que nous avons adopté pour aborder cette problèmatique et l'approche proposée.

Partie II Modélisation multi-agents à partir des interactions

Chapitre 7 Introduction

Nous proposons dans cette partie, une approche hybride impliquant la résolution distribuée de problèmes et la résolution multi-agent au sens strict. Ces deux approches de la résolution de problème se distinguent par leur prise en compte de la notion d'organisation qui est essentiellement statique pour la première et dynamique pour l'autre. Une organisation statique peut aisément se décrire globalement par une structure précisant les relations qui contraignent l'autonomie de décision des agents, alors qu'une organisation dynamique est généralement moins aisée à décrire puisqu'elle accorde plus d'autonomie aux agents et est donc naturellement lié à l'opérationnalisation du système.

La description et l'opérationnalisation sont donc deux aspects à prendre en compte dans la construction de systèmes multi-agents. Nous allons présenter la démarche que nous avons suivie pour prendre en compte ces deux aspects, avant de présenter l'approche que nous proposons.

1 Démarche suivie

Nous avons distingué dans la partie précédente trois niveaux de concepts pour le paradigme multi-agents : les interactions, les agents et l'organisation de la société. De plus, une propriété caractérisant l'approche multi-agents est la réutilisation des agents ou de parties de systèmes multi-agents dans le développement d'autres systèmes. C'est pourquoi, il nous semble nécessaire de ne pas manipuler ces concepts uniquement d'un point de vue opérationnel, mais aussi d'un point de vue descriptif. Par exemple, l'enchaînement des échanges lors des interactions exprimé uniquement selon le principe de continuité locale (adopté pour les objets) irait à

l'encontre de la propriété de réutilisation.

Nous envisageons alors une approche descriptive des trois concepts du paradigme multi-agents, qui peut être représentée selon un axe de localité de la description, comme le montre la figure 7-1.

Un système opérationnel est supporté, selon nous, par le seul concept d'agent autonome qui gère de manière décentralisée les interactions et l'organisation du système et qui représente les constituants de celui-ci. Nous parlons dans ce cadre d'agent autonome, pour différencier ce concept d'agent de celui utilisé pour l'aspect descriptif. La dynamique de l'organisation peut alors être envisagée en dotant les agents de capacités leurs permettant d'influer sur la gestion des interactions et de l'organisation.

Pour des raisons pratiques, nous limitons notre étude aux agents de type communicant et nous supposerons dans cette partie l'existence des mécanismes permettant la communication directe (par exemple, par des boîtes aux lettres associées aux identifiants uniques de chaque agent). Pour ce type d'agents, nous avons montré dans le chapitre 5 que l'organisation et les interactions qui relient ces agents ne peuvent émerger du néant. Il est alors nécessaire de concevoir globalement cette organisation et ces interactions pour pouvoir doter les agents des connaissances nécessaires (accointances, langage d'interaction,...) à la gestion de leurs communications : nous parlerons de connaissances sociales. La gestion des interactions et de l'organisation reposant sur celle de ces connaissances.

Le lien que nous voulons alors définir entre cette vision descriptive et celle opérationnelle d'un système est un lien d'instanciation [Ferb89], [Masi89] de la description des différents concepts en un ensemble d'agents et un ensemble d'interactions entre ces agents que nous organiserons. Ces deux ensembles pourront être fusionnés pour créer l'ensemble d'agents autonomes représentant le système dans son état opérationnel (cf figure 7-1).

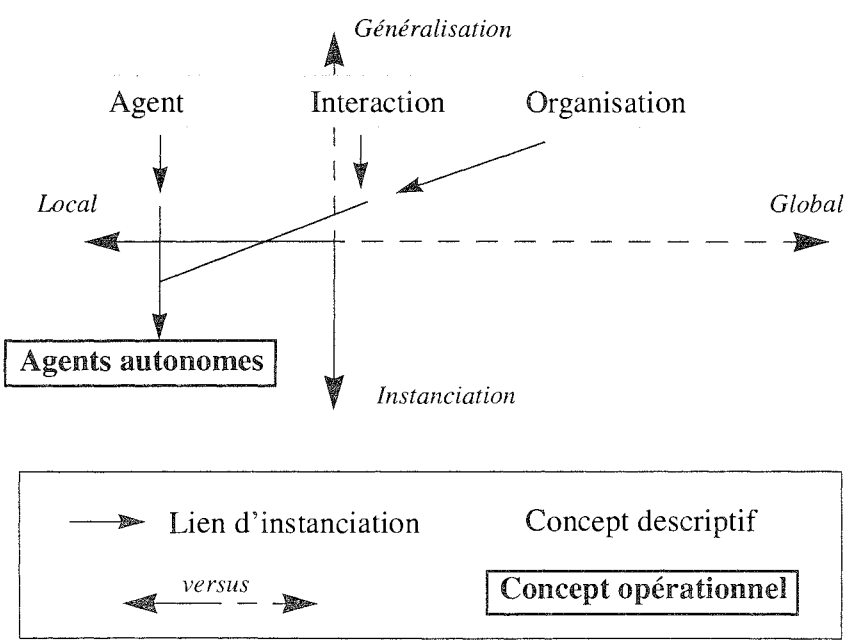


FIGURE 7-1 : Construction de systèmes multi-agents

2 APPROCHE PROPOSÉE

L'approche que nous proposons est à la fois descriptive et opérationnelle. Un modèle de description de systèmes multi-agents est proposé pour permettre de distinguer les trois niveaux de concepts, que nous avons évoqués dans la partie précédente : les interactions, les agents et l'organisation de la société.

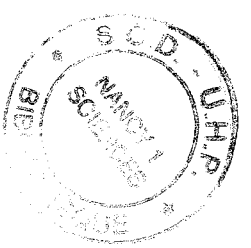
Le premier niveau de description est présenté dans le chapitre 8. Il introduit la notion de schémas d'interaction qui permet de décrire des types d'interactions utilisables entre les agents qui composerons un système applicatif.

Le second niveau de description, présenté dans la chapitre 9, est celui des modèles d'agent qui décrivent des types d'agents spécialistes de l'application, caractérisés par les connaissances du domaine qu'ils traitent, leurs compétences relatives à ce domaine (les actions ou les services qu'ils peuvent accomplir), et enfin par leurs fonctionnements internes.

Le troisième niveau de description est celui de la société d'agents qui décrit l'ensemble des interactions possibles entre les agents composant le système, ainsi que l'organisation de ces interactions. Le chapitre 10 présente ce dernier niveau de description, ainsi que les principes d'exploitation de cette description de la société pour la transformer en un système multi-agent opérationnel.

Cette transformation nécessite la mise en place de mécanismes de liaison entre les caractéristiques décrites dans les modèles d'agent et les interactions spécifiées entre les agents issus de ces modèles. Nous présentons ces mécanismes dans le chapitre 11 en précisant les choix que nous avons fait pour opérationnaliser cette liaison entre les compétences d'un agent et ses interactions.

Enfin, nous terminons cette partie en présentant dans le chapitre 12 les principes de réorganisation que nous avons adoptés dans notre modèle, et qui reposent sur une recherche de satisfaction locale des agents autonomes vis-à-vis de leur interaction. Nous parlerons dans le cadre de ces agents d'adaptation puisque notre modèle n'impose pas une centralisation du processus de réorganisation des interactions.



Chapitre 8 Description de schéma d'interaction

Comme nous l'avons évoqué dans la partie précédente, la structure organisationnelle d'un système correspond à un ensemble de relations entre ses agents. Chacune de ces relations correspond, à un niveau semi-global, à une relation potentielle qui peut prendre corps au travers d'une interaction. Dans ce cadre, nous proposons un modèle pour décrire les interactions possibles entre des agents de manière générique à un niveau semi-global que nous désignons sous le terme de schéma d'interaction.

Nous présentons dans la section 1 les caractéristiques d'un schéma d'interaction. Nous proposons ensuite dans la section 2, une grammaire générique pour exprimer des protocoles d'interactions décrivant l'enchaînement des actions à un niveau semi-global, avant de terminer en présentant une synthèse de ce schéma générique de description des phénomènes interactionnels.

1 Caractéristiques d'un schéma d'interaction

Dans un système multi-agents organisé, différents types d'interactions (selon l'objectif visé, les rôles joués, le protocole d'interaction utilisé,...) peuvent être identifiés, chacun correspondant à la réalisation d'objectifs différents ou à différents enchaînements d'actions de communication entre agents. Ces types d'interactions forment des relations potentielles pouvant être utilisées entre les agents.

Un schéma d'interaction revient alors à décrire une telle relation à un niveau semi-global. Certes, l'enchaînement des actions de communication pourrait s'exprimer à un niveau local pour

chaque agent, mais cela complexifierait la description d'un système. Toutefois cette remarque n'est valable que pour la description des interactions, car des agents autonomes doivent gérer par eux-mêmes l'enchaînement de leur communication. Nous présenterons dans les chapitres suivants la mise en place au niveau local de cette description semi-globale pour la réalisation des interactions.

Nous n'envisageons pas de proposer une classification de ces schémas, mais seulement de déterminer les caractéristiques utiles à leur description. Afin d'illustrer les caractéristiques sur lesquelles se basent notre proposition, nous allons présenter deux exemples de suite d'échanges montrant une infime partie de la diversité des interactions qui peuvent être envisagées pour un système multi-agents.

Considérons une société comportant des agents interagissant pour échanger des informations. Ces agents peuvent être caractérisés par différentes fonctions vis-à-vis d'un type d'information : connaître ou ne pas connaître cette information, avoir besoin ou non de celle-ci, etc. Nous distinguons alors pour cet exemple trois types de participants pouvant intervenir dans les interactions : les participants de type A (correspondant à un type d'agent ayant besoin de cette information et ne connaissant pas celle-ci), les participants B (correspondant à un type d'agent connaissant l'information) et les participants C (correspondant à un type d'agent ne connaissant pas cette information). Les participants de type C sont par conséquent inclus dans ceux de type A.

Les agents ne connaissant pas l'information et ayant besoin de celle-ci (participants A) doivent pouvoir interagir avec les autres agents pour savoir quels agents peuvent leur donner ce renseignement et enfin demander à ces derniers cette information. Ces agents pourront par conséquent être amenés à interagir sur le sujet de l'obtention de l'information de deux manière distinctes. Nous présentons deux exemples d'interaction : le premier exemple d'interaction consiste à interagir pour demander l'information et le second correspond à interagir pour se renseigner sur la disponibilité de ce renseignement (cf figure 8-1).

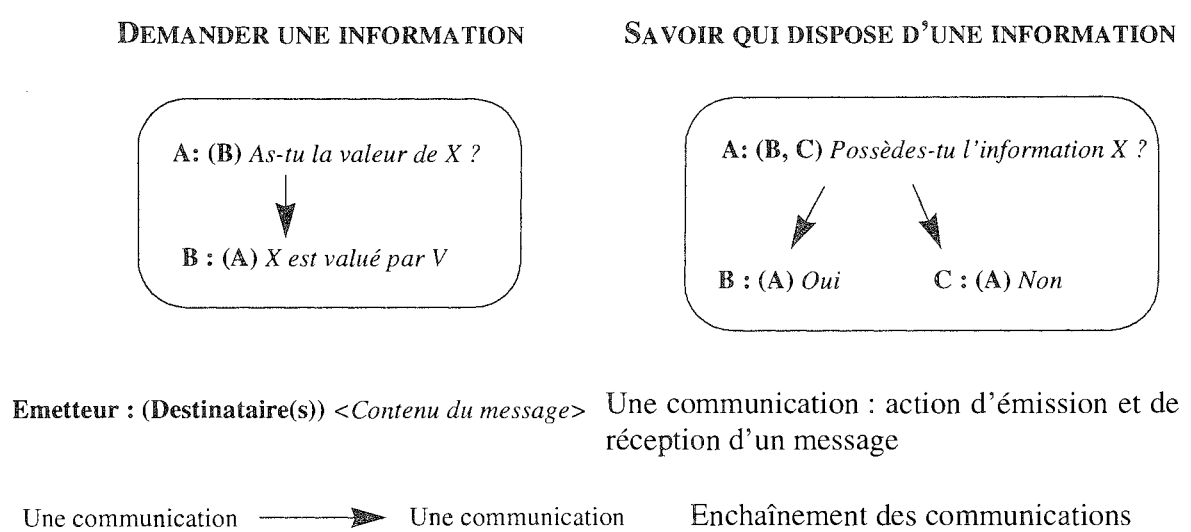


FIGURE 8-1 : Exemples de schémas d'interaction

Ces deux exemples illustrent trois caractéristiques d'un schéma d'interaction : l'objectif à atteindre par le schéma, le type des participants (A, B, C) et l'enchaînement des messages. Les notions de type de participants et d'enchaînement correspondent à une description d'un proto-

cole d'interaction. L'objectif à atteindre permet d'exprimer quelle sera la motivation qu'un agent pourra satisfaire en utilisant ce schéma.

Nous proposons alors une description de schémas d'interaction basée sur trois composants : la motivation (qui représente l'objectif, le but qu'une telle interaction permet d'atteindre), l'ensemble des participants abstraits à ce type d'interaction et le protocole d'interaction utilisable entre les agents y participant. Un schéma d'interaction correspond alors à un triplet : motivation, participants abstraits, protocole d'interaction.

<schéma d'interaction> = <motivation> <participants abstraits> <protocole d'interaction>

1.1 La motivation de l'interaction

La motivation de l'interaction se rapporte aux motifs qui vont pousser un agent à interagir et qui définissent le but qu'une interaction permet d'atteindre : par exemple, un agent ne possède pas les connaissances requises, il n'a pas le savoir-faire requis pour l'utilisation de ces connaissances (incompétence), ou le coût d'utilisation de ces connaissances est trop élevé pour lui.

Si l'on se réfère à la théorie de la dépendance de Sichman [Sich95], cette notion de motivation de l'interaction correspond à une relation de dépendance potentielle entre les agents concernés par ce schéma d'interaction. Cette relation de dépendance sera l'indice qui permettra à un agent de savoir par lui-même comment satisfaire son besoin d'autrui pour faire réaliser certaines actions ou obtenir certaines informations.

Nous considérons que la motivation peut s'exprimer par une séquence de mots-clés. Mais ceci n'est pas limitatif, elle pourrait aussi s'exprimer comme une fonction paramétrée.

<motivation> = [<mot-clé>] +[1]

1.2 Les participants abstraits de l'interaction

Afin de conférer un caractère générique aux schémas d'interaction, nous ne précisons pas directement dans ceux-ci l'identification des participants. Nous adoptons une description basée sur la notion de participants abstraits, ce qui permettra la réutilisation d'un schéma entre différents ensembles d'agents.

Ces participants abstraits situent dans un contexte semi-global les actions respectives qui incombent aux agents participants à l'interaction. Nous désignons ces participants par un ensemble de rôles que les agents peuvent tenir dans le schéma d'interaction. L'utilité de ces rôles est uniquement d'étiqueter les actions intervenant dans le protocole d'interaction du schéma (comme nous le verrons dans la description d'un protocole d'interaction).

<participants abstraits> = [<rôle>] +

1.3 Le protocole d'interaction

Un schéma d'interaction permet de définir de manière générique une relation de dépendance (la motivation) entre des participants abstraits (des rôles), mais un tel schéma doit aussi préciser comment mettre en place la suite d'actions qui va donner corps à cette relation. Ceci correspond à la notion de protocole d'interaction.

---

1. Au moins un mot-clé.

Nous proposons de décrire ces protocoles d'interaction en reprenant les rôles choisis pour représenter les participants abstraits d'un schéma d'interaction. Le but de cette description est d'exprimer l'enchaînement des actions à entreprendre pour réaliser une interaction ainsi modélisée.

La description que nous envisageons pour ces protocoles d'interaction n'est pas limitée à un enchaînement d'actions de communication. Nous intégrons également les actions des agents jouant un rôle dans cet enchaînement (d'une manière similaire à la description locale des comportements des agents dans SDL'92 [Færg94], mais à un niveau inter-agent dans notre proposition). Il ne s'agit donc pas d'un simple enchaînement de communications, mais plutôt d'un enchaînement d'actions (au sens général) concernant les partenaires abstraits du schéma d'interaction.

Quel que soit leur type, ces actions ne sont pas définies lors de la description du protocole d'interaction. Les actions sont seulement désignées par leurs interfaces d'appel pour exprimer la sémantique de l'enchaînement d'actions. Une action, quelle qu'elle soit, est propre à un agent et sera entreprise par celui-ci. Sa réalisation est donc supposée être définie localement pour chaque agent et dépend de la manière dont l'agent l'entreprend.

Par ailleurs, cette description d'enchaînement d'actions définit un langage d'expression d'interactions. Nous considérons que ce langage d'interaction n'est pas nécessairement intégralement connu par tous les agents du système, mais que les agents connaissent uniquement une partie de ce langage qui est relative à leurs compétences et à leurs relations de dépendance avec les autres agents du système. Chaque agent est alors considéré comme un spécialiste disposant d'un vocabulaire et de facultés d'expression limités, mais qui lui sont propres.

1.4 Conclusion

Un schéma d'interaction forme un cadre descriptif d'un type d'interaction. Aucun nom d'agent n'apparaît explicitement lors de la description et aucune action n'est définie complètement à ce niveau. Un schéma d'interaction est composé d'un triplet : motivation, participants abstraits, protocole d'interaction.

Nous n'envisageons pas de formuler un mécanisme de raisonnement complexe sur les caractéristiques des participants ou sur les motivations intervenant dans ce schéma. Mais, nous souhaitons seulement utiliser ces participants et ces motivations comme indices élémentaires lors de la description d'interactions : instanciation des schémas (les participants concrets) et réalisation de ceux-ci (la motivation et les interfaces d'appel des actions) comme nous le montrerons respectivement dans le chapitre 10 et 11.

La grammaire d'expression des schémas d'interaction est la suivante :

<schéma d'interaction> = <motivation> <participants abstraits> <protocole d'interaction>

<motivation> = [<mot-clé>] +

<participants abstraits> = [<rôle>] +

où un seul élément n'est pas explicité de manière élémentaire et demande plus de détails d'un point de vue de descriptif : le protocole d'interaction. Nous poursuivons ce chapitre en exprimant le formalisme de description des protocoles d'interaction.

2 DESCRIPTION DE PROTOCOLE D'INTERACTION

Les protocoles d'interaction correspondent à des connaissances décrivant l'enchaînement générique d'actions à réaliser par un ensemble d'agents. Dans ce cadre, un protocole d'interaction est représenté par un graphe d'états finis[1] décrivant les actions successives entreprises par des participants abstraits lors d'une interaction. Ce graphe est composé de noeuds et de transitions entre ces noeuds, afin d'exprimer l'enchaînement d'actions. La grammaire d'expression d'un tel graphe est la suivante :

<protocole d'interaction> = <noeuds> <transitions>

Nous utilisons trois types de noeuds : un noeud initial qui est le point d'entrée dans l'enchaînement d'actions, un noeud final qui en est le point de sortie et des noeuds intermédiaires qui prennent place entre les deux premiers types de noeuds.

<noeuds> = <noeud initial> <noeuds intermédiaires> <noeud final>

<noeud initial> = **initial** <étape>

<noeud final> = **final** <étape>

<noeuds intermédiaires> = [<noeud intermédiaire>] *

<noeud intermédiaire> = **inter** <étape>

Une étape d'un graphe correspond à une suite d'actions à réaliser durant cette étape.

<étape> = [<action>] *

Chacune de ces actions concerne uniquement un des participants abstraits (un rôle). Une action correspond à l'appel d'une action propre à l'un des participants.

<action > = <rôle> **:** <appel d'action>

Ces appels peuvent prendre deux formes : un appel d'actions avec résultat (une fonction) ou sans résultat (une procédure). L'appel décrit est composé des caractéristiques de l'interface de ces actions : leur nom (étant une simple étiquette), leurs paramètres (s'il y en a) et leur résultat si l'action en fournit un.

<appel d'action > = **name** <nom> (<paramètres>) → <résultat> |

name <nom> (<paramètres>)

La réalisation de ces actions dépend des agents jouant le rôle concerné : chaque agent pouvant adopter une définition différente pour celles-ci. Dans notre premier exemple (cf figure 8-1), chaque participant de type B a sa propre valeur ou sa propre manière de calculer l'information X, ce qui signifie que la valeur V dépendra de l'implantation choisie dans chaque agent.

Les paramètres et le résultat (s'il existe) des actions sont décrits à partir de variables contextuelles. Une telle variable est caractérisée par son nom (précédé d'un $ dans la syntaxe présen-

1. Un réseau de Pétri serait un formalisme équivalent pour l'expression de ces connaissances. Mais ce formalisme de description puissant n'est pertinent que si une validation des protocoles est envisagée, ce qui n'est pas le cas dans ce mémoire.

tée) et son type (entier, caractère, chaîne de caractère,...) pour permettre de les manipuler. L'utilisation de ces variables dans la description d'un protocole d'interaction permet d'exprimer les connexions existantes entre les actions, les étapes et les transitions décrivant ce protocole.

<paramètres> = [<variable>] *

<résultat> = <variable>

<variable> = $<nom de variable> <type>

L'enchaînement des noeuds est exprimé grâce à des actions de transition qui peuvent être de différentes catégories : un prédicat à valider, un envoi ou une réception[1] de message à effectuer. Comme les actions précédemment décrites, ces actions de transitions concernent uniquement un des participants abstraits dans le schéma d'interaction (un rôle) et correspondent à une description générique de l'appel de ces actions.

<transitions> = [<noeud> <action de transition> <noeud>] +

<noeud> = <noeud initial> | <noeud intermédiaire> | <noeud final>

<action de transition> = <rôle> : [<prédicat> | <envoi> | <réception>]

L'interface d'appel d'un prédicat est décrite de la même manière qu'une fonction. Les autres actions de transitions sont des actions de communication : action d'envoi ou de réception. Ces dernières sont également décrites par un appel correspondant respectivement, au niveau des agents communicants, à l'envoi de message ou à la mise en correspondance d'un message reçu avec un modèle de ce message (la réalisation de ces actions étant considérée comme fiable et asynchrone).

Ces appels d'actions spécifiques utilisent également la notion de participants abstraits pour désigner le(s) destinataire(s) (pour l'envoi) ou l'expéditeur (pour la réception) du message.

<envoi> = **send to** [<rôle>] + : <modèle de message>

<réception> = **receive from** <rôle> : <modèle de message>

Le modèle d'un message correspond à l'expression de l'assemblage (ou du désassemblage pour la réception) des informations transitant par un message entre des agents. Cette expression peut prendre différentes formes, en se basant par exemple sur les travaux autour des actes de langage. Nous avons adopté pour le moment une expression simple de ces modèles de message basée sur des mots-clés (choisis par le concepteur du schéma d'interaction en fonction de l'application) et des noms de variables contextuelles, ce que nous représentons par la syntaxe suivante :

<modèle de message> = [<mot-clé> | $<nom de variable>] *

En reprenant le premier exemple de la figure 8-1, une description du protocole d'interaction à partir de cette grammaire se réalise comme le montre la figure 8-2. Cette description comprend deux appels d'actions *InfoX* et *PriseEnCompteX* qui sont des actions définies respectivement pour les agents jouant le rôle B et le rôle A.

1. La réception correspond plutôt à un événement qu'à une action, mais un agent devra consulter sa boîte aux lettres pour prendre en compte cet événement, donc réaliser une action de consultation.

Dans cette figure 8-2, les flèches simples représentent l'enchaînement des noeuds et des transitions entre ceux-ci. Les flèches doubles illustrent le lien de réciprocité entre un envoi et une réception. Le texte en italique correspond aux mots-clés choisis pour les messages, qui s'inspirent dans ce cas de la syntaxe et de la sémantique de la langue française. Les noms de variable sont précédés du signe **$**, et s'ils apparaissent dans un appel d'action, ils sont suivis du type d'information véhiculée par cette variable (le type entier dans cet exemple). Comme nous l'avons défini dans la grammaire, l'appel d'une action comprend l'énumération des paramètres de celle-ci qui peut éventuellement être un ensemble vide (voir le cas des fonctions *InfoX* dans la figure 8-2 : ()).

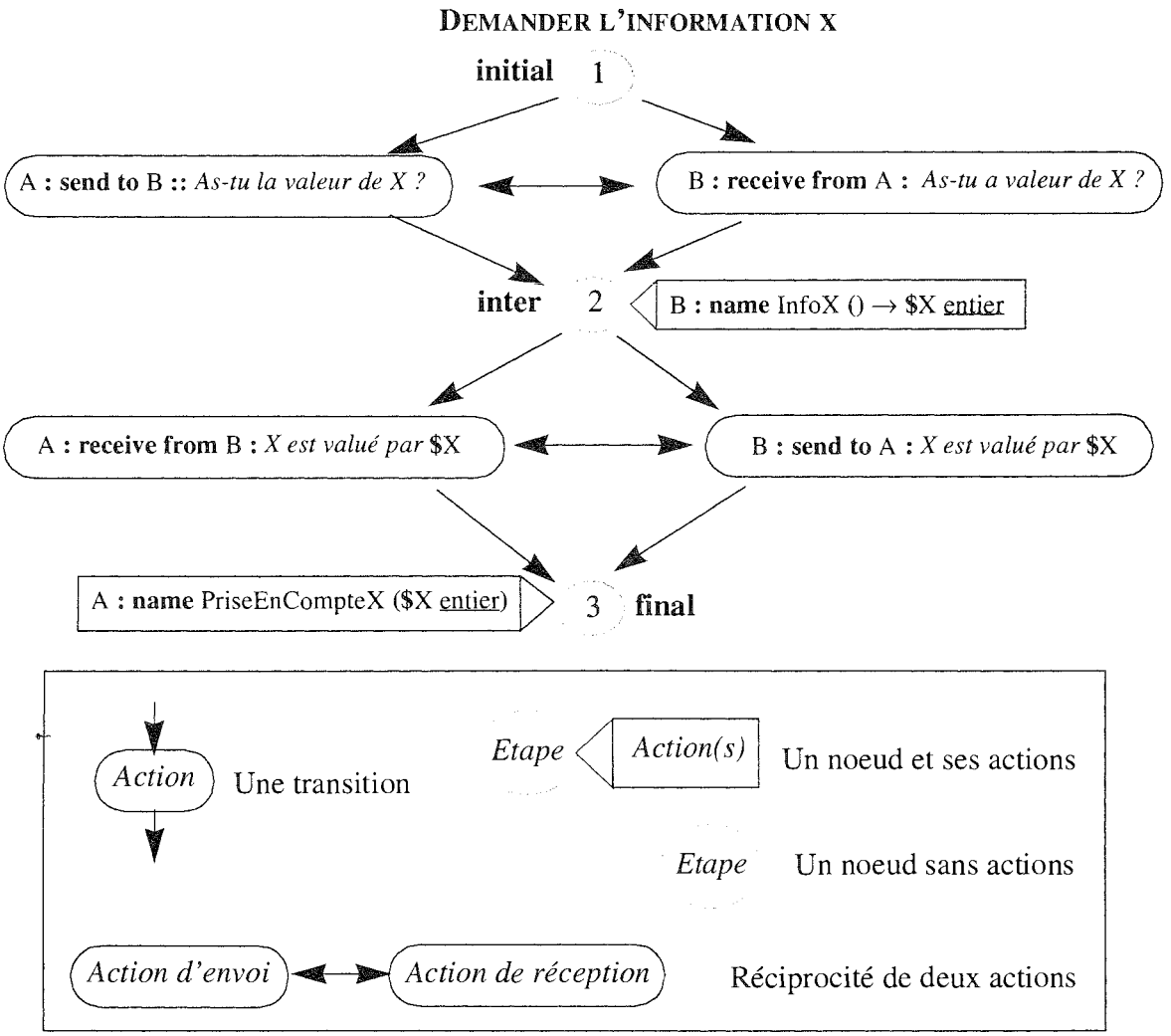


FIGURE 8-2 : Description d'un protocole pour demander l'information X.

Ce protocole correspond suivant notre grammaire aux expressions suivantes :

noeuds = { **initial** 1

inter 2

final 3 }

1 = Λ[1]

2 = { B : **name** InfoX() → $X entier }

3 = { A : **name** PriseEnCompteX ($X entier) }

transitions = { **initial** 1 A : **send to** B : *As-tu la valeur de X ?* **inter** 2

initial 1 B : **receive from** A : *As-tu la valeur de X ?* **inter** 2

inter 2 B : **send to** A : *X est valué par $X* **final** 3

inter 2 A : **receive from** B : *X est valué par $X* **final** 3 }

Plus généralement, il faut aussi signaler certaines spécificités dans l'emploi de cette grammaire d'expression des protocoles d'interaction. Ces spécificités correspondent à deux contraintes à respecter lors d'une modélisation de protocole d'interaction :

- La première contrainte est une contrainte de réciprocité entre les actions de communication (cf figure 8-2). Celle-ci impose que chaque action d'envoi de message doit correspondre à une réception ayant le même modèle de message et où le rôle du récepteur correspond au rôle du destinataire désigné lors de l'envoi (et réciproquement).

- La seconde contrainte est une contrainte de connexité sur le graphe décrivant le protocole d'interaction. Elle impose qu'au moins un chemin connecte le noeud initial au noeud final et que tout chemin partant du noeud initial permette d'atteindre le noeud final.

Nous ajoutons en plus une contrainte supplémentaire qui nous servira d'hypothèse pour la mise en place des mécanismes de gestion des interactions (*cf chapitre 11*) :

- Toutes les actions de transition liées au noeud initial sont des actions de communication (envoi ou réception).

3 Conclusion

Dans ce chapitre, nous avons proposé un modèle original de description d'interaction semi-global et générique. Ce modèle prend en compte à la fois les liens structurels (rôles et motivation) et interactionnels (enchaînement des actions de communication) liant des agents du système. Il considère qu'un schéma d'interaction est composé d'une motivation, d'un ensemble de participants abstraits et d'un protocole d'interaction exprimant l'enchaînement d'actions à effectuer pour réaliser cette interaction.

Le protocole d'interaction est décrit selon une grammaire spécifique permettant l'expression des interactions à un niveau inter-agent. La particularité de cette grammaire est de permettre une description inter-agent suffisamment explicite pour exprimer l'enchaînement des actions, tout en restant générique. Cet enchaînement peut correspondre à une suite d'actions déterminée de manière statique, si elle ne comporte que des actions de transition de type communication. Mais cette suite d'action peut également être déterminée dynamiquement en fonction des prédicats

1. Aucun appel d'action

qui interviennent comme action de transition.

Les aspects essentiels de notre description des schémas d'interaction résident dans la description abstraite et partielle des actions qu'ils enchaînent, qui correspond à la généricité des schémas :

- la notion de participants abstraits (rôles) remplace la désignation des agents,

- les actions sont définies par leur interface d'appel,

- les variables contextuelles expriment des liaisons entre les actions composant la description du protocole.

Notre approche de la description des schémas présente la particularité de ne pas supposer qu'un langage d'interaction est défini de manière globale [Dema95]. Le langage d'interaction est pour nous spécifique à chaque agent (qui est considéré comme un spécialiste ayant des domaines privilégiés et un langage spécifique pour ces domaines). Le langage d'interaction de chaque agent se construit en fonction des schémas d'interaction dans lesquels il intervient comme participant. Chaque protocole d'interaction définit une partie de ce langage au niveau inter-agent et non au niveau global du système. L'enrichissement du langage est alors aisément réalisable en ajoutant des schémas d'interaction supplémentaires pour décrire d'autres interactions de l'agent.

Chapitre 9 Description d'agents

La conception d'un système multi-agents peut amener à introduire dans un système plusieurs agents similaires. Nous proposons de prendre en compte cette similarité lors de la description des agents en introduisant des modèles génériques d'agents. Ces modèles définissent les connaissances, les compétences (savoir-faire) et le comportement des agents, et offrent la possibilité de les paramètrer pour spécifier les particularités de chaque agent.

Néanmoins, la description de ces modèles n'est pas entièrement spécifiée pour éviter d'imposer un formalisme d'expression de ceux-ci, cette expression pouvant selon nous aussi bien être réalisée de manière procédurale ou sous d'autres formes, par exemple des bases de connaissances et un ATN (Augmented Transition Network), comme le modèle d'agent proposé dans DIMA [Gues97].

Ce chapitre présente les différents éléments nécessaires et minimaux à partir desquels nous proposons de construire la description du comportement des agents, ainsi que les hypothèses relatives à ceux-ci. Nous précisons aussi la description que nous proposons pour les agents basée sur l'instanciation de modèles d'agent.

1 ELÉMENTS DE DESCRIPTION D'UN AGENT

Un agent est une entité qui encapsule des connaissances et des capacités d'actions. Ces connaissances et ces capacités d'actions, d'un point de vue privé, peuvent prendre des formes bien différentes selon l'application pour laquelle l'agent a été conçu. Nous parlons alors de connais-

sances et d'actions individuelles permettant de décrire le comportement des agents.

Mais pour concevoir aisément des systèmes multi-agents, il semble commode, pour notre approche, de disposer d'un cadre homogène pour exprimer le comportement social des agents (cet aspect du comportement permettant aux agents d'entrer en interaction avec les autres membres du système). Il est alors nécessaire que les agents disposent de moyens d'interaction, de communication et de connaissances relatives à l'utilisation de ces moyens.

Nous proposons d'adopter les définitions suivantes pour la description des agents :

- Les **connaissances individuelles** sont des connaissances spécifiques pouvant prendre des formes variées. Leur forme est fonction de l'agent en tant que spécialiste d'un aspect de la résolution du problème.

- Les **actions individuelles** permettent à l'agent de manipuler ses connaissances individuelles et correspondent à ses compétences relatives à la résolution (pour lui permettre d'agir sur lui-même).

- L'**identifiant** des agents est un élément supplémentaire de la description des agents. Il est l'élément déterminant de tout envoi de message. Il permet l'acheminement d'un message en désignant de manière unique chaque agent du système. Il est utilisé par les agents comme une connaissance leur permettant de communiquer avec un autre agent. Cette connaissance est généralement désignée dans ce cadre sous le terme d'accointance dans les langages d'acteurs [Hewi77], [Masi89].

- Les **connaissances sociales** sont des connaissances utilisant un formalisme uniforme permettant aux agents de disposer d'une représentation de leur accointances, ou plus généralement de leur interactions potentielles. Elles sont issues des schémas d'interaction que nous venons de présenter.

- Les **actions de communication** permettent aux agents de communiquer. Ces actions de communication sont, par hypothèse dans notre approche, issues des deux mécanismes suivants :

  - un mécanisme d'envoi de message,

  - un mécanisme de réception de messages et de stockage[1] des messages.

- Les **actions d'interaction** permettent aux agents de réaliser des interactions en liant les connaissances sociales dont ils disposent avec leurs actions individuelles ou leurs actions de communication.

L'ensemble de ces éléments caractéristiques permet alors de définir en fonction de l'application différentes formes de **comportements** internes. La définition de ces comportements constitue l'élément principal de la description des agents. Ils spécifient les liens existant entre tous les autres éléments caractérisant les agents.

La définition d'un comportement précise quand et comment un agent va utiliser ses connaissances (individuelles ou sociales) et ses capacités d'actions, ce qui consiste comme nous l'avons déjà évoqué dans la partie précédente à expliciter : "comment assembler les différentes parties d'un agent de manière qu'il accomplisse les actions que l'on attend de lui ? " [Ferb95]

1. Dans une boîte aux lettres

2 MODÈLES D'AGENT : DESCRIPTION ET INSTANCIATION

Nous proposons de décrire un agent en précisant son identifiant (qui le différencie de tous les autres), le modèle qu'il suit et les paramètres de ce modèle (exprimant les spécificités de l'agent). Ainsi, un agent correspond à une instance identifée d'un modèle d'agent :

<agent> = <identifiant> **apply** <modèle d'agent> (<paramètres>)

<modèle d'agent> = <connaissances> <compétences> <comportement>

Les paramètres intervenant dans les modèles d'agent peuvent être variés. Nous n'explicitons pas ici la déclaration ou l'instanciation de ceux-ci, car elle ne ferait que reprendre des techniques usuelles de programmation orienté-objet (classes génériques). Nous présentons plutôt comment nous relions cette grammaire définissant un modèle d'agent aux différents éléments de description.

<connaissances> = <connaissances individuelles> <connaissances sociales>

<compétences> = <actions individuelles> <actions sociales>

Les actions sociales constituent pour nous l'une des caractéristiques de base sur lesquelles repose le cadre homogène de description des aspects sociaux du comportement des agents. Elles sont indépendantes de toute application, et correspondent aux actions permettant aux agents d'interagir :

- les actions de communication, qui correspondent aux mécanismes de communication utilisés,

- les actions d'interaction, qui lient les connaissances sociales de l'agent et son modèle aux mécanismes de communications.

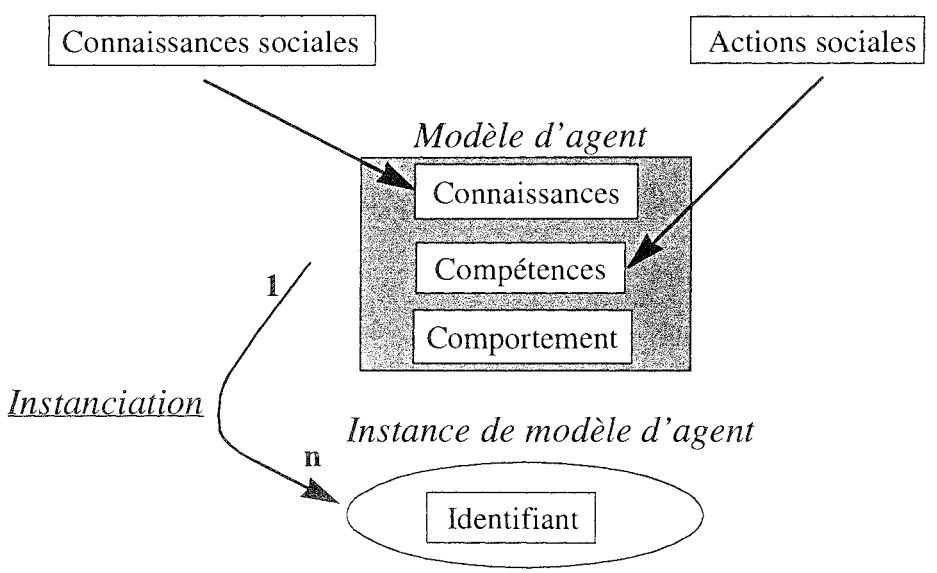


FIGURE 9-1 : Du modèle d'agent aux instances.

Ces actions sociales et les connaissances sociales sont considérées dans ce chapitre comme

connues pour chaque modèle d'agent. Elles constituent pour nous les éléments de base de la spécification de l'aspect social du comportement des agents, comme illustré sur la figure 9-1. Tout modèle d'agent se base alors sur l'existence de ces actions et de ces connaissances pour l'expression du comportement de celui-ci.

D'autres caractéristiques peuvent venir compléter notre grammaire de description des modèles d'agent (les engagements, les intentions, etc.). Mais nous supposons que les éléments présentés servent de base pour exprimer les autres. En ce qui concerne le formalisme d'expressions de ces éléments, nous considérons qu'il dépend du langage de programmation retenu pour implanter ce modèle et nous ne faisons aucune hypothèse sur celui-ci.

Afin de préciser comment cette grammaire de modèle d'agent permet de spécifier les comportements des agents composant un système, nous reprenons l'exemple du schéma d'interaction présenté dans le chapitre précédent : demander une information X.

Cet exemple fait intervenir deux participants abstraits pour lesquels nous allons proposer deux modèles d'agent. Le premier de ces modèles correspond aux agents demandant l'information et peut s'exprimer par l'énumération des éléments suivants :

- Une connaissance individuelle :

  Une structure pour mémoriser l'information X qui est inconnue initialement.

- Une action individuelle :

  Une action permettant de mémoriser une information X : *PriseEnCompteX*.

- Un comportement :

  Le comportement d'un tel agent consiste à décider ou non d'obtenir cette information X. Quand la décision d'obtenir cette information est prise, l'agent débute alors une interaction basée sur une de ses connaissances sociales à partir des actions d'interaction dont il dispose. Ces connaissances sont issues du schéma d'interaction décrit précédemment où cet agent joue le rôle A (voir la figure 8-2).

Nous utiliserons dans le chapitre suivant l'instanciation de ce modèle pour un agent identifié par le nom d'Alfred.

Le second modèle correspond aux agents recevant des demandes d'information et s'exprime par les éléments suivants :

- Une connaissance individuelle :

  Une structure représentant l'information X.

- Une action individuelle :

  Une action pour accéder à la valeur de cette information : *InfoX*.

- Un comportement :

  Le comportement de ce type d'agent consiste à décider de fournir ou non l'information X. Quand la décision de fournir cette information est prise, l'agent consulte les messages de sa boîte aux lettres et débute des interactions pour répondre à ces sollicitations. Il utilise pour cela ses actions d'interaction selon les messages reçus et les connaissances

sociales issues du schéma d'interaction où cet agent joue le rôle B (voir la figure 8-2).

Nous utiliserons également dans le chapitre suivant l'instanciation de ce deuxième modèle pour un agent identifié par le nom de Bernard.

3 CONCLUSION

Nous avons présenté dans ce chapitre la grammaire de description partielle que nous adoptons pour définir les agents composant le système et leurs comportements. Cette grammaire est partielle, car nous avons souhaité rester général et éviter de proposer une architecture interne pour définir le comportement des agents.

La description des agents est proposée sur la base de modèles d'agent, dont nous avons présenté les caractéristiques essentielles : connaissances, compétences et comportement. L'expression de ces modèles peut revêtir des formes diverses en fonction des langages de programmation qui seront utilisés pour les décrire : les connaissances peuvent être des faits ou des variables, les compétences peuvent être des règles, des fonctions ou des méthodes, et le comportement peut être par exemple le programme principal exploitant le tout.

Nous avons abordé ici uniquement la description des agents. Il est important de signaler que dans ce cadre un agent décrit n'est qu'une instance d'un modèle d'agent et ne correspond nullement à un agent opérationnel puisqu'aucune connaissance sociale n'a encore été fournie à cet agent. L'agent opérationnel est le résultat de l'activation d'une telle instance que nous désignerons sous le terme d'agent autonome.

Le propos de cette thèse n'est pas de désigner le langage ou le formalisme le plus adéquat pour décrire ces modèles, ni même de proposer un modèle standard d'agent, mais d'expliciter comment prendre en compte les interactions telles que nous les avons définies pour exprimer l'aspect social du comportement des agents. C'est pourquoi, nous avons considéré dans ce chapitre l'existence de connaissances sociales et d'actions sociales dont nous allons expliciter la nature et le fonctionnement dans les chapitres suivants.

Chapitre 10 Description et mise en place d'une société d'agents

La description de société est basée sur un ensemble de schémas d'interaction et un ensemble de modèles d'agent. Elle consiste à instancier les agents composant la société à partir des modèles, et à instancier les schémas entre les agents ainsi définis. Ceci correspond à exprimer les interactions prenant place dans la société et la structure organisationnelle adoptée par celle-ci.

Nous allons expliciter dans la section 1, comment nous décrivons concrètement les interactions prenant place dans la société, avant de présenter, dans la section 2, comment nous définissons la structure organisationnelle de celle-ci. Nous terminons en reprenant les éléments essentiels de cette description.

1 DESCRIPTION DES INTERACTIONS

La description des interactions entre les agents d'une société correspond à préciser des schémas d'interaction entre des agents. Cela consiste à désigner les noms des agents (identifiants) et le schéma d'interaction suivant ce schéma, afin d'associer à chacun de ces agents un rôle dans le schéma *(cf figure 10-1)*. Chaque rôle peut être joué par un ou plusieurs agents : une interaction peut donc être multi-partite [Barb95] (au moins deux rôles), et comme dans le réseau contractuel [Smit80], plusieurs agents peuvent avoir le même rôle.

La description d'une société se base donc sur l'instanciation de schémas qui décrivent des interactions et sur l'instanciation d'agents *(cf figure 10-2)*, ce qui correspond à la grammaire suivante :

<société> = [<agent>] +
[<interaction>] +

<interaction> = <schéma d'interaction> (<agents participants>)

<agents participants> = [<identifiant> **plays role** <rôle>] +

L'ensemble de ces instances de schémas définit un ensemble d'interactions potentielles pour chaque agent de la société. Il s'agit alors de distribuer à chacun de ces agents la représentation locale de chacune de ses interactions, ce qui correspond à créer les connaissances sociales évoqués dans le chapitre précédent.

Ces connaissances sociales correspondent à une motivation et à un schéma de conversation. La motivation est celle spécifiée dans le schéma d'interaction dont elle est issue. Le schéma de conversation correspond au protocole d'interaction où chaque rôle a été remplacé par le ou les identifiants des agents jouant ce rôle. Une connaissance sociale est alors automatiquement déduite d'un schéma d'interaction et peut se représenter sous la forme suivante :

<connaissance sociale> = <motivation> <schéma de conversation>

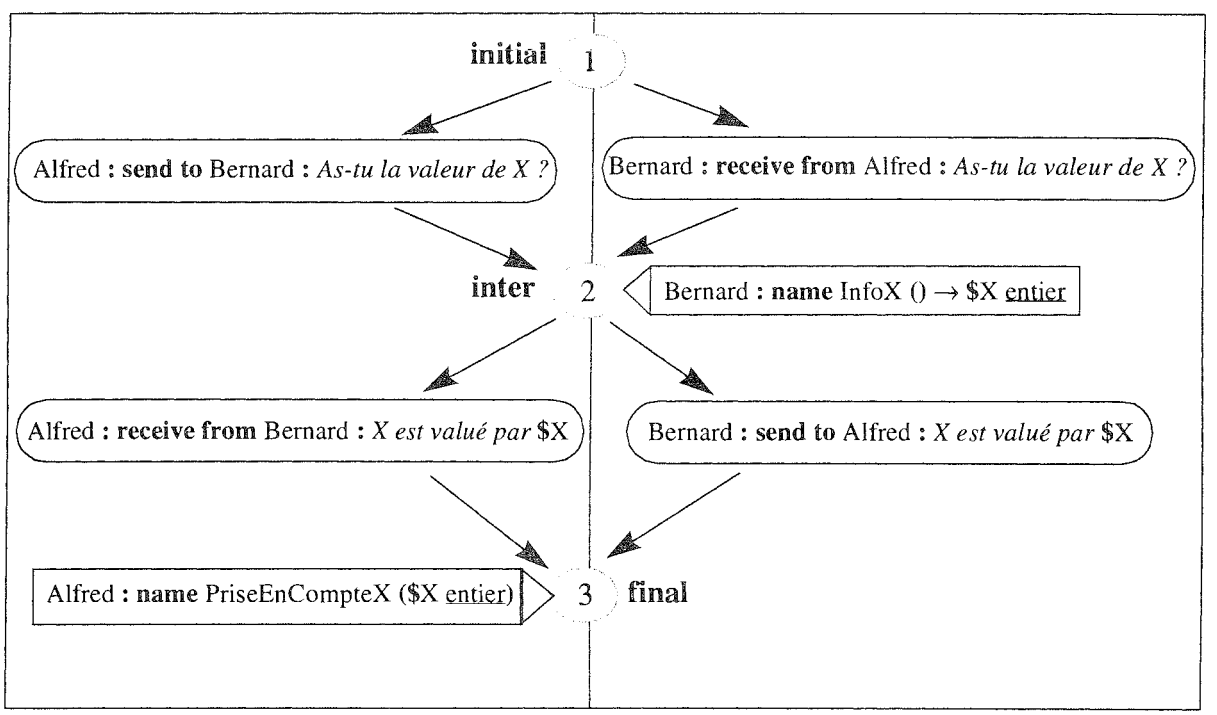


FIGURE 10-1 : Le schéma de conversation pour Alfred et Bernard

Ces connaissances permettent aux agents de réaliser localement une interaction. Nous n'envisageons pas ici d'utiliser ces connaissances afin de réaliser un raisonnement complexe sur les interactions qu'elles représentent, mais seulement d'utiliser ces représentations pour operationnaliser les interactions. Il est envisageable de simplifier cette représentation locale en retirant toutes les actions dont l'agent n'est pas l'instigateur, afin par exemple de minimiser la taille de cette base de connaissances sociales. Nous parlerons dans ce cadre d'une focalisation des schémas de conversation spécifiques à chaque agent. Sur notre exemple, cela signifie que Alfred et Bernard auraient une expression différente du schéma de conversation. Voici l'expression du schéma qu'aurait Alfred dans ce cas :

noeuds = { **initial** 1
inter 2
final 3 }

1 = Λ

2 = Λ

3 = { A : **name** PriseEnCompteX ($X entier) }

transitions = { **initial** 1 Alfred : **send to** Bernard : *As-tu la valeur de X ?* **inter** 2
inter 2 Alfred : **receive from** Bernard : *X est valué par $X* **final** 3 }

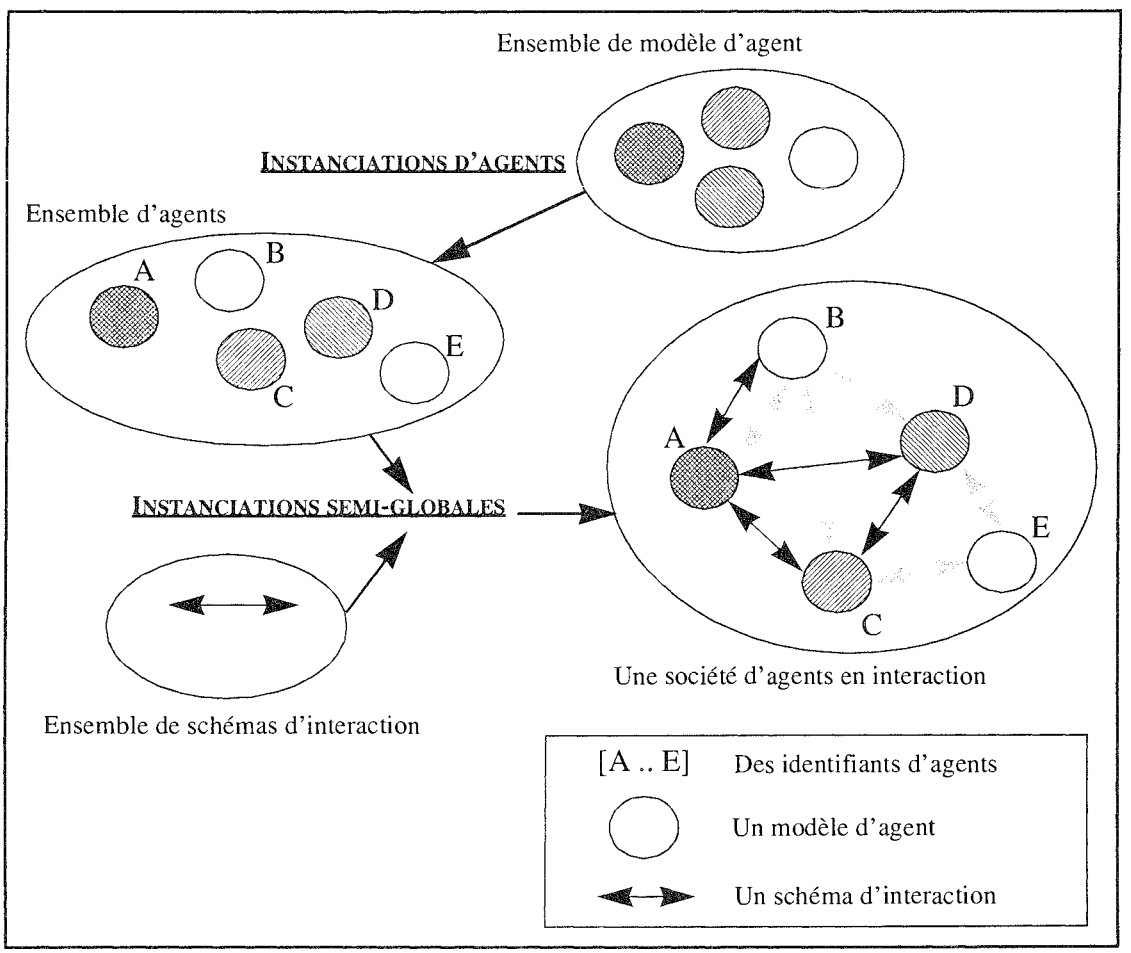


FIGURE 10-2 : Instanciation d'une société.

Un schéma de conversation fournit à un agent un point de vue sur les actions à enchaîner et à réaliser pour effectuer une interaction. Le nombre de schémas construits à partir d'un protocole d'interaction correspond au minimum au nombre de rôles différents intervenant dans celui-ci et au maximum au nombre d'agents participant à cette interaction. La construction d'un schéma de conversation *(cf figure 10-1)* est illustrée sur l'exemple du protocole de demande d'une information *(cf figure 8-2)* qui est instancié entre deux agents Alfred et Bernard tenant respectivement les rôles A et B.

Nous distinguons par la mise en place de ces connaissances sociales une nouvelle forme d'instanciation, sous le terme d'instanciation locale qui correspond à la distribution d'une instanciation d'un schéma localement dans chaque agent y participant. A l'issue de l'application de ce mécanisme de distribution, l'ensemble des membres de la société dispose de connaissances sociales *(cf figure 10-3)* dont nous avions supposé l'existence dans le chapitre précédent. Les agents peuvent alors être activés pour former une société opérationnelle d'agents autonomes (ce qui correspond à l'activation des prototypes pour les langages d'acteurs [Masi89]).

L'ensemble des connaissances sociales d'un agent forme son potentiel d'interaction qui peut présenter des connaissances alternatives. En effet, plusieurs de ces connaissances correspondant à des interactions potentielles différentes (protocole d'interaction ou agents participants différents) peuvent se rapporter à une même motivation *(cf figure 10-2)*. Cette particularité implique que les connaissances formant le potentiel d'interaction d'un agent représentent plus qu'une structure concrète (cf chapitre 5 de la partie I) locale à un agent [Ferb95]. Elles représentent un potentiel de structures utilisables par un agent. Le problème est alors de choisir une structure de départ parmi ce potentiel de structures pour l'activité des agents. Nous allons présenter dans la section suivante comment nous proposons d'identifier la structure à adopter pour les agents lors de la description de la société.

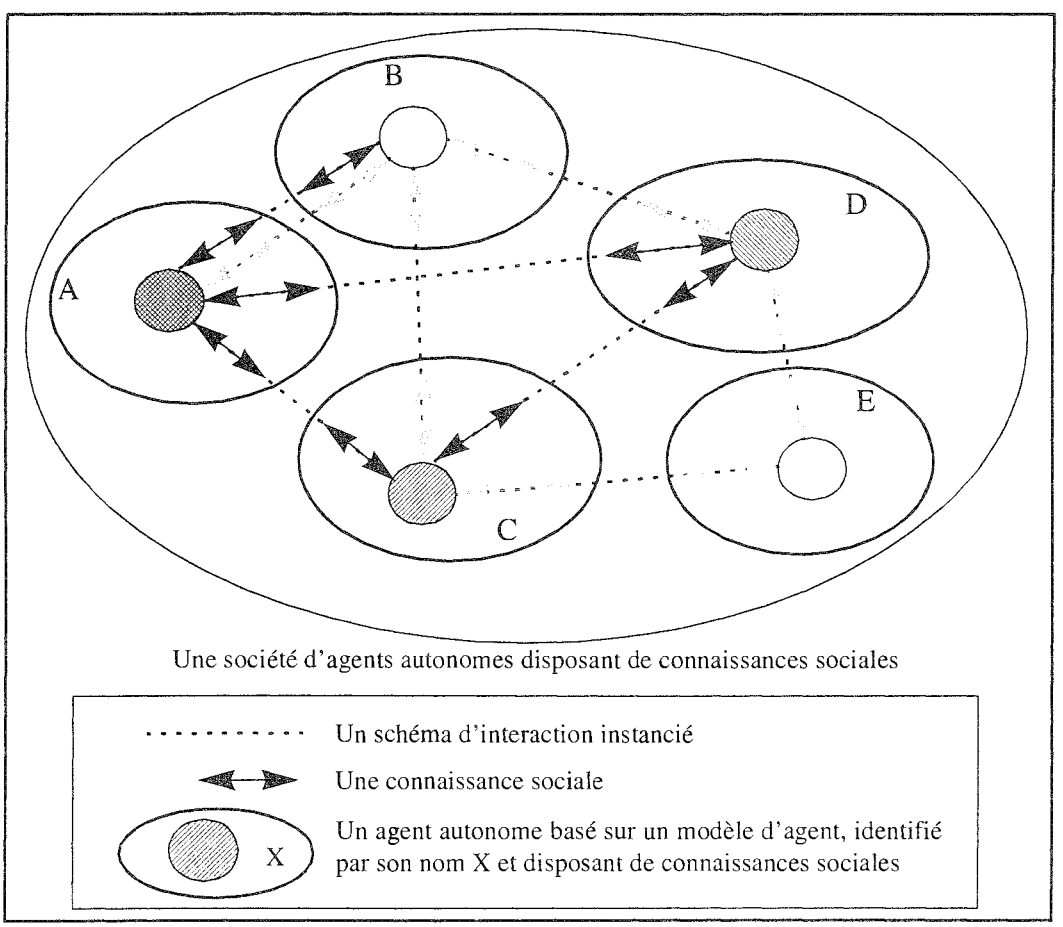


FIGURE 10-3 : Une société d'agents autonomes : instanciation locale des connaissances sociales.

2 STRUCTURE ORGANISATIONNELLE

Pour identifier la structure organisationnelle que les agents doivent adopter parmi le potentiel de structures dont ils disposent, nous proposons d'établir un ordre préférentiel sur les interactions potentielles. Construire une société organisée correspond alors à indiquer une structure préférée lors de l'instanciation des schémas d'interaction *(cf figure 10-4)*.

Nous proposons d'obtenir ceci en ajoutant un niveau de préférence pour chaque agent participant d'une interaction instanciée. Ce niveau de préférence est un élément p d'un ensemble de préférences appelé P et sur lequel une relation d'ordre est définie. Cette préférence est donnée pour chaque agent désigné lors de l'instanciation des schémas d'interaction.

La description d'une société correspond alors à la nouvelle grammaire :

<société> = [<agent>] +

[<interaction>] +

<instance d'interaction> = <schéma d'interaction> (<agents participants> <préférence des agents>)

<agents participants> = [<identifiant> **plays role** <rôle>] +

<préférence des agents> = [<identifiant> **with preference level** <p>] +

Ces instances de schémas sont toujours distribuées de la même manière aux agents participants pour leur fournir l'ensemble de leurs connaissances sociales. Les niveaux de préférence, qui sont associés à ces connaissances, permettront alors aux agents de gérer l'utilisation de ces connaissances. Ces connaissances sociales fournissent aux agents une vision partielle des interactions prenant place dans la société *(cf figure 10-3)* et un point de vue personnel sur la structure à adopter (la préférée). Les connaissances sociales d'un agent respectent alors la forme suivante :

<connaissance sociale> = <p> <motivation> <schéma de conversation>

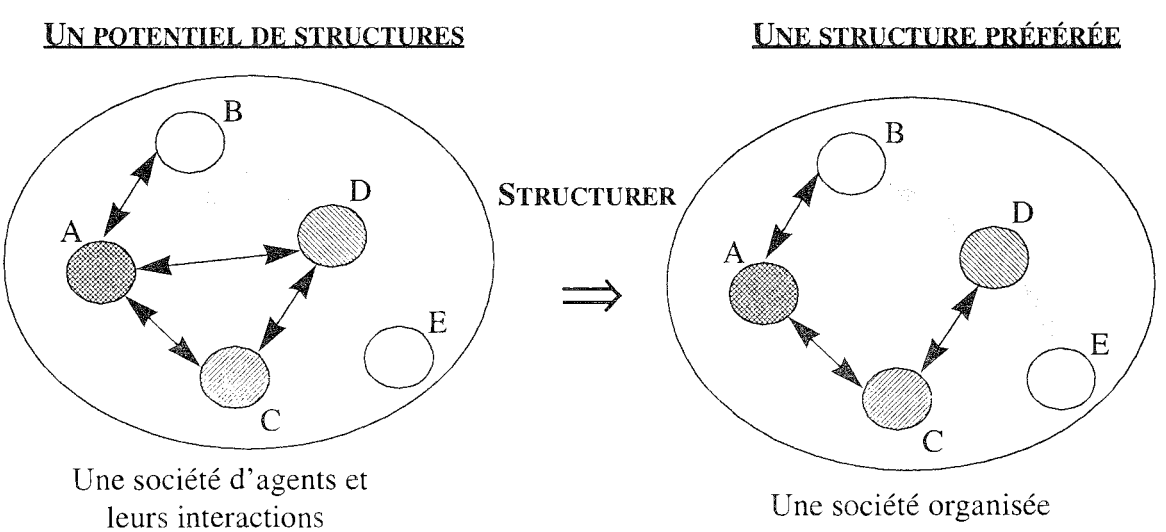


FIGURE 10-4 : Une description de société organisée

3 SYNTHÈSE

La description d'un système multi-agents consiste à définir un ensemble de schémas d'interaction, un ensemble de modèles d'agent et une description de la société qui explicite les agents, les interactions et la structure organisationnelle prenant place entre les agents. A partir des différentes grammaires que nous avons présentées dans ce chapitre et dans les deux précédents, une description de système correspond à :

<système multi-agents> = <schémas d'interaction> <modèles d'agent> <société>

<schémas d'interaction> = [<schéma d'interaction>]+

<modèles d'agent> = [<modèle d'agent>]+

<société> = [<agent>] +

[<interaction>] +

<interaction> = <schéma d'interaction> (<agents participants> <préférence des agents>)

<agents participants> = [<identifiant> **plays role** <rôle>] +

<préférence des agents> = [<identifiant> **with preference level** <p>] +

Rendre cette description opérationnelle correspond alors à construire des connaissances sociales à partir des interactions décrites et à distribuer celles-ci dans chaque agent. Cette distribution permet aux agents de disposer de leur propre point de vue sur leurs interactions potentielles et sur une partie des structures organisationnelles possibles du système. Un système multi-agents opérationnel correspond alors à un ensemble d'agents autonomes, disposant de connaissances sociales qu'ils sont supposés utiliser pour interagir et participer à l'organisation de la société.

Nous présentons dans le chapitre suivant, les actions sociales évoquées dans le chapitre 9 qui permettent aux agents autonomes d'utiliser leurs connaissances sociales. Ces actions se basent sur différents mécanismes de liaison permettant de relier les connaissances sociales aux actions d'un agent. Ceci afin de montrer comment le fonctionnement des agents est rendu opérationnel après une interprétation automatique d'une description de système.

Chapitre 11 Mise en place de mécanismes de liaison dans les agents.

Dans les chapitres précédents, nous avons présenté une grammaire de description de systèmes et son interprétation pour la construction d'un ensemble d'agents autonomes disposant de connaissances sociales. Notre approche n'étant pas que descriptive, nous présentons dans ce chapitre les mécanismes permettant aux agents d'utiliser leurs connaissances pour réaliser des interactions.

Ces différents mécanismes ont pour but de mettre en liaison les connaissances sociales d'un agent avec son comportement et ses actions individuelles qui sont les caractéristiques internes différenciant chaque agent, afin d'expliciter comment fonctionnent les agents.

Nous avons envisagé ces mécanismes pour la réalisation des interactions dans un cadre interruptible [Bela96], afin qu'un agent puisse mener simultanément différentes interactions. L'agent initie alors ses interactions les unes après les autres (selon ses préférences), mais la poursuite de celles-ci se réalise de manière simultanée.

Cette gestion des interactions reprend des principes qui se retrouvent généralement dans les protocoles des systèmes répartis [Juan95] et en particulier, nous l'avons envisagé en nous basant sur les automates d'états finis utilisés pour SDL'92 [Tata97] qui correspondent à nos schémas de conversation.

Nous explicitons dans la section 1 comment utiliser les connaissances sociales d'un agent dans la définition de son comportement pour lui permettre d'initier et de réaliser les interactions qu'elles décrivent. Dans la section 2, nous montrons comment effectuer les appels d'actions exprimés dans ces connaissances en utilisant les actions individuelles spécifiques à chaque agent. Enfin dans la section 3, nous présentons les techniques utilisées pour mener une interaction initiée à son terme, avant de conclure.

1 COMPORTEMENT ET INTERACTIONS

Les connaissances sociales issues de l'instanciation des schémas d'interaction peuvent être utilisées de différentes manières pour initier une interaction. Nous présentons ici les situations d'utilisation de ces connaissances, ainsi que les mécanismes permettant de relier le comportement d'un agent à ses connaissances sociales pour l'initiation des interactions.

1.1 Situations d'interaction

L'utilisation des connaissances sociales consiste à mettre en place une suite d'échanges avec les autres agents. Cette suite d'échanges peut débuter de deux manières pour un agent : soit par l'envoi d'un message, soit par la réception d'un message (conformément à la contrainte que nous avons adoptée pour les actions de transition liées au noeud initial d'un protocole d'interaction, dans le chapitre 8).

Un agent doit donc être capable de catégoriser ses connaissances en fonction de cette première action. On distingue alors deux situations d'interaction :

- une interaction débute par une action de communication vers les autres agents. L'agent peut donc utiliser cette interaction pour ***agir***, car c'est lui qui provoque l'interaction en émettant le premier message. L'agent placé dans cette situation sera appelé ***agent initiateur*** de l'interaction, c'est le cas d'Alfred *(cf figure 10-1)*,
- une interaction débute par une action de communication dépendant des autres agents. L'agent ne peut donc utiliser cette interaction que si les autres agents décident d'agir selon celle-ci. L'agent ne peut que ***réagir*** alors à une sollicitation prenant la forme d'un message reçu. Nous appellerons un agent dans cette situation : l'***agent sollicité*** de l'interaction, c'est le cas de Bernard *(cf figure 10-1)*.

La première situation est basée sur une utilisation des connaissances sociales d'un agent à partir d'une motivation poussant l'agent à interagir avec d'autres. Cette motivation est issue du raisonnement et des décisions prises par l'agent au travers de son comportement. Nous utilisons cette motivation comme déterminant dans le choix d'une connaissance sociale (parmi les connaissances sociales lui permettant d'agir).

La deuxième situation est basée sur une utilisation des connaissances sociales à partir d'un message reçu sollicitant l'agent pour une nouvelle interaction. Nous appelons ce type de message, un message de sollicitation. L'initiation de l'interaction consiste à rechercher parmi les connaissances sociales de l'agent celle qui concerne l'agent émetteur du message et dont la première action est une réception caractérisée par l'identifiant de l'agent émetteur et par un modèle du message correspondant au contenu du message reçu.

Ces deux situations correspondent pour nous à deux mécanismes permettant d'initier une interaction. Cette initiation correspond à réaliser le noeud initial du schéma de conversation, à réaliser la transition partant de ce noeud et à effectuer le noeud suivant (cf figure 11-1). L'initiation d'une interaction ne consiste donc pas à réaliser complètement une interaction, mais simplement à la débuter. Une interaction sera désignée sous le terme d'interaction initiée. Il est alors nécessaire d'introduire un mécanisme pour ***poursuivre les interactions initiées*** et mener ainsi la réalisation complète d'une interaction préalablement initiée. Nous détaillons ce mécanisme dans

la dernière section de ce chapitre qui présente la gestion des interactions initiées.

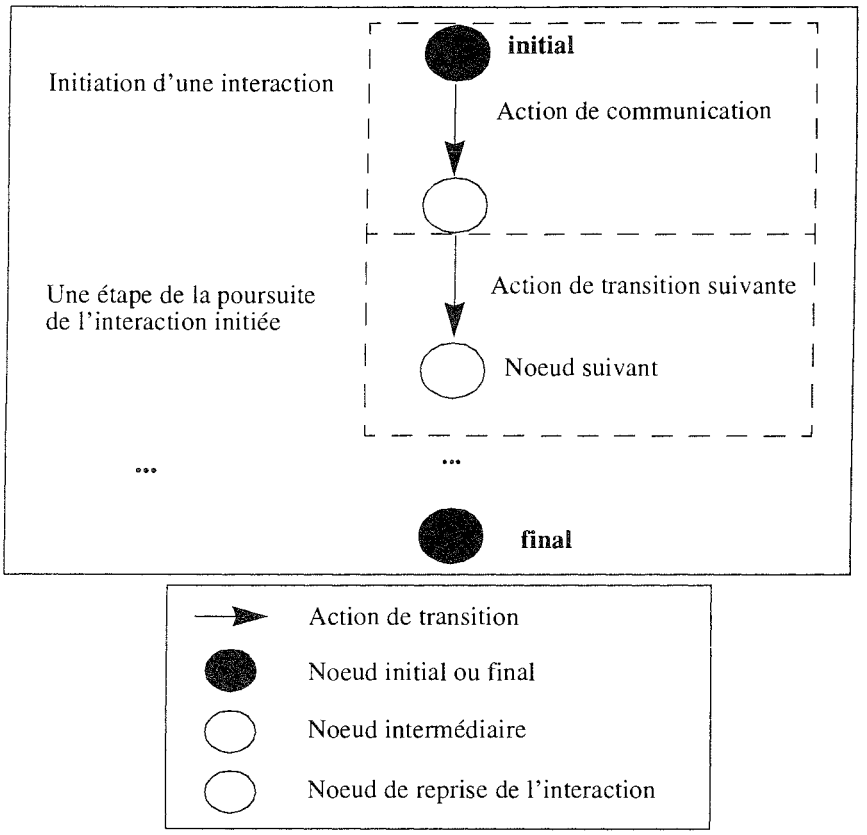


FIGURE 11-1 : Principe d'initiation et de poursuite d'une interaction.

1.2 Initiation d'une interaction

Nous avons montré dans le chapitre précédent comment nous proposons de fournir une structure organisationnelle à une société d'agents en désignant des préférences pour chaque connaissances sociales concernant un agent. Ces préférences permettent d'organiser les connaissances sociales d'un agent et nous allons maintenant expliciter comment celles-ci sont utilisées pour l'initiation des interactions.

1.2.1 Initiation d'une interaction à partir d'une motivation

Agir en fonction d'une motivation peut mettre l'agent dans la nécessité de faire un choix. En effet, il peut disposer de plusieurs connaissances sociales caractérisées par la même motivation pour lesquelles il est initiateur de l'interaction.

Dans ce cas, le mécanisme d'initiation d'une interaction d'un agent consiste tout d'abord à extraire de ses connaissances sociales celles qui correspondent à la motivation initiatrice. Cette extraction a pour résultat un sous-ensemble de ces connaissances, parmi lesquelles l'agent choisit celle qui a le niveau de préférence le plus élevé. Nous dirons que cette connaissance est celle pour laquelle l'agent éprouve la plus forte confiance. L'interaction peut alors être initiée selon le schéma de conversation de la connaissance sociale choisie.

1.2.2 Initiation d'une interaction à partir des messages de sollicitation reçus

Pour initialiser une nouvelle interaction en fonction des messages de sollicitation reçus, un agent examine sa boîte aux lettres. Si celle-ci ne contient qu'un message de sollicitation, il se contente de le prendre en compte, si plusieurs messages sont dans sa boîte aux lettres, l'agent doit choisir lequel prendre en compte en premier. Pour cela, l'agent met alors ces messages en correspondance avec ses connaissances sociales. Nous supposons ici qu'un agent est capable de trouver une connaissance sociale unique pour chaque message de sollicitation qu'il reçoit, ce qui correspond à faire l'hypothèse que la description des interactions de la société est cohérente et n'a pas introduit d'indéterminisme pour la prise en compte d'un message de sollicitation.

L'agent dispose ainsi pour chaque message de sollicitation reçu d'une connaissance sociale décrivant l'interaction à réaliser pour réagir à celui-ci. L'agent choisit alors parmi ces connaissances celle qui a le niveau de préférence le plus élevé. Nous dirons que cette connaissance est celle pour laquelle l'agent a la plus forte attention. L'interaction peut alors être initiée selon le schéma de conversation de la connaissance sociale choisie.

1.2.3 Autonomie de décision des agents pour l'initiation d'interactions

L'autonomie accordée aux agents pour la gestion de leurs interactions permet d'obtenir une co-construction d'une interaction entre les participants de celle-ci, selon le principe suivant :

- une interaction est débutée par un agent ***initiateur*** quand celui-ci décide de manière autonome d'interagir selon la motivation qui caractérise cette interaction. L'agent initiateur débute alors cette interaction selon la connaissance sociale qui correspond à cette motivation et envers laquelle il a le plus confiance. Il envoie donc un message.
- l'agent ***sollicité*** ne participe pas forcément immédiatement à la construction de cette interaction quand il décide de prendre en compte les messages de sollicitation qu'il a reçus. En effet, il construira une interaction à partir du message envoyé par l'initiateur seulement s'il a suffisamment d'attention pour la connaissance sociale qui correspond à ce message. L'agent sollicité participe alors à la construction de l'interaction seulement quand celle-ci retient son attention.

D'une point de vue global, l'observation de mise en place d'une interaction entre différents participants n'est donc pas uniquement fonction de l'agent initiateur de celle-ci, mais aussi du bon vouloir de l'agent sollicité (ou des agents successivement sollicités) qui décide(nt) quand réagir à des sollicitations et dans quel ordre prendre en compte ces sollicitations. C'est pourquoi, nous parlons de co-construction de l'interaction.

2 CONNAISSANCES SOCIALES ET ACTIONS PROPRES

Nous avons montré dans la section précédente que l'initiation d'interactions, quel que soit le mécanisme utilisé, correspond à la réalisation d'appels d'actions décrits dans un schéma de conversation.

La description des protocoles d'interactions est à la base de l'instanciation des schémas de conversation. Conformément à cette description, un appel d'actions à effectuer est de l'un des

cinq types suivants : envoi, réception, prédicat, action avec ou sans résultat. Il faut alors relier ces descriptions d'appels aux actions propres de l'agent.

Pour la description des modèles d'agent, nous avons posé l'hypothèse que les actions de communications (envoi ou réception) sont possédées par tous les agents, alors que les actions individuelles (prédicat, action avec ou sans résultat) sont spécifiques à chaque agent. L'interprétation de la description d'une action de communication est donc la même pour tous les agents, alors que l'interprétation de la description d'une action individuelle est spécifique à chaque agent.

Il s'agit alors de définir des mécanismes de liaison entre ces descriptions d'appels et les actions définies pour le modèle de l'agent. Nous détaillons maintenant ces mécanismes pour réaliser l'envoi et la réception de message après avoir préciser la structure adoptée pour ceux-ci, puis nous détaillons les mécanismes de liaison permettant d'effectuer les actions individuelles.

Les mécanismes de communication dont dispose chaque agent se basent sur la structure suivante des messages :

<message> = <indices> <contenu>

<indices> = <émetteur> <destinataire(s)> <identifiant de conversation>

Les indices d'un message [Camp90] sont fournis lors de l'envoi de celui-ci et permettent de mettre en correspondance les messages reçus avec les schémas de conversation grâce à :

- l'identifiant de l'émetteur et du (ou des) destinataire(s) qui correspondent évidemment aux agents désignés comme tels lors de la description d'un envoi ou d'une réception.

- l'identificateur de conversation qui correspond à un indice supplémentaire permettant de différencier les messages de sollicitation (initiation de l'interaction) des messages suivants (poursuite de l'interaction initiée). Cet identificateur de conversation permet d'associer directement et de manière unique un message à une interaction préalablement initiée.

Pour les envois de messages, le mécanisme de liaison consiste à générer un message à partir du modèle spécifié dans le schéma de conversation. Il suffit alors de remplacer les variables contextuelles désignées dans le modèle du message par leur valeur pour obtenir le contenu du message à envoyer et de fournir les indices de ce message. Un message ainsi construit est alors envoyé à partir du mécanisme d'envoi de message et ainsi franchit la transition du schéma de conversation.

D'une manière réciproque, pour une réception de message, le mécanisme consiste à mettre en correspondance un message reçu par l'agent (qui peut être un message de sollicitation ou non) avec le modèle de cette réception. Cette mise en correspondance consiste alors à extraire éventuellement les valeurs contenues dans le message reçu pour affecter les variables contextuelles désignées par le modèle de celui-ci et à franchir cette transition.

Le franchissement d'une transition peut également s'effectuer à partir d'une variable contextuelle de type booléen. Il s'agit dans ce cas des appels de type prédicat. Si le prédicat est vrai la transition est franchie, sinon l'interaction reste bloquée sur cette transition.

Pour les appels d'actions avec ou sans résultat, les mécanismes de liaison doivent mettre en relation les caractéristiques d'un appel d'action avec une action individuelle définie lors de la description du modèle de l'agent. La définition de ces mécanismes est donc spécifique à chaque modèle d'agent, et doit préciser, en fonction des caractéristiques de l'interface d'appel d'une

action (nom, paramètres et résultat), l'action individuelle à sélectionner et à effectuer avec les valeurs des variables contextuelles désignées comme paramètres, et fournissant éventuellement un résultat dont la valeur sera affectée à la variable contextuelle désignée comme résultat par l'interface de cet appel.

3 GESTION DES INTERACTIONS INITIÉES

Les interactions initiées sont des représentations locales aux agents qui permettent de piloter le déroulement de l'automate correspondant au schéma de conversation de l'interaction. Un agent peut gérer simultanément plusieurs de ces représentations, ce qui signifie que les interactions peuvent être effectuées en simultané.

Nous allons commencer par présenter la structure adoptée pour représenter une interaction initiée et comment celle-ci est créée, puis nous explicitons comment l'utiliser pour poursuivre les interactions initiées qu'elle représente.

3.1 Représentation des interactions initiées

Une interaction initiée est associée à une représentation locale créée lors de l'initiation d'une interaction. Elle est composée des éléments suivants :

- l'***identificateur de la conversation*** permet de différencier les interactions en cours. Il est construit par l'agent initiateur de l'interaction à partir de son identifiant et du nombre d'identificateurs qu'il a préalablement construit, ce qui permet d'assurer l'unicité de cet identificateur. Un agent sollicité adopte l'identificateur contenu dans le message de sollicitation qu'il reçoit, car cet identificateur interviendra dans tous les messages concernant cette interaction. Ce principe est donc similaire à la gestion des connexions dans HTTP [Fiel97] à partir du numéro de port et de l'adresse IP.

- le ***schéma de conversation*** de la connaissance sociale dont est issue l'interaction,

- l'***étape courante*** dans ce schéma de conversation qui correspond au noeud initial lors de la création de cette représentation et qui est modifiée par les mécanismes d'initiation et de poursuite de l'interaction (cf figure 11-1).

- les ***variables contextuelles*** (nom, type et valeur) qui se rapportent à cette interaction.

L'identificateur de la conversation est utilisé dans tous les messages échangés, afin de permettre aux agents de gérer l'interprétation des messages reçus : si une interaction initiée correspond à l'identificateur d'un message reçu, il suffit alors de poursuivre celle-ci; par contre si aucune interaction initiée ne correspond à celui-ci, cela signifie que ce message est un message de sollicitation à traiter via le mécanisme d'initialisation d'une interaction.

3.2 Poursuite des interactions

Ces représentations sont utilisées par le mécanisme de poursuite des interactions pour mener à leur terme les interactions et enchaîner les actions de transition. Nous allons seulement présenter un exemple de politique possible pour ce mécanisme qui a pour principe de continuer en

priorité les interactions en attente d'une réception.

Selon ce principe, le mécanisme de poursuite prend le premier message de la boîte à lettres qui n'est pas un message de sollicitation. En fonction de l'identificateur de conversation, l'agent est capable d'associer ce message à une interaction initiée. Il réalise alors la réception correspondante (grâce au mécanisme de liaison défini pour les réceptions) et effectue l'étape suivante (qui devient alors l'étape courante pour cette interaction).

Si aucun message n'est contenu dans la boîte aux lettres de l'agent, celui-ci choisit alors une de ses interactions initiées qui doit effectuer un envoi de message ou qui doit vérifier un prédicat. Ce choix consiste à prendre l'interaction initiée en premier par cet agent (FIFO). Il suffit alors de réaliser cette action de transition grâce au mécanisme de liaison correspondant et d'effectuer l'étape suivante.

Le mécanisme de poursuite d'interaction possède une caractéristique commune avec le mécanisme d'initiation d'interaction à partir des messages de sollicitation reçus : il débute par une prise en compte des messages de la boîte aux lettres. Nous envisageons donc une fusion de ces deux mécanismes, afin de simplifier la gestion des interactions au niveau du comportement des agents. Le mécanisme résultant de cette fusion consiste alors à examiner la boîte aux lettres pour initier en priorité de nouvelles interactions (prise en compte des messages de sollicitation) et si aucun message de sollicitation n'a été reçu, l'agent poursuivra alors les interactions initiées selon les principes que nous venons de présenter.

4 Synthèse

Dans ce chapitre, nous avons présenté les mécanismes élémentaires permettant de réaliser les interactions à partir des comportements des agents : initiation des interactions à partir d'une motivation ou à partir des messages de sollicitation reçus et poursuite des interactions initiées.

Ces mécanismes utilisent et gèrent les représentations des interactions initiées dans chaque agent. Cette gestion est réalisée dans un cadre interruptible où le déroulement des interactions s'effectue selon une progression étape par étape. Cette gestion utilise des liaisons entre les schémas de conversation et les actions individuelles ou entre ces schémas et les actions de communication des agents.

Ces liaisons sont explicitées par des mécanismes permettant de lier les appels d'actions décrits dans les schémas de conversation aux actions individuelles ou aux actions de communication exprimées lors de la description des modèles d'agent. Ces mécanismes permettent la réalisation des actions composant un schéma de conversation et la mise à jour des valeurs des variables contextuelles.

L'intérêt essentiel des mécanismes d'initiation des interactions est d'exprimer le comportement d'un agent en se focalisant sur les spécificités de celui-ci liées au domaine traité (buts et actions individuelles), et en explicitant dans ce comportement seulement quand et pourquoi un agent va interagir, sans exprimer le traitement des communications directement au niveau de l'agent. Ces mécanismes correspondent aux actions d'interaction évoquées dans le chapitre 9. Nous illustrons leur utilisation dans les exemples de construction de système de la dernière par-

tie de ce mémoire.

De plus, les mécanismes de liaison relatifs aux actions individuelles des agents sont spécifiques à chaque agent et permettent de mettre en place de manières différentes des schémas d'interaction identiques en respectant l'autonomie et les spécificités de ces agents.

L'ensemble de ces mécanismes et des représentations locales (connaissances sociales et interactions initiées) manipulées par les agents contribuent à définir un noyau interactionnel prenant en compte des contraintes organisationnelles (la structure organisationnelle) au niveau des agents. Ce noyau permet au concepteur des agents de s'affranchir de la gestion des communications (interactions) et de l'organisation (préférences).

Chapitre 12 Mise en place d'une réorganisation locale des interactions.

Notre proposition se base sur une description de structures d'interaction qui est distribuée au niveau des agents sous la forme de connaissances sociales, afin de permettre une gestion décentralisée de l'organisation du système. Nous avons présenté dans le chapitre 10 comment décrire et distribuer ces connaissances, puis nous avons présenté, dans le chapitre 11, les mécanismes permettant aux agents d'utiliser de telles connaissances.

L'utilisation de ces connaissances est basée sur la notion de préférence et nous allons présenter dans ce chapitre comment nous proposons de mettre en place des capacités de réorganisation dans les agents afin de modifier la structure d'interaction utilisée par le système.

Nous commençons par présenter dans la section 1, les principes d'évaluation, de jugement et de remise en cause que nous adoptons pour la mise en place locale de la réorganisation. Nous détaillons ensuite dans les sections suivantes ces principes. Dans la section 2, nous expliquons comment nous envisageons l'évaluation du bénéfice d'une interaction et comment mettre en place cette évaluation au niveau des modèles d'agent et des schémas d'interaction. Ensuite dans la section 3, nous présentons comment nous proposons de juger le résultat de cette évaluation. Puis, nous présentons les techniques utilisables pour remettre en cause l'organisation des interactions en adaptant les préférences de chaque agent, avant de faire un bilan sur les principes de réorganisation que nous avons présentés.

1 GÉNÉRALITÉS

Selon un point de vue sociologique [Schm91], une organisation est un ensemble d'intérêts divergents. Elle représente donc un compromis entre les aspirations que ses constituants espèrent voir se réaliser et les concessions qu'ils doivent faire à l'organisation. Un individu reste dans une organisation s'il en retire une contrepartie satisfaisante pour les contraintes qu'il accepte de subir [Fois96a], [Fois96c].

Nous utilisons ce principe pour modéliser la structure organisationnelle et pour permettre aux agents de la réorganiser. Un agent estime le bénéfice qu'il a à participer à une organisation en évaluant le gain qu'il retire de chacune de ses interactions. Lorsque ce gain est considéré comme non satisfaisant ou de peu d'intérêt, la préférence pour l'interaction impliquée est remise en cause.

L'approche du processus de réorganisation que nous proposons est donc résolument locale. Elle se base sur différentes étapes vis-à-vis des interactions utilisées par les agents : l'évaluation du bénéfice retiré de l'interaction, le jugement de ce bénéfice (satisfaisant ou non) et l'adaptation des préférences qui structurent la représentation locale des interactions en fonction de ce jugement.

En ce qui concerne notre modèle, la remise en cause d'une interaction peut être envisagée pour deux raisons distinctes : soit le protocole suivi est jugé inadapté pour mener l'interaction (la manière dont est menée l'interaction est remise en question), soit les autres participants ne satisfont pas correctement ou suffisamment la motivation qui la conditionne (l'utilité, la pertinence des agents est remise en question).

Le propos de ce chapitre n'est pas d'expliciter comment modifier un protocole ou comment changer les participants d'une interaction, mais seulement comment changer les préférences des interactions potentielles. En effet, comme nous l'avons évoqué dans le chapitre 10, décrire une société d'agents organisée n'est pas uniquement définir une structure organisationnelle concrète, c'est aussi décrire un potentiel de structures utilisables par les agents de cette société.

2 EVALUER LE BÉNÉFICE D'UNE INTERACTION

Dans notre approche, réaliser une interaction correspond pour chaque agent à utiliser une connaissance sociale précisant un schéma de conversation. Evaluer une interaction pour un agent consiste alors à évaluer le bénéfice de cette utilisation.

Les connaissances sociales évoquées sont issues de l'instanciation et de la répartition de schémas d'interaction qui peuvent varier en fonction des applications. Il est donc difficilement envisageable de trouver une méthode d'évaluation complètement indépendante de l'application ou du moins indépendante des différents types d'enchaînements d'actions qui peuvent être exprimés par les protocoles d'interactions. L'évaluation, telle que nous la concevons alors est fonction de critères relatifs à l'enchaînement des actions, ou de la nature des informations obtenues par les échanges avec les autres participants à l'interaction.

Ces critères dépendent des résultats des actions entreprises par les autres agents pendant leurs

interactions. Ils ne sont donc mesurables que par la nature et le contenu des messages échangés (ou non) par les participants à une interaction : un agent ne peut percevoir l'activité des autres agents lors d'une interaction que par ce biais dans les systèmes composés d'agents communicants. La mesure de ces critères intervient alors durant l'interaction et peut par conséquent être décrite à un niveau inter-agent au moyen des variables contextuelles.

Nous proposons alors de définir la mesure de ces critères dans l'expression des protocoles d'interaction pour chaque participant abstrait, ceci afin d'exprimer une différenciation de la mesure et des critères en fonction des protocoles et des rôles distincts intervenant dans ceux-ci.

La mesure de ces critères est une perception du déroulement de l'interaction obtenue à partir des messages reçus. Plus précisément, un critère peut être mesuré de deux manières : soit par une valeur extraite du contenu du message (ce qui correspond à une variable dans le modèle de message à recevoir), soit par l'affectation d'une variable en fonction de l'arrivée ou non d'un message (par exemple : une réponse à une question peut être oui ou non, et en fonction de cette réponse un critère pourra être affecté de manière booléenne).

L'évaluation consiste pour un agent à agréger ces critères, afin de calculer le bénéfice retiré d'une interaction. Cette évaluation dépendant de l'application et de la nature des critères mesurés, nous considérons qu'elle correspond à une action individuelle de l'agent. L'appel de cette action est décrit dans le protocole d'interaction sous la forme d'une fonction qui fournit un résultat représentant le bénéfice ainsi calculé. L'évaluation est alors locale à chaque agent puisqu'elle correspond à une action propre de celui-ci. Son mode de calcul pourra différer d'un agent à l'autre en fonction des schémas d'interaction ou des modèles d'agent utilisés (même si cette évaluation se rapporte à des instanciations différentes d'un même schéma d'interaction).

3 JUGER LE BÉNÉFICE D'UNE INTERACTION

Ce bénéfice évalué peut se révéler positif ou négatif [Mart92]. Le problème est alors de donner aux agents les moyens de juger la valeur de ce bénéfice et d'entreprendre en fonction de celle-ci des actions pour adapter leur représentation de l'organisation à la situation.

Nous avons présenté l'évaluation du bénéfice d'une interaction en précisant qu'il s'agissait d'un appel d'action au niveau du protocole d'interaction. Cette action étant une fonction, nous utilisons alors la variable contextuelle résultant de celle-ci pour juger le bénéfice qu'elle représente. Nous proposons alors, tout comme pour l'évaluation, d'exprimer le jugement de l'interaction au niveau du protocole d'interaction au moyen d'un appel à une autre action individuelle de l'agent ayant comme paramètre cette variable correspondant au bénéfice.

Le principe que nous proposons pour exprimer cette action de jugement est d'utiliser des heuristiques pour agir en fonction de ce bénéfice. Différentes techniques peuvent être utilisées pour exprimer ces heuristiques : fonction de seuillage, partitionnement du domaine des valeurs du bénéfice, ou encore mesure d'une distance par rapport à une référence (qui peut elle-même être apprise au cours des interactions), etc. Le but de celles-ci est d'exprimer la décision de l'agent vis-à-vis de la remise en cause de l'organisation des interactions : quand et comment modifier les préférences.

Pour exprimer cette action de jugement, l'agent doit disposer de moyens lui permettant la modification de ses préférences. Nous allons détailler dans la section suivante ces moyens sur lesquels reposent notre gestion de la structure organisationnelle.

4 ADAPTER LES PRÉFÉRENCES

Quelle que soit l'action de jugement utilisée par l'agent vis à vis d'une interaction, les décisions de l'agent consistent à ne rien changer ou à changer la préférence pour une ou plusieurs connaissances sociales.

Ne rien changer ne pose aucun problème ! Mais pour exprimer une ou plusieurs modifications de connaissances sociales, il faut que les agents disposent de mécanismes leur permettant de désigner les connaissances à modifier et de réaliser une adaptation de la préférence pour ces connaissances.

Désigner des connaissances sociales peut être envisagé de plusieurs manières. La plus simple est de désigner la connaissance sociale qui se rapporte à l'interaction qui a été évaluée et jugée. En fonction des applications, d'autres types de désignation de connaissances sociales peuvent aussi être envisagés selon leurs caractéristiques : motivation, participants, etc. Par exemple, l'évaluation d'une interaction peut conduire à affaiblir les préférences pour toutes les interactions ayant la même motivation ou encore en fonction de certains participants.

L'adaptation de la préférence d'une connaissance sociale revient à changer la représentation locale de la structure organisationnelle d'un agent. Nous proposons de concrétiser cette adaptation par un affaiblissement ou un renforcement du niveau de préférence de la connaissance ou des connaissances désignées comme modifiables. Cet affaiblissement ou renforcement peut être fonction d'une constante ou de la valeur du bénéfice.

5 BILAN

La réorganisation locale d'une structure organisationnelle concrète, comme nous l'avons présenté dans ce chapitre, correspond à utiliser une connaissance sociale pour interagir, à évaluer son utilisation et à la juger pour décider comment adapter cette structure organisationnelle des interactions si cela s'avère nécessaire.

Afin de permettre une meilleure appréhension de la mise en place locale de la réorganisation, nous présentons ici un pseudo-algorithme des étapes d'évaluation et de jugement intervenant dans les modèles d'agent et les appels correspondant dans l'expression d'un protocole d'interaction.

Par exemple, appelons Q(C) l'action d'évaluation calculant le bénéfice d'une interaction en fonction d'un ensemble de critères C mesurés au préalable durant l'interaction :

$$Bénéfice = Q(C)$$

L'appel correspondant à cette évaluation dans un protocole d'interaction est alors de la forme :

<Evaluation> = <rôle> : **name** Q (<C>) → <bénéfice>

Où <rôle> désigne le participant abstrait concerné par cette évaluation, Q est le nom de cette fonction d'évaluation, <C> et <bénéfice> désignent respectivement les paramètres et le résultat de cette action au niveau d'un agent.

Une action de jugement peut alors être décrite en fonction de ce bénéfice sous la forme suivante :

J(Bénéfice) {
  Si *Bénéfice > T*
    alors *R(Bénéfice)* *(3)*
    sinon *W(Bénéfice)* *(4)*
  fsi
}

où T est un seuil fixé, et W et R sont respectivement des actions permettant à un agent d'affaiblir et de renforcer le niveau de préférence de la connaissance sociale qui a été utilisée pour l'interaction, proportionnellement au *Bénéfice* qui a été évalué.

L'appel correspondant à ce jugement dans un protocole d'interaction est alors de la forme :

<Jugement> = <rôle> : **name** J (<bénéfice>)

Le renforcement ou l'affaiblissement des préférences permet de gérer la dynamique de réorganisation en adaptant au niveau de chaque agent la structure préférée par celui-ci, ce qui fera émerger globalement une nouvelle structure organisationnelle : chaque changement de préférence effectué au niveau de l'agent ayant une influence sur le choix d'une connaissance sociale lors de l'initiation des interactions suivantes (voir section 1.2 du chapitre 11).

Le principe utilisé pour faire évoluer les connaissances sociales est similaire à celui de l'algorithme du W-learning [Hump95]. Mais contrairement à ce dernier, nous ne considérons pas qu'une période d'apprentissage est suffisante, les agents évaluent, jugent et adaptent en permanence leurs utilisations de leur potentiel d'interaction : les interactions étant jugées négatives à un moment donné pouvant n'être que temporairement négatives ou encore se révéler utiles pour permettre un fonctionnement du système en mode dégradé. Notre mécanisme de réorganisation est donc toujours actif pour réagir à toute perturbation pouvant arriver durant l'activité du système, comme nous le montrerons au travers d'expérimentations dans la troisième partie de ce mémoire.

Chapitre 13 Conclusion

Dans cette partie, nous avons présenté un formalisme original qui intègre une approche descriptive et opérationnelle pour la construction de systèmes multi-agents sur la base du concept d'interaction [Fois98b].

Nous avons proposé un cadre de description permettant de distinguer les concepts d'agent, d'interaction et d'organisation aux niveaux de la description de système multi-agents [Fois98a]. La description de ces concepts se base sur des modèles et des schémas pour décrire une société d'agents organisés représentant le système. La construction d'un système correspond à une description selon l'équation SMA = A ⊕ I où A correspond à la déclaration de modèles d'agent, I correspond à la déclaration de schémas d'interaction et où l'opérateur ⊕ correspond à l'instanciation des modèles d'agent, des schémas d'interaction et de la structure organisationnelle par la même occasion.

L'opérationnalisation d'un système se fonde sur les instanciations de modèles d'agent et de schémas d'interaction pour créer des agents adoptant les comportements décrits et pour répartir dans ces agents les connaissances sociales correspondant aux interactions décrites et à l'organisation de celles-ci.

L'activation de ces agents permet alors d'obtenir un système multi-agents opérationnel composé d'un ensemble d'agents autonomes organisés, capables d'interagir et de se réorganiser, tout ceci grâce aux mécanismes de liaison que nous avons présentés.

Un agent autonome peut alors être représenté par un schéma en couche (cf figure 13-1). Le noyau de ce schéma correspond aux capacités communes à tous les agents quel que soit le modèle qu'ils suivent, par exemple les capacités de communication et d'interaction. La première couche, que nous appelons couche comportementale, regroupe toutes les caractéristiques du

comportement individuel de l'agent dont nous avons proposé la description au travers des modèles d'agent. La deuxième couche est appelée la couche sociale, car elle correspond à toutes les représentations locales permettant à l'agent d'interagir : nous retrouvons dans cette couche les connaissances sociales, et les interactions initiées. Les mécanismes de liaison que nous avons présentés assurent les liens entre ces différentes couches pour le fonctionnement des agents.

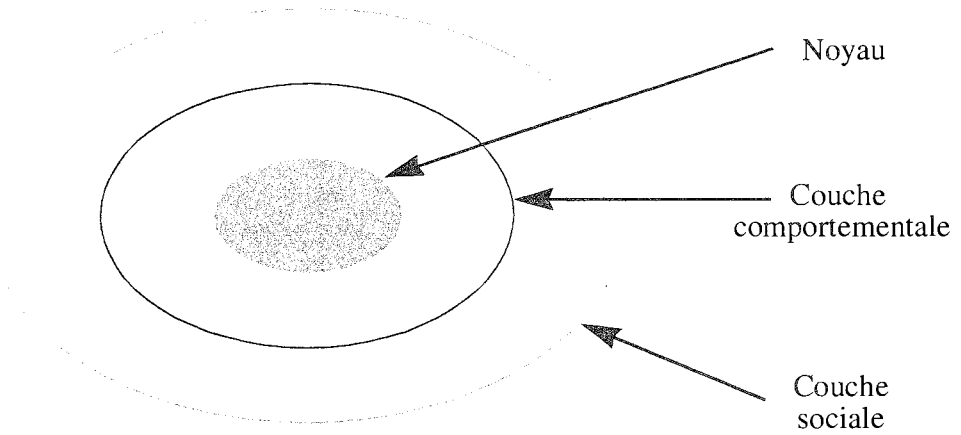


FIGURE 13-1 Schéma en couche d'un agent autonome.

Cette approche descriptive et opérationnelle constitue un premier pas pour la construction aisée de systèmes multi-agents. Les systèmes ainsi construits peuvent être qualifiés d'hétérogènes en faisant abstraction du mode de communication. Cette hétérogénéité peut s'interpréter différemment selon les différents concepts auxquels nous nous référons.

Pour les agents, nous considérons que des agents hétérogènes peuvent constituer le système puisque nous n'imposons aucune architecture du fonctionnement interne pour les agents. Cette architecture peut prendre des formes variées et il nous semble plus important de laisser le choix de celle-ci au concepteur d'un système plutôt qu'en imposer une qui le contraigne trop et qui puisse se révéler inadaptée au domaine d'application visé. Le concepteur peut ainsi définir ou utiliser différents modèles d'agent correspondant à des architectures internes variées au sein d'un même système, ce qui permet de conserver un maximum de souplesse dans la construction de systèmes.

De manière similaire, nous ne considérons pas qu'il est suffisant de fournir au concepteur un ou plusieurs protocoles d'interaction standard. C'est pourquoi, nous avons préféré proposer un cadre de description des interactions et plus particulièrement une description des protocoles d'interaction, afin de permettre aux concepteurs de définir, de modifier ou de réutiliser des schémas d'interaction en fonction des applications développées.

Enfin, l'organisation des sociétés est décrite simplement à partir d'interactions potentielles et de préférences parmi ces interactions, ce qui permet d'envisager une grande variété d'organisations possibles des interactions (aussi bien statiques qu'adaptatives). Cette description prend place à différents niveaux : au niveau de la société en précisant la structure d'organisation préférée parmi les interactions instanciées, mais aussi au niveau des agents eux-mêmes aux moyens d'actions spécifiques permettent d'évaluer, de juger et d'adapter l'utilisation des interactions en modifiant la structure préférée de manière décentralisée.

L'accent est mis par conséquent sur la réutilisation, la modularité et la souplesse de description. Par ailleurs, comme nous nous sommes inspiré de SDL'92 [Færg94] et des automates d'états finis pour la spécification des schémas de conversation, nous envisageons une possibilité d'intégration des techniques de validation de protocole à notre approche.

Cette intégration pourrait se mettre en place via une traduction de nos schémas d'interaction dans le formalisme utilisé pour SDL'92, nous pourrions alors envisager une validation partielle de la structure des interactions des systèmes décrits à partir d'outils dédiés à la validation, comme ObjectGeode [Tata97].

Nous parlons de validation partielle en raison d'une différence essentielle existant entre les processus des systèmes répartis et les agents. Dans les premier les protocoles expriment des échanges de services (qui peuvent être qualifiés de stables vis-à-vis de leur comportement opératoire). Mais dans le cadre des agents, les services prenant place dans les interactions correspondent à des actions qui peuvent évoluer. En effet, l'un des caractères propre à l'intelligence artificielle et de la notion d'agent en particulier est d'intégrer la notion de réflexivité [Ferb89], [Masi89].

La réflexivité est la capacité pour un programme de se modéliser lui-même en termes de descriptions statiques (en accédant à sa propre représentation) et d'exécution dynamique (en contrôlant le contexte de son exécution). Le but de la réflexivité est alors de pouvoir produire des programmes intelligents capables de raisonner sur leur propre comportement et d'être ainsi à même de le modifier [Ferb89]. Cette notion de réflexivité peut être partiellement prise en compte dans notre approche en permettant aux agents de modifier le mécanisme de liaison spécifiant pour chaque modèle d'agent les liens entre les appels d'actions lors de la réalisation des interaction et les actions propres des agents suivants ce modèle. Les interactions seraient alors uniquement validées de manière dynamique puisque les agents peuvent influer sur celle-ci, or SDL'92 ne permet pas de prendre en compte la réflexivité.

Enfin, précisons qu'il existe d'autres travaux qui visent à donner des principes pour établir des descriptions de système à partir de certains concepts et de relier de telles descriptions à l'opérationnalisation des systèmes. Les travaux les plus significatifs dans ce domaine sont pour nous VOWELS [Dema97] et AALAADIN [Gutk98].

L'approche des voyelles (VOWELS) affiche une volonté de description à partir de l'équation SMA = A + E + I + O, ou A correspond aux agents, E à l'environnement, I aux interactions et O à l'organisation. Cette approche nous semble très intéressante bien que le lien entre ce type de description et son opérationnalisation n'est à notre connaissance pas fournit par l'approche.

L'approche utilisée par le modèle AALAADIN est similaire à la notre, bien que la terminologie des concepts ne soit pas tout à fait la même. AALAADIN se base sur les concepts d'agent, de groupe, d'interaction et de rôle, alors que notre approche se base sur les agents, les interactions et les rôles des agents dans les interactions. Notre modèle présente l'inconvénient de ne pas intégrer la notion de groupe au niveau descriptif, mais néanmoins offre l'avantage de prendre en compte la réorganisation des interactions au niveau opérationnel.

PARTIE III MISE EN OEUVRE ET VALIDATION EXPÉRIMENTALE

Chapitre 14 Introduction

Nous avons proposé dans la partie précédente une approche descriptive et opérationnelle de la construction de systèmes multi-agents en présentant les principes de description et les mécanismes permettant d'obtenir des ensembles d'agents autonomes opérationnels. Dans cette partie nous présentons les différentes étapes que nous avons suivies pour vérifier la faisabilité et le bien fondé de notre approche.

La première étape a consisté à implanter et à vérifier la faisabilité d'une représentation partielle de la structure des interactions au niveau des agents et d'une gestion locale de cette représentation pour coordonner leurs interactions. Cette étude de la faisabilité de la couche sociale est présentée dans le cadre des expérimentations et des extensions que nous avons effectuées à partir de GTMAS [Chev93a].

La seconde étape correspond à l'implantation d'une plate-forme dédiée au prototypage de systèmes multi-agents se basant sur notre proposition. Nous en présenterons l'architecture de fonctionnement aussi bien au niveau de l'interprétation des descriptions qu'au niveau du fonctionnement des agents autonomes qu'elle permet de générer.

Nous illustrons ensuite l'utilisation de notre approche pour la conception d'applications multi-agents à partir de notre plate-forme, avant de présenter une évaluation des possibilités de celle-ci et des extensions envisagées.

Chapitre 15 Mise en oeuvre autour de GTMAS

Ce chapitre présente l'outil GTMAS (General Toolkit for Multi-Agent Systems) [Chev93a] et les expérimentations préliminaires que nous avons menées avec celui-ci. Le but de cette première phase expérimentale fût de vérifier le bien-fondé de notre approche des phénomènes interactionnels et organisationnels au niveau du fonctionnement de la couche sociale des agents autonomes.

Après une rapide présentation de l'outil et des modifications que nous y avons apportées, nous présentons les expérimentations que nous avons effectuées. Nous terminerons ce chapitre en tirant les conclusions de cette première étape d'implantation, qui nous ont permis entre autre de concevoir un nouvel outil, plus adapté à notre modèle.

1 DESCRIPTION GÉNÉRALE DE GTMAS

Les objectifs de GTMAS sont le développement, l'évaluation et la comparaison de systèmes multi-agents. GTMAS est un outil générique qui propose des moyens de représentation des agents, de communication par envoi de messages ou par partage d'informations. Enfin, différentes informations fournies par l'outil permettent l'évaluation et la comparaison des applications construites. L'outil propose ces fonctionnalités sous la forme de modules utilisables seuls dans la mesure du possible. Il est alors possible de tester et de mettre au point partiellement chacune des fonctionnalités intervenant dans la future application puis de les intégrer à celle-ci.

GTMAS est conçu comme une boîte à outils dans laquelle l'utilisateur puise les fonctionnalités

qui lui sont nécessaires au gré des modules. Les fonctionnalités proposées possèdent un fonctionnement par défaut qui peut être étendu. Elles sont suffisamment variées pour ne pas être restreintes à une catégorie de systèmes et sont conçues indépendamment du domaine d'application.

Les fonctionnalités primitives fournies par l'outil que nous avons utilisées pour réaliser notre application sont (se reporter à [Chev94] et [Chev93b] pour des informations complémentaires et pour les autres fonctionnalités) :

- Représentation des agents :

La représentation des agents de GTMAS ne propose aucune structuration des agents a priori. Il n'y est fait aucune distinction entre agents selon leur rôle pendant la résolution. Cette représentation peut être étendue par l'adjonction de nouveaux modèles d'agent.

Un agent est décrit par des attributs qui lui sont propres. Il s'agit de :

- son nom, qui permet de le distinguer des autres agents,

- son état interne correspondant à son activité courante (actif, inactif,...),

- son corps, cette composante de l'agent correspond à son savoir-faire et à son comportement ; il s'agit de sa partie procédurale,

- sa boîte aux lettres, elle contient les messages reçus par l'agent,

- ses intérêts, il s'agit d'un prédicat spécifiant les messages que l'agent juge intéressants. Il est utilisé pour certaines formes de communication,

- sa base de faits locale, elle correspond à différentes variables spécifiques à l'agent, comme par exemple, ses accointances. Chaque variable est définie par un nom ; l'accès et la manipulation se font via des fonctions prédéfinies,

- sa partie initialisation, c'est une suite d'actions effectuées lors de la création de l'agent. Cela permet, entre autre, d'initialiser la base de faits.

- Communication :

GTMAS fournit des moyens de communication permettant le partage d'informations et l'envoi de messages. Ces deux fonctionnalités font l'objet de deux modules différents utilisables séparément. Nous présentons ici uniquement l'envoi de message.

La communication par message est supposée fiable : le message n'est pas altéré par sa transmission ; lorsqu'un message est envoyé, il est supposé toujours arriver à son destinataire. Celui-ci ne délivre pas d'accusé de réception.

Un message correspond à un objet structuré. Par défaut, un message est composé d'un contenu et d'un identifiant. L'utilisateur peut définir de nouveaux types de messages selon ses besoins.

Les échanges s'effectuent de deux manières : en point à point ou par diffusion. Nous n'avons utilisé que le mode point à point dans l'application que nous présentons dans ce chapitre.

Les agents reçoivent leurs messages dans leur boîte aux lettres où ceux-ci sont empilés en attendant d'être traités. Il existe deux modes de réception des messages au niveau des agents :

avec ou sans filtrage. Le filtrage consiste, lorsqu'un message est reçu et avant qu'il ne soit stocké dans la boîte aux lettres, à lui appliquer le prédicat spécifiant les intérêts de l'agent. S'il est vérifié, le message est alors effectivement ajouté dans la boîte aux lettres.

GTMAS est donc un outil permettant de décrire un système à partir du comportement et du savoir-faire (connaissances individuelles et possibilités d'actions) des différents agents le composant, avant de rendre cette description du système opérationnelle en lançant les agents. La description du système se réalise donc uniquement à un niveau local, directement au niveau des agents. Nous reviendrons sur les inconvénients d'une telle description dans la conclusion de ce chapitre.

2 MODIFICATION ET EXTENSION DE GTMAS

Afin de vérifier la validité de notre représentation à partir de GTMAS, nous avons apporté quelques fonctionnalités supplémentaires relatives à la gestion d'une base de connaissances sociales (correspondant à l'ensemble des connaissances relatives aux interactions), ainsi que des mécanismes d'initiation et de gestion des interactions.

La base de connaissances sociales correspond à une variable spécifique de la base de faits pour chaque agent. Chacune des connaissances est une structure constituée d'attributs désignant la motivation de l'interaction, les agents concernés, le schéma de conversation utilisé et le niveau de préférence. L'ensemble de ces connaissances est défini a priori pour chaque agent dans sa partie initialisation.

Lorsqu'un agent est motivé pour interagir, il utilise un mécanisme lui permettant d'initier une interaction à partir d'une motivation. Cela consiste, pour l'agent, à réaliser les actions décrites dans l'étape initiale du schéma de conversation, à créer un identifiant unique pour cette interaction, et à mémoriser cette interaction initiée dans une variable spécifique regroupant toutes les interactions initiées de l'agent.

Cette dernière variable mémorise les interactions débutées et non terminées et correspond à une liste. Chaque élément de cette liste reprend les caractéristiques d'une interaction initiée : identifiant de la conversation, schéma de conversation, étape courante dans ce schéma, liste des variables contextuelles utilisées et leurs valeurs, ainsi que la connaissance sociale dont cette interaction est issue.

La structure des messages par défaut de GTMAS a été étendue en ajoutant dans les messages l'identification de la conversation dont ils sont issus, ainsi que l'identifiant de l'expéditeur du message. L'identifiant de la conversation est celui créé par l'agent initiateur de l'interaction. Lorsqu'un agent cherche à exploiter un message de sa boîte aux lettres, il extrait différentes informations de celui-ci : l'identifiant de la conversation et de l'expéditeur. Ces informations sont utilisées par les mécanismes d'initiation et de poursuite d'interaction à partir des messages reçus, dont nous avons déjà évoqué le fonctionnement dans le chapitre 11 de la partie précédente.

3 APPLICATION ET MISE EN ŒUVRE

Pour vérifier la validité de notre approche locale, il nous fallait une application où l'aspect dynamique de l'organisation soit présent. Rappelons que, pour nous, la dynamique d'organisation a pour objectif principal l'adaptation de l'organisation. Chaque agent doit être capable d'adapter ses interactions aux autres agents, tout en maintenant une efficacité acceptable du système. Ceci est possible dans notre modèle, grâce aux capacités permettant aux agents d'adopter par eux-même des utilisations différentes de leur potentiel d'interactions (la base de connaissances sociales).

Afin d'étudier les propriétés de notre modèle pour la construction collective d'une structure, nous l'avons appliqué sur un problème de flux multiples de marchandises [Well93], [Müll95] que nous présentons dans la section suivante. Nous décrivons ensuite la société type que nous avons utilisée pour expérimenter notre modèle. Puis nous expliquons les interactions et la dynamique utilisées dans nos expérimentations, avant d'expliciter leur mise en place au niveau des agents, de présenter nos expérimentations, et les résultats obtenus en terminant par un commentaire de ces résultats.

3.1 Problème de flux multiples de marchandises

Une société est composée d'un ensemble de producteurs et consommateurs : chaque agent a une activité qui consiste à trouver les biens ou fournitures (nous désignerons ces biens comme le stock entrant dans l'agent) qui lui sont nécessaires pour produire à son tour d'autre biens (que nous désignerons comme le stock sortant de l'agent) qu'il pourra fournir à d'autres agents.

Chaque agent a la capacité de gérer sa dotation en biens et de produire à chaque cycle de nouveaux biens selon sa propre fonction de production. Ces agents peuvent être assimilés par exemple à des entreprises et chaque cycle à un mois d'activité de l'entreprise. L'activité de production des agents correspond à activer au maximum une fois par cycle sa fonction de production pour consommer une certaine quantité de biens du stock entrant afin de les transformer en une certaine quantité de biens du stock sortant (nous parlerons de capacité de production).

L'objectif que nous envisageons alors pour des systèmes composés de ces agents est d'obtenir une activation, la plus fréquente possible, de la fonction de production des différents agents, avec un minimum de communication entre les agents. Cette dernière contrainte nous a amené à mettre en place cette application sans utiliser la diffusion de messages.

Les agents utilisent deux types d'interaction pour simuler le flux des biens : un type d'interaction pour offrir les biens de leur stock sortant et un autre pour demander des biens pour leur stock entrant. Les propositions de biens se basent sur le niveau du stock sortant et peuvent être acceptées ou refusées par le consommateur potentiel, alors que les demandes de biens se basent sur des besoins pour produire qui correspondent à des quantités de biens. Une demande ne peut être refusée, mais étant donné que les agents ont une activité cyclique de production qui est limitée par leur capacité de production, elle sera satisfaite dans un certain nombre de cycles. Ce nombre de cycles (supérieur ou égal à 0) est désigné sous le terme de délai de livraison et dépend de la quantité demandée, de la quantité disponible et de la capacité de production du fournisseur.

Dans cette application, chaque agent a trois buts : minimiser le niveau de ses stocks et les délais de livraison de ses fournisseurs, tout en activant sa fonction de production le plus souvent possible.

Selon le premier but, les agents vont proposer à la fin de chaque cycle la quantité de bien excédentaire. L'agent recevant une de ces offres l'acceptera (ou la refusera), si son stock entrant du bien proposé est déficitaire. Cet excédent ou ce déficit dépend d'un seuil maximal et minimal pour chaque type de bien ;

Selon le deuxième but, les agents vont livrer, le plus rapidement possible, des biens de leur stock sortant que d'autres agents leur demandent. Si la quantité demandée n'est pas disponible et que le stock entrant est insuffisant pour produire les biens demandés, l'agent sollicité devra alors à son tour demander à ses fournisseurs les biens entrant manquant, le délai de livraison, qu'il annoncera, dépendra alors des délais que ses fournisseurs lui annonceront.

Enfin, selon le troisième but, les agents vont activés leur fonction de production le plus souvent possible, afin d'approvisionner régulièrement leur stock sortant pour fournir rapidement les demandes éventuelles qu'ils peuvent recevoir.

Les interactions entre les agents sont motivées par le type de biens qu'un des agents veut demander ou offrir :

- si un agent n'a pas le stock entrant nécessaire pour produire, il utilise un protocole de demande (en tant qu'initiateur, voir figure 15-2) où la réponse finale est un message simulant la livraison de ce qui a été demandé ;

- si un agent a un niveau de stock sortant élevé pour un bien donné, il utilise un protocole d'offre (en tant qu'initiateur, voir figure 15-2), où le dernier message échangé correspond à une livraison.

Les échanges modélisés par nos deux protocoles d'interaction s'effectuent au cours du même cycle, excepté pour le premier protocole où un délai de livraison intervient. Dans ce cas, le protocole est suspendu jusqu'à la fin du délai : c'est-à-dire le cycle où la quantité demandée a été promise et sera livrée.

3.2 La société

La société que nous avons utilisée pour nos expérimentations est composée d'agents ayant les caractéristiques que nous venons de présenter. Nous pouvons distinguer trois types d'agents dans cette société, correspondant à trois comportements particuliers : les agents du niveau initial, les agents du niveau final et agents des niveaux intermédiaires.

Les agents de type initial (il s'agit des agents Agt 0-0 et Agt 0-1 sur la figure 15-1) permettent d'assurer la régularité des approvisionnement en biens de la société. Ces agents créent des biens à chaque cycle, afin de fournir (en fonction des demandes ou par des offres) les agents qui dépendent d'eux. Ils peuvent être assimilés à des agents auto-générateurs de biens puisqu'ils produisent à chaque cycle des biens sans devoir se soucier de l'approvisionnement de leur stock entrant.

Les agents de type final (il s'agit de l'agent Agt 3-0 sur la figure 15-1) permettent de créer un effet moteur. Cela consiste pour eux à tenter d'activer leur fonction de production à chaque cycle pour augmenter leur stock sortant. Si cette fonction de production ne peut être activée, ces agents demandent alors les biens nécessaires à leurs fournisseurs, ce qui peut engendrer des demandes récursives jusqu'aux agents du niveau initial. Par ailleurs, le niveau du stock sortant de ces agents permet d'avoir une représentation de l'efficacité globale du système en terme de circulation des flux de marchandises.

Les agents du niveau intermédiaire respectent les caractéristiques générales que nous avons précisées dans la section précédente et relient les agents du niveau initial à ceux du niveau final en assurant l'échange et la transformation des biens.

Nous avons utilisé cette société dans nos différentes expérimentations, afin de pouvoir en comparer les résultats. La figure 15-1 présente cette société. Les valeurs au-dessus (respectivement en-dessous) de chaque agent correspondent à la quantité et au type de biens consommés (respectivement produits) par la fonction de production de l'agent. Les traits entre les agents représentent les liens potentiels d'interaction donnés aux agents lors des expérimentations. Ces interactions sont caractérisées par le bien échangé qui en constitue la motivation.

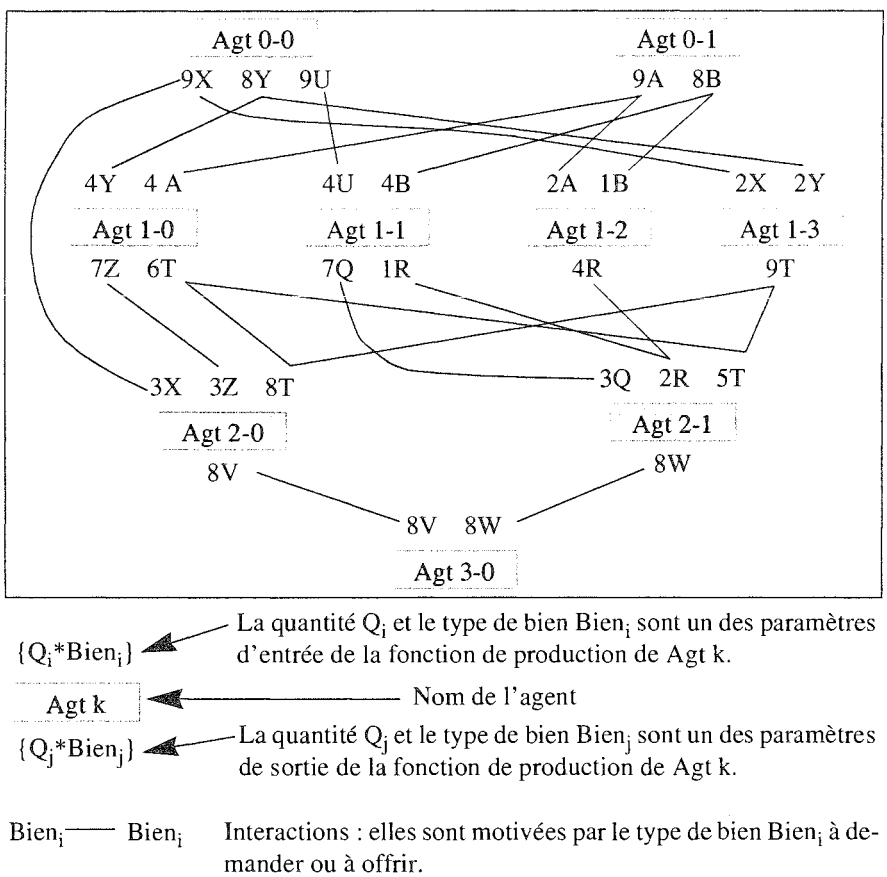


FIGURE 15-1 : La société des agents utilisée à chaque expérimentation.

La configuration de cette société n'est pas totalement aléatoire. Elle respecte une contrainte que nous avons évoquée dans la section précédente : obtenir une production la plus fréquente possible pour chaque agent. Cette configuration a été choisie de telle manière que chaque agent soit indispensable à l'activité globale de par sa capacité de production.

3.3 Les interactions des agents et la dynamique des interactions

Les deux types d'interactions (demander et offrir) utilisées pour cette application peuvent être décrites pour chaque agent, en fonction de la nature du type du bien motivant l'interaction.

Si l'agent ne dispose pas d'une quantité suffisante de bien entrant pour produire, il est alors motivé par la nécessité d'obtenir ce bien. Cette nécessité pourra être satisfaite de deux

manières : par une demande ou en acceptant les offres de ce type de bien. Dans le premier cas, l'agent est initiateur d'une interaction pour demander ce bien manquant selon le schéma de conversation P1.1 du protocole de demande désigné par P1. Dans le second cas, l'agent est sollicité (consommateur d'offre) par des interactions d'offres de ce type de bien selon le schéma de conversation P2.2 du protocole d'offre désigné par P2.

De manière réciproque, si l'agent dispose de biens dans son stock sortant, il est alors motivé par la nécessité de fournir ces biens. Cette motivation pourra prendre forme de deux manières : par une offre de ce bien ou en allouant ce stock disponible aux agents le demandant. Dans le premier cas, l'agent est initiateur d'offre concernant ce type bien selon le schéma de conversation P2.1 du protocole P2. Dans le second cas, l'agent est sollicité (serveur de demande) pour ce type bien selon le schéma de conversation P1.2 du protocole P1.

La dynamique d'organisation consiste alors à utiliser des heuristiques évaluant la qualité des interactions durant la réalisation du schéma de conversation en se basant sur l'acceptation ou le refus des offres ((2) dans la figure 15-2), et sur le délai annoncé ((1) dans la figure 15-2) pour les demandes.

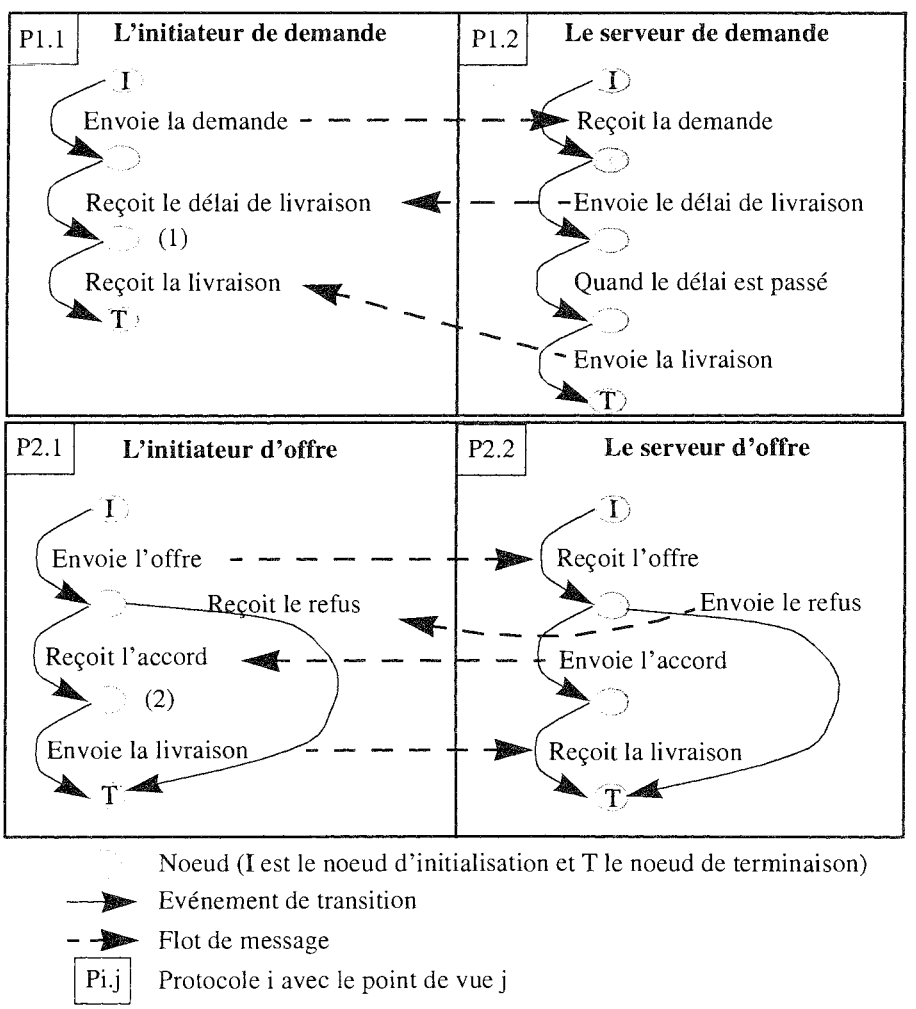


FIGURE 15-2 : Représentation de deux protocoles bipartites (de demande et d'offre).

Ces évaluations sont effectuées par des fonctions (dénotées $Q(C)$, où C est le critère mesuré) estimant la qualité d'un interaction. Par exemple, nous utilisons dans cette application pour le

délai de livraison (voir (1) dans la figure 15-2) la fonction d'évaluation suivante : $Q(C) = 1/C$ pour C dans $[1, +\infty[$ et $Q(0)=1$. Le délai annoncé est donc considéré comme idéal s'il est de 0 ou 1 cycle, et les délais les plus courts sont favorisés. Un autre exemple pour les offres serait d'utiliser comme fonction d'évaluation (voir (2) dans la figure 15-2) : $Q(C) = C$ ou C vaut 0 quand il y a eu refus et 1 en cas d'accord. L'agent favorise ainsi les interactions où l'interlocuteur accepte souvent les offres qu'il lui fait.

Ces fonctions d'évaluation sont utilisées par les agents pour juger leurs interactions et mettre à jour leurs préférences pour ces interactions grâce à une fonction de décision (ou fonction de jugement) :

Si $Q(C) > T$
alors $p = R(Q(C), p)$,
sinon $p = W(Q(C), p)$.

où T est un seuil fixé (0.9 dans nos expériences), et où W et R sont respectivement les fonctions d'affaiblissement et de renforcement de la préférence de l'interaction (abrégée par p), proportionnellement à la fonction d'évaluation $Q(C)$ de l'interaction.

Les préférences des interactions évoluent en fonction des interactions utilisées et des jugements portés lors de ces utilisations. Ces préférences sont valuées dans l'intervalle [min, max] où le min (respectivement le max) correspond au niveau de préférence le plus bas (respectivement le plus haut). Les agents influencent ainsi leurs interactions en réordonnant de manière dynamique leur potentiel d'interactions.

3.4 Mise en place de la représentation des interactions dans les agents

Une société décrite par un graphe d'interaction (voir figure 15-1) doit être représentée localement dans les différents agents du système multi-agents, selon les quatre types de schémas de conversation P1.1, P1.2, P2.1 et P2.2. La figure 15-3 illustre cette représentation locale de connaissances sociales pour l'agent Agt2-1.

Pourquoi ?	Comment ?		
	Avec qui ?	Rôle	Schéma
offrir T	Agt 1-0	S	P2.2
offrir T	Agt 1-3	S	P2.2
offrir R	Agt 1-1	S	P2.2
offrir R	Agt 1-2	S	P2.2
offrir Q	Agt 1-1	S	P2.2
demander W	Agt 3-0	S	P1.2
demander T	Agt 1-0	I	P1.1
demander T	Agt 1-3	I	P1.1
demander R	Agt 1-1	I	P1.1
demander R	Agt 1-2	I	P1.1
demander Q	Agt 1-1	I	P1.1
offrir W	Agt 3-0	I	P2.1

FIGURE 15-3 : La base de connaissances sociales de l'agent Agt 2-1.
1) Le rôle de l'agent dans l'interaction est représenté par I si il est initiateur et par S si il est serveur.
2) Les schémas de conversation sont désignés par leur référence dans la figure 15-2.

3.5 Expérimentations et résultats

Nous avons mené trois types d'expérimentations pour évaluer les propriétés suivantes : la stabilité des structures construites (devant être insensibles à des changements mineurs : par exemple, le changement de la quantité de bien T produit par l'agent Agt1-3 qui passerait de 9 à 8 (voir figure 15-1)), la justesse (quand une structure est trouvée, est-elle opportune pour le problème à résoudre) et l'adaptabilité (être capable de trouver une nouvelle structure (stable) quand des changements majeurs interviennent).

Toutes les expériences, que nous avons présentées, ont été conduites sur 250 cycles d'activation des agents et les résultats présentés sont une moyenne sur dix exécutions pour chacune des expériences menées.

Dans les premières expérimentations, nous avons utilisé deux types de structure organisationnelle pour initialiser le système :

- La première est la société décrite dans la figure 15-1 dans laquelle les préférences des interactions sont assignées aux agents de manière à associer chacun d'eux aux producteurs adéquats (nous désignons cette expérience par 1.1).

- La seconde est une mauvaise structure organisationnelle. Les préférences des interactions sont assignées aux agents de manière à associer les producteurs les moins adéquats avec le meilleur niveau de préférence (cette expérience est désignée par 1.2).

Pour ces structures, les niveaux de préférences peuvent varier dans l'intervalle [*min*, *max*]. Les niveaux de préférences sont positionnés à la valeur initiale *max* (dans le cas ou l'interaction correspondante est favorisée au départ), et *min* (dans le cas ou elle ne l'est pas).

La première série d'exécutions a été menée avec une organisation statique dans laquelle il n'y avait aucune possibilité de réorganisation (aucune fonction de jugement des interactions n'était fournie aux agents). Dans la second série, nous utilisons notre modèle pour observer l'effet de la réorganisation. Nous désignons les expériences de cette série respectivement par 2.1 et 2.2 (reprenant les structures organisationnelles présentées).

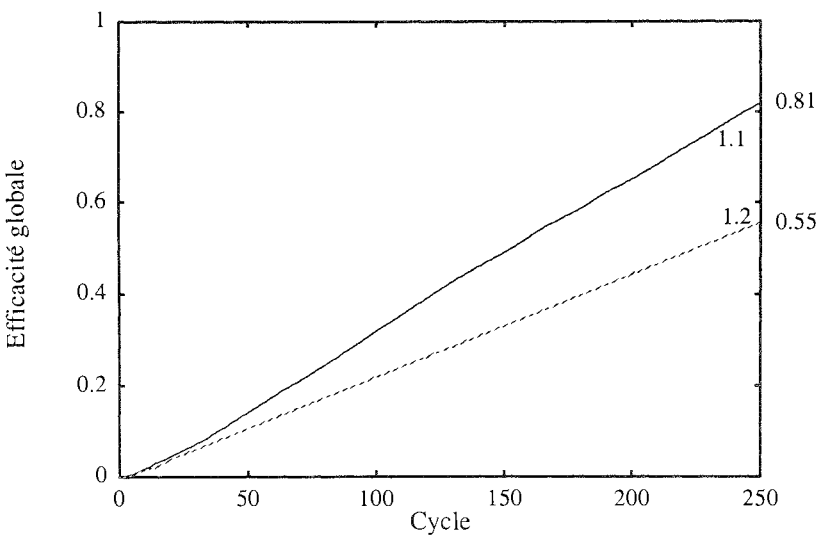


FIGURE 15-4 : Efficacité globale dans les expériences 1.1 et 1.2.

Pour comparer les résultats des exécutions, nous avons mesuré l'efficacité globale en terme de productivité (voir figure 15-4 et 15-5) qui correspond à la somme des agents activant leur fonction de production et cela à chaque cycle. Le cas idéal devrait être la courbe x=y signifiant que chaque agent est actif à chaque cycle.

Pour juger la capacité de trouver une structure et de la conserver, nous mesurons sur les expériences 2.1 et 2.2 l'évolution de l'organisation en cumulant le nombre de changement de connaissance sociale préférée à chaque cycle pour tous les agents. Ceci dénote les changements locaux de structure et les résultats sont présentés dans la figure 15-6.

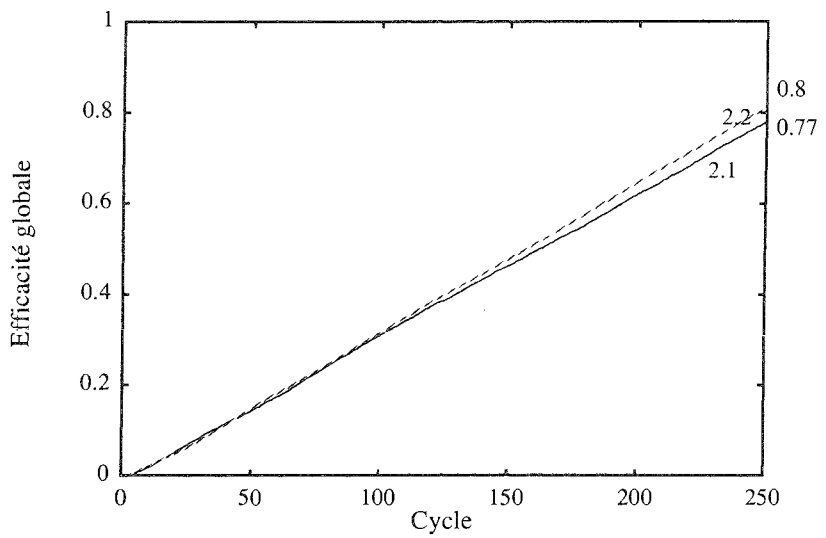


FIGURE 15-5 : Efficacité globale dans les expériences 2.1 et 2.2.

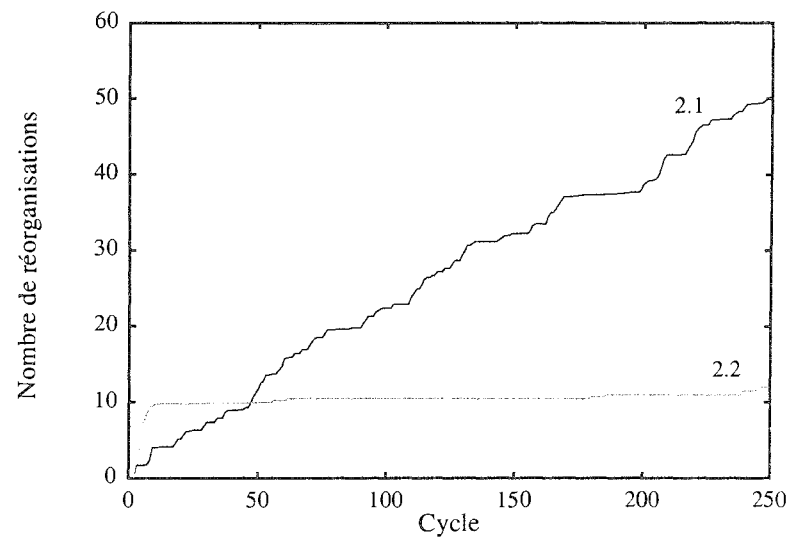


FIGURE 15-6 : Evolution de l'organisation dans les expériences 2.1 et 2.2

Dans la troisième et dernière série d'expériences, nous introduisons une perturbation dans le système : certains agents changent leur fonction de production (en terme de quantité de bien produit) lors du 80ème cycle. Cette perturbation consiste à une changement de la quantité de bien T

produit par les agents Agt 1-0 et Agt 1-3 (parce que leur valeur constituait un frein majeur de l'activité globale dans les expériences de type 1.2 et 2.2), ils sont respectivement de 9 et 5 après la perturbation.

Les figures 15-7 et 15-8 présentent l'activité de production du système selon le temps, respectivement pour l'expérience 1.2 et 2.2 avec la perturbation au 80ème cycle. Cette activité correspond au pourcentage d'agents activant leur fonction de production à chaque cycle (le cas idéal étant de 100% à chaque cycle).

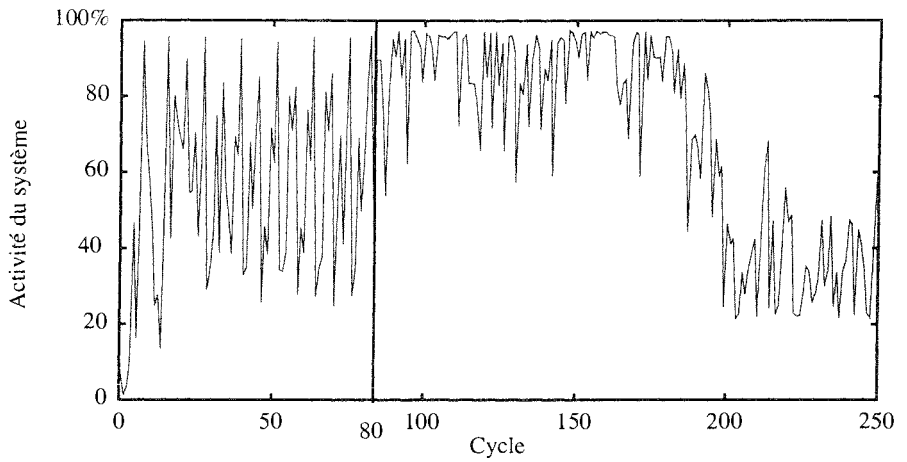


FIGURE 15-7 : Pourcentage d'agents étant actifs à chaque cycle (organisation statique)

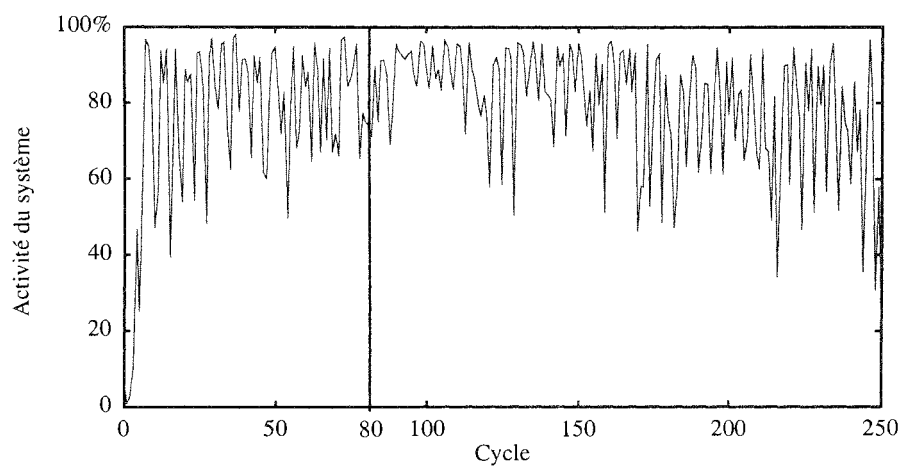


FIGURE 15-8 : Pourcentage d'agents étant actifs à chaque cycle (organisation dynamique).

3.6 Discussion

A propos de la justesse des structures construites, les expériences montrent que notre modèle peut améliorer la performance du système quand nous utilisons des structures initiales non-optimales ; en effet quand nous utilisons des structures optimales (comme c'est le cas pour l'expérience 2.1), le système n'est pas capable d'améliorer la performance globale. Pour être plus précis, dans le premier cas, les performances sont augmentées de 45%, et dans le second, elles baissent de 5% à cause de la difficulté (ou plutôt l'impossibilité) à trouver des interactions meilleures quand le système a initialement déjà les meilleures.

La courbe de réorganisation de l'expérience 2.1 (voir figure 15-6) illustre cette difficulté en montrant le nombre de réorganisations exécutées par le système pour essayer de trouver une meilleure structure dans le potentiel des interactions. Le système cherche à améliorer la structure initiale (même si c'est impossible). Localement, ceci correspond à des tentatives de chaque agent pour trouver un interlocuteur plus adéquat. Les agents interagissent alors temporairement avec des interlocuteurs plus mauvais. Il est difficile pour les agents de se rendre compte localement que la structure initiale était la bonne. Par contre dans l'expérience 2.2 (voir figure 15-6), le système réalise un grand nombre de réorganisations au début de son activité et beaucoup moins après. Il est donc préférable d'utiliser notre modèle avec une structure initiale éloignée de celle optimale.

L'adaptabilité des structures construites avec notre modèle est bien illustrée par la troisième série d'expériences. Tout d'abord, nous pouvons constater que le système peut améliorer la performance globale de la société en adaptant localement la structure organisationnelle. Ensuite, quand un changement majeur se produit (certains agents ont des changements significatifs dans leur comportement), les résultats montrent que l'utilisation de la dynamique choisie permet de réduire l'impact de ces événements sur l'activité globale du système (voir figures 15-7 et 15-8). La figure 15-7 montre que la mauvaise performance initiale dans le cadre de l'organisation statique est améliorée sous l'effet de la perturbation, mais cette amélioration n'est que temporaire. En effet, cette perturbation pallie à la mauvaise répartition des biens T qui pénalisait l'activité globale du système, mais d'autres modes de répartitions pénalisaient à des degré moindre cette activité. Par exemple la préférence entre les interactions pour les biens de type B entre les agents Agt0-1 et Agt1-2 permet certes de produire réguliérement une quantité suffisante de biens de type R, mais ceci au détriment de la production des biens de type Q par l'agent Agt1-1. L'effet de ces mauvais modes de répartition se traduit au niveau de la courbe d'activité par la chute des performances après le cycle 190.

En ce qui concerne la stabilité, nous allons tout d'abord clairement la définir avant de juger si notre modèle remplit ou non cette propriété. La stabilité signifie qu'une structure donnée demeure (elle ne change pas l'ordre local des préférences pour les interactions) durant une période de temps donnée. Si nous considérons les résultats de la figure 15-6, nous constatons que notre modèle est capable de fournir une structure stable si la période de temps considérée est autour de 20 ou 30 unités et si la structure initiale est non optimale. Si nous considérons les 250 cycles comme période de temps, il apparaît clairement qu'il n'y a pas de stabilité dans l'expérience 2.2.

Après toutes ces expériences, nous constatons que notre modèle possède certaines propriétés intéressantes. Premièrement, il permet l'amélioration du comportement global du système par jugement local quand des structures organisationnelles non-optimales sont utilisées initialement. Deuxièmement, il permet la construction de systèmes qui peuvent s'adapter eux-mêmes en réponse à des changements de leur environnement (modification du comportement des autres agents). La stabilité est difficile à atteindre parce que deux paramètres sont concernés : la période de temps de référence qui définit la stabilité, et le type de structure initiale fournie au système.

Selon les résultats fournis jusqu'ici, il semble difficile d'obtenir une organisation optimale ou une organisation stable quelque soit le cas. De même, certains travaux ont prouvés qu'une organisation optimale est difficile, voir impossible à trouver [Galb73], le modèle et l'approche que nous proposons est néanmoins intéressant dans deux situations :

- quand il est difficile de décider globalement si une organisation est convenable ou non,

- quand l'environnement du SMA est fluctuant et implique que les agents adaptent dynamiquement la structuration du système aux fluctuations.

4 CONCLUSION

Les expérimentations décrites dans ce chapitre ont permis de montrer que la représentation d'une organisation à partir d'un ensemble d'interactions potentielles, plus ou moins privilégiées par les agents, permet d'obtenir certaines propriétés globales intéressantes pour des systèmes : justesse, stabilité, et surtout adaptabilité.

Ces expérimentations ont été menées uniquement dans le cadre d'un jugement local d'interactions où ce jugement est effectué par l'agent initiateur de l'interaction. Il serait alors intéressant d'approfondir l'étude de ces propriétés dans le cadre d'un jugement où l'agent serait sollicité ou initiateur. Néanmoins, ces premières expérimentations soulèvent différentes questions : Comment conserver la structure optimale fournie au départ ? Quel est l'impact des contraintes locales de l'application sur la stabilité et la convergence de l'adaptation du système ? Les deux protocoles utilisés semblent être complémentaires, quels seraient les résultats obtenus avec un seul type de protocole, ou encore avec un protocole différent ?

Il ressort de nos expérimentations que la justesse et la stabilité de l'organisation sont difficiles à assurer à partir d'un jugement purement local des interactions. En effet, celui-ci se base sur la mesure d'un critère qui évolue en fonction des jugements portés de manière indépendante par les agents. Le consensus n'est donc pas forcément possible et dépend fortement de la structure de départ fournie aux agents composant le système. Néanmoins, l'avantage de cette approche est d'obtenir un système capable de s'adapter aux changements influençant ce critère.

Ces remarques font clairement apparaître la nécessité de conduire une étude plus poussée de la réorganisation locale des interactions selon différents aspects de l'application que nous avons traitée [Fois97]. Mais certains problèmes majeurs nous ont conduit à envisager une autre suite pour nos travaux : la construction d'un système autrement que par une déclaration uniquement locale des interactions. Nous avons donc étendu notre approche de la réorganisation et des interactions pour prendre en compte la description globale et semi-globale des systèmes (voir partie II) et ainsi développé une nouvelle plate-forme pour le prototypage de SMA pour les raisons suivantes :

- GTMAS ne prend pas en compte l'aspect global et semi-global de la description des applications. La modification des implantations, que nous avons réalisées pour nos expérimentations, est alors difficile et fastidieuse pour prendre en compte de nouvelles structures initiales. En effet, GTMAS ne permet qu'un développement de SMA à partir d'une description des connaissances sociales dans la partie initialisation des agents et ne fournit pas de possibilités d'expression et d'utilisation pour les protocoles d'interaction. Il est alors très difficile de changer les protocoles utilisés puisque ceux-ci sont définis uniquement à partir des schémas de conversation les composant ;

- Les expérimentations autour de GTMAS nous ont amené à constater un manque de fiabilité

de cette boîte à outils. Ce manque de fiabilité est du à un bogue de l'interpréteur du langage Common Lisp de Sun, langage support de GTMAS. Ce bogue est lié au mécanisme de changement de contexte de la gestion multi-tâches des processus Lisp qui est utilisée dans GTMAS pour l'opérationnalisation des agents. Il se traduit par des pertes inexpliquées de contexte empêchant l'exécution de certains agents et pénalisant tout le système (puisque GTMAS est conçu sur l'hypothèse de fiabilité des communications). Ces pertes de contexte sont peu fréquentes, mais le risque de leur avènement est d'autant plus élevé quand le système comprend plus d'agents ou que le fonctionnement du système entraîne de nombreux changements de contexte.

Ces problèmes nous ont amené à développer une nouvelle plate-forme dédiée à notre approche de la construction de SMA prenant en compte la description globale d'une société et les descriptions semi-globales des interactions y prenant place. Nous allons présenter cette nouvelle plate-forme JAVAMA dans le chapitre suivant.

Chapitre 16 La plate-forme JAVAMA

Ce chapitre présente la plate-forme JAVAMA (une boîte à outils en JAVA pour prototyper des systèmes Multi-Agents) que nous avons développée pour décrire et implanter des systèmes multi-agents. Cette plate-forme se présente sous la forme d'une API MultiAgent (interface de programmation pour les applications multi-agents) définissant les outils de description, les mécanismes de mise en place et de fonctionnement des agents et de leurs interactions.

Nous commençons par donner les objectifs que nous avons adoptés pour cet outil. Puis, nous donnons une description générale de l'outil, avant d'énumérer les différentes fonctionnalités qu'il propose à un concepteur au travers de l'API MultiAgent. Enfin, nous terminons par un bilan de cette boîte à outils et proposons les extensions futures que nous envisageons.

1 OBJECTIF DE L'OUTIL

L'objectif de JAVAMA est le développement de systèmes multi-agents clos basés sur la communication directe. Ce développement se fonde sur une description déclarative [Marc97] d'un système comprenant la déclaration des agents le composant, la déclaration de leur comportement à un niveau local (modèle d'agent), la déclaration des schémas d'interaction et de leurs instanciations entre les composants. L'outil doit permettre de rendre automatiquement ce type de description opérationnel, mais aussi réutilisable partiellement ou totalement lors de développement futur.

JAVAMA a été conçu dans le but de permettre un développement des systèmes multi-agents modulaire et incrémental basé sur une collection de descriptions, et aussi afin de permettre une

réutilisation et une modification aisées de ces descriptions. Cette collection de descriptions a pour objectif principal de faciliter la description des phénomènes complexes intervenant dans les systèmes et de permettre le réajustement éventuel de ces phénomènes selon l'application aussi bien au niveau global (société), semi-global (interaction), que local (agent).

2 DESCRIPTION GÉNÉRALE

JAVAMA est une boîte à outils générique qui propose des moyens de définition d'un système (où aucun agent ne peut être ajouté ou tué pendant la vie de celui-ci : nous ne décrivons que des systèmes appelés clos par opposition aux systèmes ouverts). Ces moyens de définition correspondent à ceux de notre modèle : modèles d'agent, schémas d'interaction, instanciations de ceux-ci, structuration des interactions et réorganisation de celles-ci.

Cette plate-forme fournit par conséquent l'embryon d'un langage déclaratif de système par les moyens de description qu'elle propose et utilise JAVA comme langage support pour interpréter les éléments décrits et construire un système opérationnel.

La description d'un système comporte trois phases :

- Description des modèles des agents : ceci consiste à décrire les comportements, les connaissances individuelles et les actions propres des agents à partir d'un noyau d'agent définissant les mécanismes de communications et d'interactions utilisables ;

- Description des schémas d'interaction : ceci consiste en une description générique et abstraite basée sur des prototypes de protocoles d'interaction. Cette description consiste à identifier la motivation du schéma d'interaction, et le protocole d'interaction associé qui décrit un enchaînement d'action en se basant sur la notion de rôle ;

- Description de la société des agents du système, c'est-à-dire réaliser une composition du système à partir des modèles cités ci-dessus, ce qui consiste à :

    - décrire les agents à partir des différents modèles d'agent décrits, en instanciant ceux-ci et en identifiant chacune de ces instanciations de manière unique dans le système ;

    - décrire les interactions entre ces agents, c'est-à-dire instancier les différents schémas d'interaction, en désignant les identifiants des agents qu'ils mettent en relation, les rôles joués par ces agents vis-à-vis du schéma, et en précisant la préférence de chacun de ces agents pour ces instanciations.

Ce langage de description permet de spécifier un système à partir de modèles d'agent et de schémas d'interaction, et de préciser les structures potentielles (les interactions possibles) pour cette société d'agents, ainsi que la structure organisationnelle préférée par les agents.

Outre ce langage de description permettant de spécifier la composition d'un système, JAVAMA permet évidemment d'interpréter et de rendre opérationnel de telles descriptions pour obtenir l'ensemble des agents autonomes formant le système multi-agents.

L'interprétation de ces descriptions est explicitée par la figure 16-1. Dans cette figure, nous présentons le lancement de la plate-forme pour une application correspondant à une description

de système multi-agents nommée X. Cette description est interprétée à un niveau macroscopique dans un premier temps pour déterminer les différentes caractéristiques des interactions potentielles entre chacun des agents et les structures d'interaction du système. Ces caractéristiques peuvent alors être utilisées pour construire les connaissances sociales de chaque agent, avant d'interpréter à un niveau microscopique les caractéristiques des modèles d'agent pour créer des agents autonomes combinant ces caractéristiques aux connaissances sociales construites par l'interpréteur macroscopique. Nous obtenons ainsi un système opérationnel.

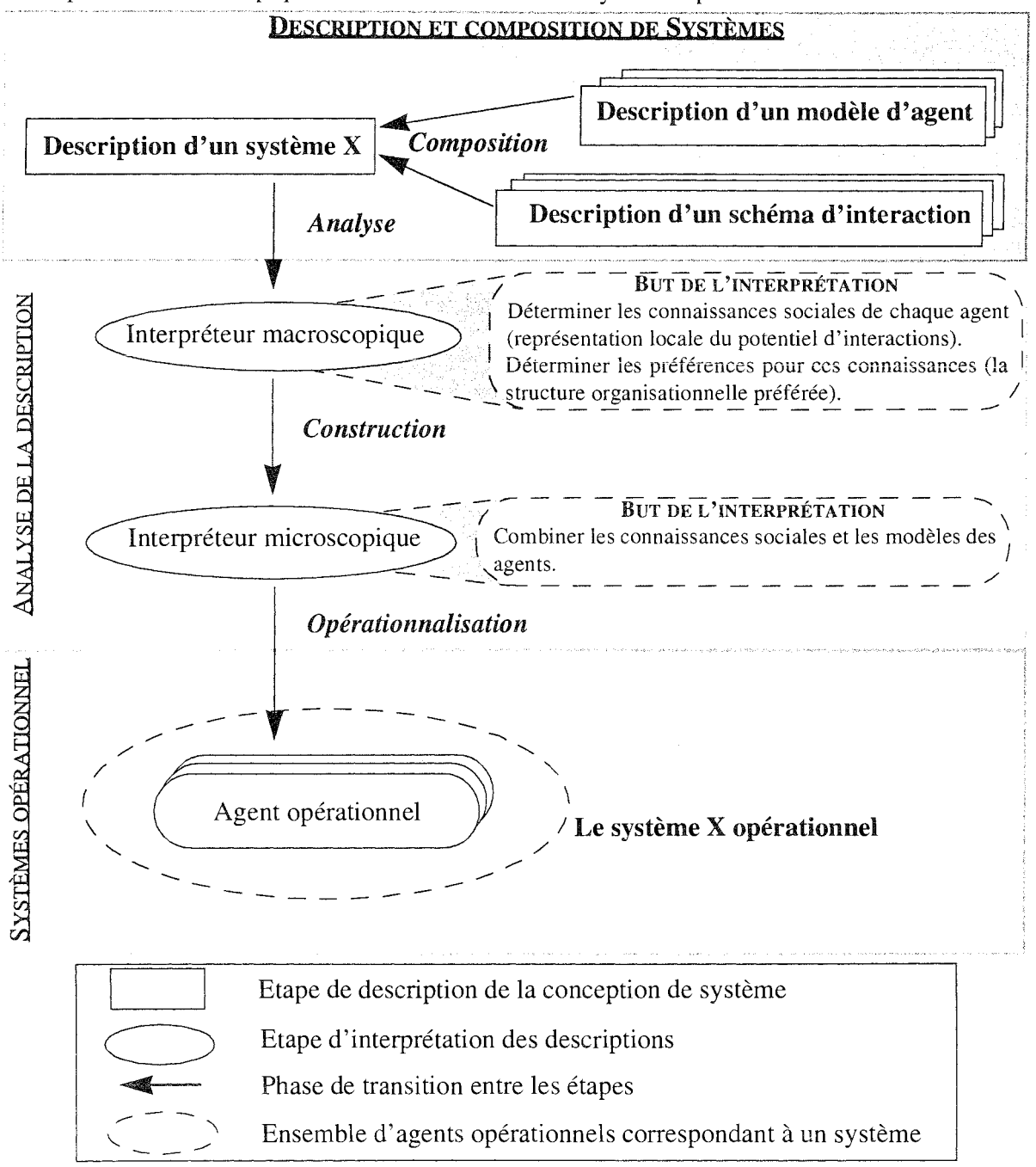


FIGURE 16-1 : Schéma de fonctionnement général de la plate-forme.

3 LES FONCTIONNALITÉS EXISTANTES ET L'ARCHITECTURE

La plate-forme se présente sous la forme de différents modules (ou packages Java) regroupant les outils de base pour la description de systèmes, l'interprétation de cette description et le fonctionnement des agents créés et rendus opérationnels par cette interprétation.

Nous présentons ici ces modules en fonction de la nature des outils qu'ils proposent :

- des outils utilisés pour la description et l'interprétation automatiques de cette description :

  - le module *MultiAgentSystem* regroupe les fonctionnalités permettant de décrire la composition d'un système multi-agents, et de construire une représentation globale de celui-ci en terme d'agents et d'interactions entre ceux-ci. Ce module comprend également les fonctionnalités de lancement de la plate-forme et d'analyse automatique d'une représentation d'un système car l'interpréteur macroscopique est défini dans ce module.

  - le module *LogicalAgentModel* regroupe les fonctionnalités permettant de décrire des modèles d'agent, ainsi que des mécanismes de liaisons entre les appels d'actions pouvant intervenir dans les schémas de conversation et le modèle de l'agent. Il permet aussi de rattacher ce modèle d'agent à un modèle physique d'exécution (que nous avons évoqué dans la partie précédente en parlant de noyau) comprenant toutes les fonctionnalités internes nécessaires à un agent autonome (action de communication, d'interaction, etc.),

  - le module *Agent* regroupe les fonctionnalités permettant de construire une représentation d'un agent pour exprimer la composition d'un système. Ce module fournit également l'interprétation microscopique qui permet d'instancier ces représentations pour obtenir des agents autonomes,

  - le module *Interaction* regroupe les fonctionnalités permettant de décrire des schémas d'interaction, de représenter des connaissances sociales, ainsi que les schémas de conversation. Il fournit également les mécanismes nécessaires à la réalisation de ceux-ci (appel de fonction, procédure, d'envoi de message, etc) qui permettent de les relier aux mécanismes de liaison définis pour chaque modèle d'agent.

  - le module *ProtocolLibrary* regroupe les fonctionnalités de description des protocoles d'interaction, et de liaison de cette description avec les mécanismes nécessaires à la réalisation des appels d'actions qu'elle décrit. Ce module comprend également une bibliothèque extensible de protocoles d'interaction,

  - le module *Preference* regroupe les fonctionnalités de description, de représentation et de manipulation des préférences intervenant dans la déclaration des interactions au niveau de la déclaration d'un système et dans la gestion des connaissances sociales au niveau du fonctionnement des agents,

- des outils utilisés pour le fonctionnement des agents autonomes créés :

  - le module *MailBox* regroupe toutes les fonctionnalités de base des communications au moyens de boîtes aux lettres, ce qui comprend au niveau d'un agent la représentation de sa boîte au lettre et des opérations lui permettant de la gérer, ainsi que les mécanismes permettant le dépôt de messages dans les boîtes aux lettres des autres agents,

- le module *Message* regroupe toutes les fonctionnalités relatives à la structuration des messages que ce soit pour leur construction ou leur consultation,

- le module *PreferencesManager* regroupe les fonctionnalités permettant à chaque agent de gérer ses connaissances sociales, de représenter les interactions initiées et les mécanisme d'initiation et de suivi de ces interactions manipulant ces représentations pour une gestion automatique de celles-ci,

- le module *PhysicalAgentModel* a pour but de rassembler différents modèles physiques d'exécution (ou noyaux) des agents autonomes qui pourront être mis en place pour rendre opérationnelles les descriptions de systèmes. Il ne contient actuellement qu'un modèle basé sur les processus légers (*Thread* de JAVA), mais d'autres modèles peuvent être définis pour le remplacer.

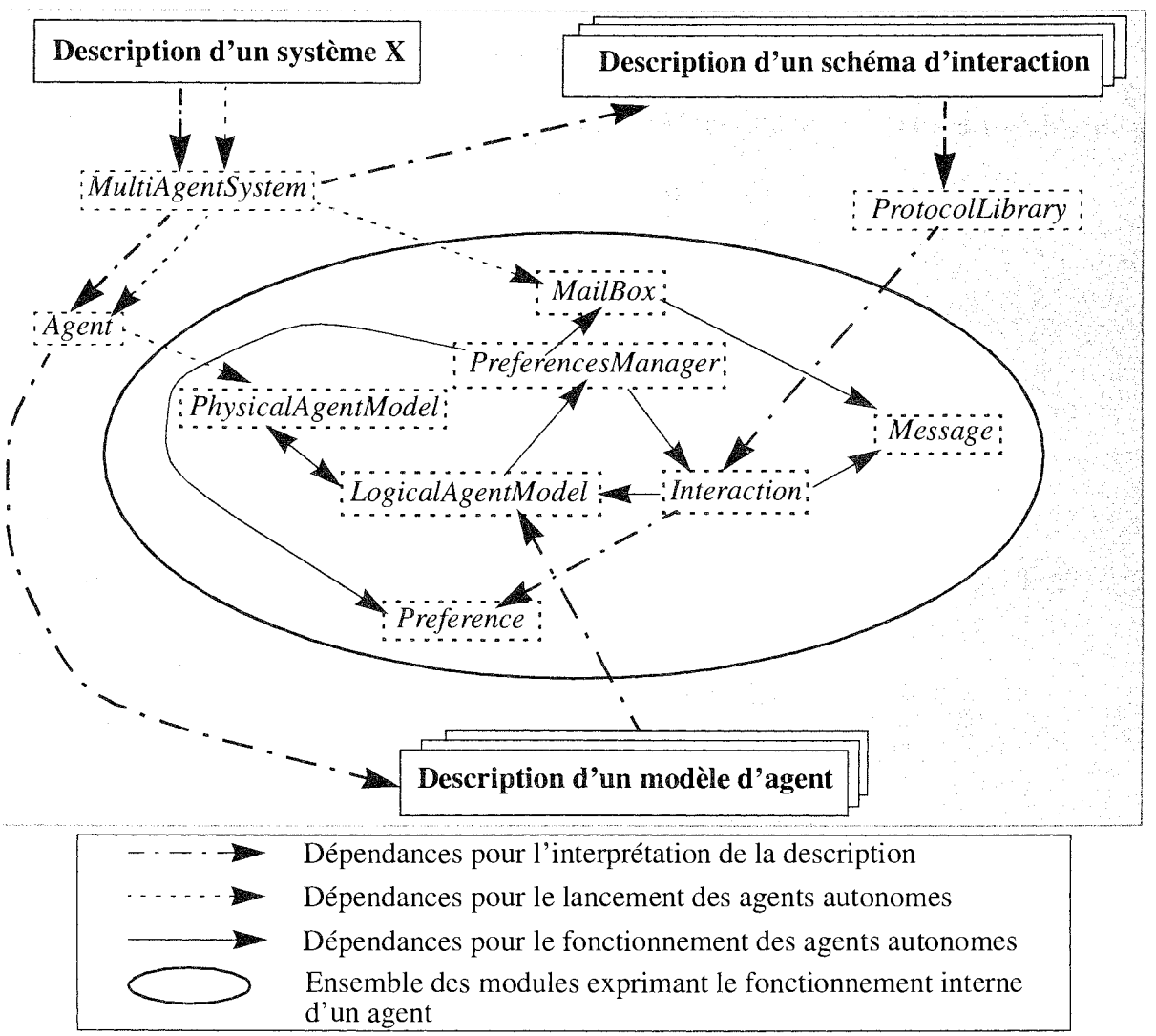


FIGURE 16-2 : Architecture et interdépendance des modules

Comme le montre la figure 16-2, ces différents modules sont interdépendants et sont tous nécessaires pour la mise en place d'un système à partir de l'architecture de notre plate-forme. Les dépendances sont présentées dans cette figure pour l'interprétation d'une description, pour le lancement des agents et le fonctionnement des agents du système décrit.

Nous avons aussi deux modules complémentaires qui ne sont pas forcément requis pour toutes descriptions et exécutions de systèmes :

- le module *FileUtility* qui a pour but de fournir les outils de base pour décrire les entrées/sorties sur des fichiers. Il définit également un outil de deboggage post-mortem qui consiste à créer des fichiers relatifs au lancement et au fonctionnement chaque agent, ainsi qu'un fichier relatif à l'interprétation, au lancement et au fonctionnement du système dans sa globalité,

- le module *Cycle* a pour but de fournir les outils permettant d'exprimer une activité cyclique au niveau de la description du comportement des agents et d'opérationnaliser cette activité cyclique. La figure 16-3 montre une modélisation de la description de cette activité cyclque à l'aide de réseaux de Petri. Le comportement de chaque agent est modélisé sous la forme d'une boucle sans fin où trois places représentent le début, le milieu et la fin d'un cycle. Les transitions entre ces places correspondent à des barrières de synchronisation de l'activité cyclique des agents. Ce principe s'utilise au niveau de chaque agent pour exprimer à un niveau local la synchronisation pour passer au cycle suivant et au niveau du système pour opérationnaliser l'activité du système sur un nombre déterminé de cycles (représenté par le nombre de jeton à la place intitulée cycles).

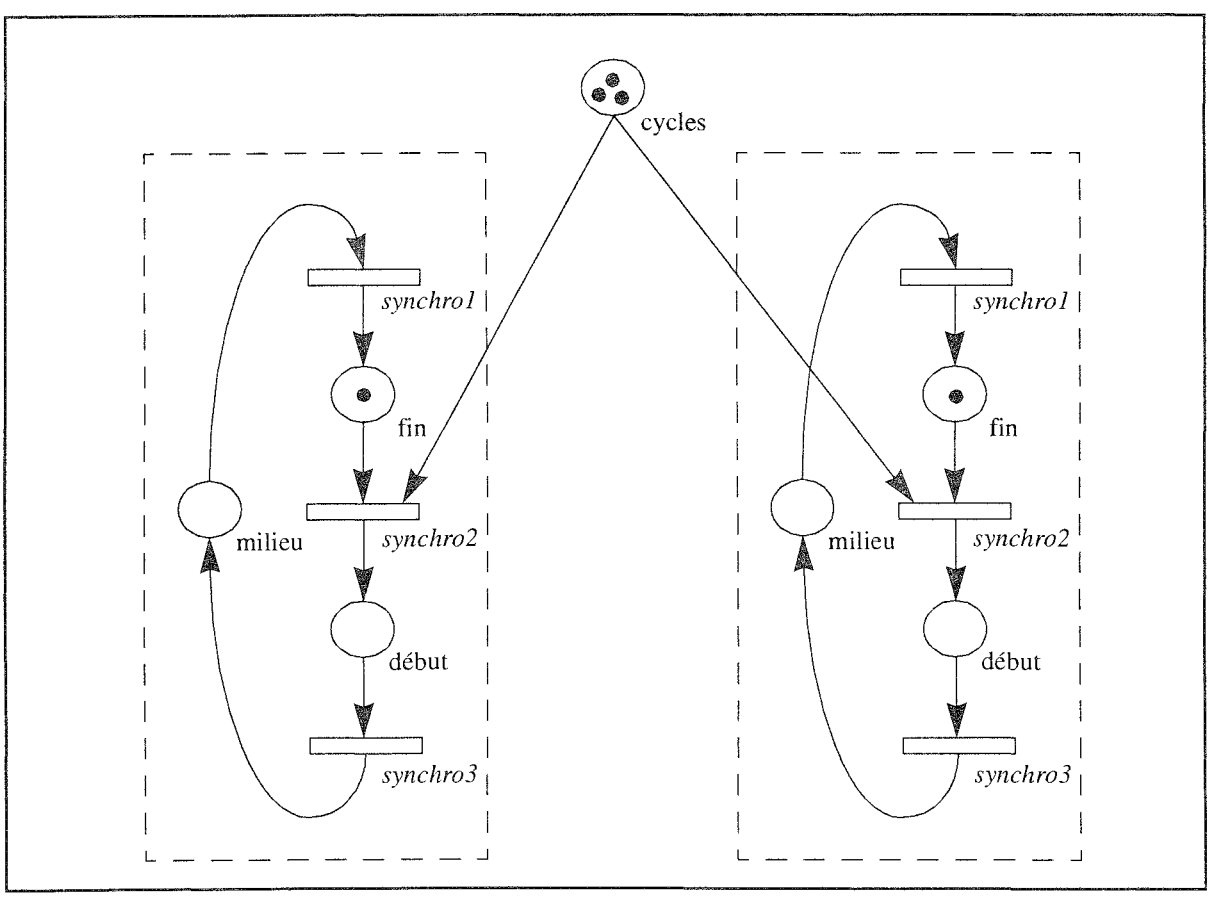


FIGURE 16-3 : Modélisation de l'activité cyclique des agents.

4 CONCLUSION

La plate-forme que nous venons de présenter fournit un cadre descriptif pour la construction de système. Ce cadre se fonde sur une description de système basée sur des modèles d'agent et d'interaction. Cette plate-forme permet ainsi de construire un système selon l'équation SMA = A ⊕ I que nous avons évoqué dans le chapitre 13.

La taille du code réalisant l'implantation de cette plate-forme est néanmoins raisonnable comme le montre le tableau suivant :

TABLEAU 16-4 : Taille du code JAVA

Modules	Nombre de lignes	Pourcentage
MultiAgentSystem	279	5,2%
LogicalAgentModel	621	11,5%
Agent	189	3,5%
Interaction	2067	38,4%
ProtocolLibrary[a]	75	1,4%
Preference	69	1,3%
MailBox	161	3%
Message	79	1,5%
PreferenceManager	841	15,6%
PhysicalAgentModel	195	3,6%
FileUtility	243	4,5%
Cycle	564	10,5%
TOTAL	**5383**	**100%**

a. Nous ne comptabilisons pas ici les protocoles décrits dans la bibliothèque, mais uniquement le nombre de lignes de la classe héritée par toutes les descriptions de protocoles.

Cette plate-forme se base sur l'approche descriptive et opérationnelle que nous avons présentée dans la partie précédente. Une extension envisagée pour cette plate-forme est d'ajouter un environnement graphique de conception permettant de construire aisément le squelette d'un système (description des agents et des structures d'interactions entre ceux-ci, de même que pour la définition des protocoles d'interaction), ainsi qu'une interface graphique en temps réel permettant de visualiser l'évolution des interactions au sein du système.

La plate-forme ne se limite pas pour autant aux aspects statiques de l'organisation de système. Elle permet aussi d'exprimer des organisations dynamiques selon les principes que nous avons montrés dans la partie précédente pour la mise en place de processus de réorganisation des interactions. Nous allons illustrer l'utilisation de cette plate-forme dans le chapitre suivant.

Chapitre 17 Construction d'applications Multi-Agents

Ce chapitre présente l'élaboration d'applications multi-agents à partir de notre plate-forme JAVAMA en terme de description d'un système (agents, interactions et société), et d'organisation des interactions du système (statique ou dynamique).

Nous illustrons les possibilités de construction de systèmes applicatifs à partir de JAVAMA en reprenant l'exemple de flux multiples de marchandises que nous avons présenté dans le premier chapitre de cette partie.

Après une brève précision des simplifications apportées à l'énoncé de ce problème, nous présentons deux exemples applicatifs sur celui-ci. Nous détaillons deux étapes préalables à la construction de ces systèmes applicatifs. Ces deux étapes consistent à identifier les modèles d'agent et les schémas d'interaction qui prendront place dans les systèmes construits. Enfin, nous présentons la description de nos deux exemples d'applications, avant de terminer par un bilan des possibilités et les limites de notre plate-forme en terme de description.

1 UN PROBLÈME SIMPLIFIÉ

Ce chapitre a pour but de présenter comment décrire une application à partir des différents niveaux (local, semi-global et global) de description incorporés dans JAVAMA (agent, interaction et société organisée). Cette construction d'application reprend le problème développé avec GTMAS (cf chapitre 15) auquel nous avons apporté quelques simplifications :

- une restriction de la forme d'expression des fonctions de production en imposant un seul

paramètre (un seul bien) en entrée et en sortie (ceci afin de diminuer les types de biens échangés et donc la complexité de la description de la société traitée) ;

- la suppression des interactions visant à offrir les biens des stocks sortants (qui n'auraient apportées qu'un deuxième exemple illustratif de la description de protocole d'interaction dans ce chapitre).

1.1 Deux exemples d'applications

Nous présentons ici les deux exemples applicatifs qui sont utilisés pour présenter la construction de systèmes multi-agents dans ce chapitre. La première application consiste à décrire une société simple *(cf figure 17-1)*, afin de permettre une analyse préliminaire de la complexité de la deuxième application qui comprend des interactions et des agents plus complexes.

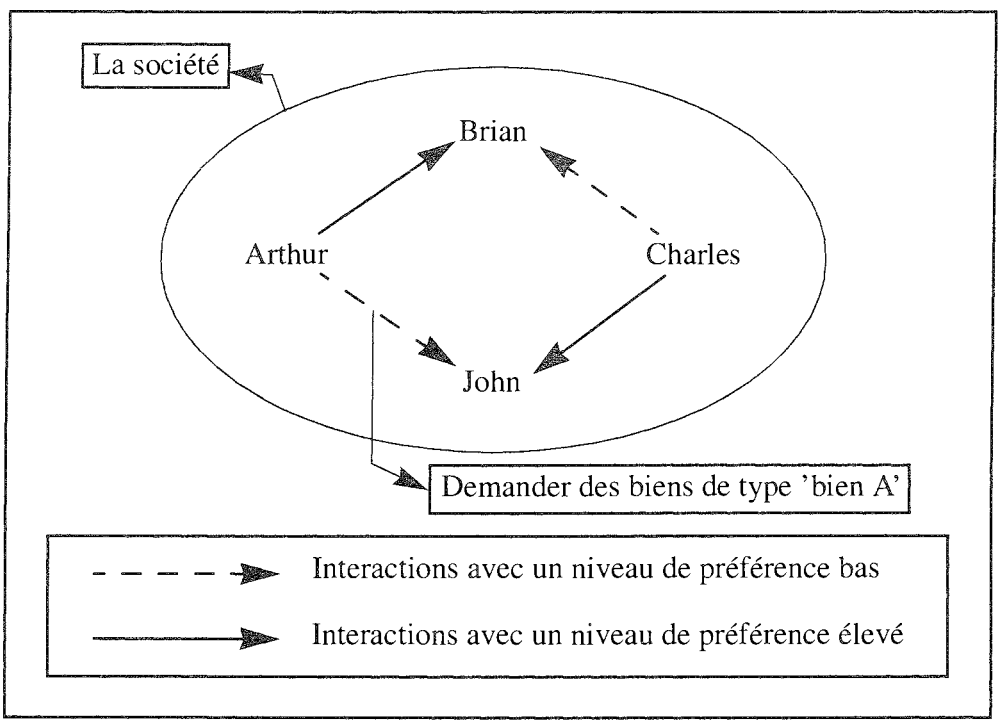


FIGURE 17-1 : Une société d'agents simple

Cette application correspond à une société composée de deux types d'agents (modèles d'agent) : des agents producteurs et des agents consommateurs. Les premiers produisent de manière cyclique une certaine quantité d'un bien identifié (le bien A) et fournissent les demandes des agents consommateurs qui ont besoin de ce type de bien en entrée. La figure 17-1 présente l'organisation de départ pour les interactions de cette société.

La seconde application est plus complexe. La figure 17-2 présente l'organisation des interactions de la société avec des niveaux hiérarchiques : initial, intermédiaire et final. Ces niveaux correspondent à des types d'agents différents. Les niveaux initial et final correspondent respectivement aux modèles des agents producteurs et consommateurs de l'application précédente. Le niveau intermédiaire correspond à un modèle d'agent combinant les deux modèles précédents. Cette application diffère de la première par le nombre d'agents composant la société et par le type d'interactions utilisées. En effet, ces interactions s'enchaînent de manière récursive : si

l'agent intermédiaire Agent2_1 ne dispose pas de la quantité de bien B demandé par l'agent Agent3_1 (10 biens de type B), il enchaînera alors sur une interaction avec l'agent Agent1_1 pour se fournir les biens (de type A) nécessaires pour produire les biens qui lui ont été demandés (la capacité de production de l'agent Agent2_1 étant de 3 biens de type B par cycle et comme 10B < 4*3B, il demandera alors à l'agent Agent1_1 au maximum 4*3A = 12A et ne pourra satisfaire la demande de l'agent Agent3_1 avant d'avoir produit les biens B manquants).

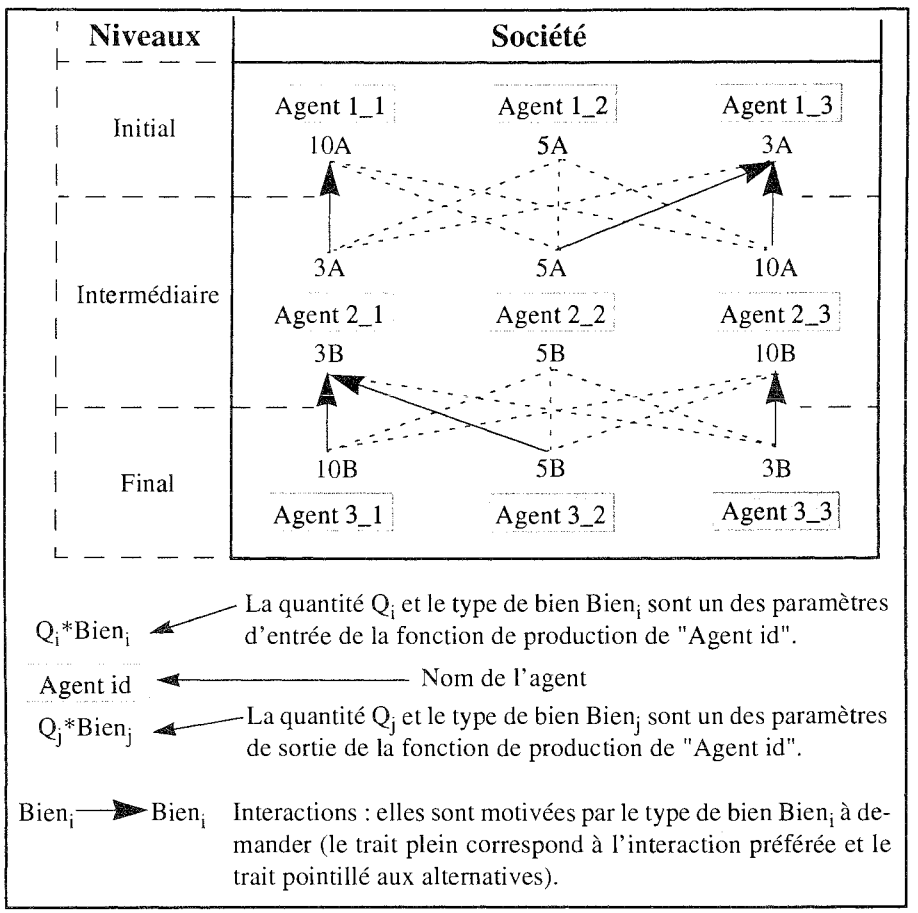


FIGURE 17-2 : Une société d'agents complexe.

2 ANALYSE PRÉALABLE

Cette phase d'analyse du problème consiste à identifier et à concevoir les schémas d'interaction et les modèles d'agent nécessaires à la description d'une application à partir de notre plate-forme. Cette analyse se base avant tout sur la conception des interactions qui interviendront entre les agents et qui utiliseront des actions spécifiques aux agents.

Nous allons présenter la conception des schémas d'interaction pour nos deux applications, puis nous présentons les modèles des agents de la seconde application qui seront reliés par l'instanciation des schémas : tout ceci au travers d'une présentation sous forme de grammaire des descriptions (le lecteur pourra se reporter à l'annexe A pour un extrait du pseudo-code JAVA

équivalent à la description de la deuxième application).

2.1 Les schémas d'interaction identifiés

Afin de simplifier la présentation de ces schémas, nous commençons par détailler le schéma d'interaction de notre première application, avant d'indiquer les modifications à effectuer dans celui-ci pour obtenir les schémas utilisés pour la seconde.

La première application se base sur des interactions entre des agents consommateurs et des agents producteurs. Nous désignons ces agents en fonction de leur rôle vis à vis de l'interaction décrite : les consommateurs seront les demandeurs (notés D) de biens et les producteurs seront les fournisseurs (notés F) de ces biens. En se basant sur notre grammaire, nous pouvons alors déclarer le schéma d'interaction sous la forme suivante :

<PA> = "Demander A" {D, F} <protocole-de-demande>

Où *<PA>* désigne le schéma, et ou *<protocole-de-demande>* désigne le protocole d'interaction.

Le protocole d'interaction désigné se présente sous la forme de différents noeuds et transitions où des agents abstraits (D ou F) doivent effectuer des actions spécifiques :

<protocole-de-demande> = <noeud initial> <noeuds intermédiaires> <noeud final> <transitions>
<noeud initial> = ***initial*** *<0>*
<noeud final> = ***final*** *<6>*
<noeuds intermédiaires> = { ***inter*** *<2>*
inter *<3>*
inter *<4>*
inter *<5> }*[1]

Où les suites d'actions à effectuer sont :

<0> = <Initialisation>
<2> = <Engagement-CalculCycle>
<3> = <Memoriser-JUGER>
<4> = Λ
<5> = <Livraison>
<6> = <Incorporer>

Les transitions entre ces noeuds sont principalement des envois et des réceptions de messages. Mais étant donné que les agents doivent avoir une activité cyclique nous utiliserons aussi des prédicats ayant pour but d'assurer la synchronisation des interactions avec le cycle où elles ont débuté (le cycle de l'initiation de l'interaction) et le cycle ou elles doivent se terminer (le cycle où la livraison s'effectue). Les transitions s'expriment selon notre grammaire de la manière suivante :

Pour un agent demandeur :

initial *<0> <senddemande>* ***inter*** *<2>*
inter *<2> <recvannonce>* ***inter*** *<3>*
inter *<3> <Barriere-D>* ***inter*** *<4>*
inter *<4> <recvlivraison>* ***final*** *<6>*

Pour un agent fournisseur :

1. Notation adopté pour les ensembles de noeuds

initial *<0> <recvdemande>* ***inter*** *<2>*
inter *<2> <sendannonce>* ***inter*** *<3>*
inter *<3> <Barriere-F>* ***inter*** *<4>*
inter *<4> <AttenteCycle>* ***inter*** *<5>*
inter *<5> <sendlivraison>* ***final*** *<6>*

Les transitions de type envoi ou réception sont les suivantes :

<senddemande> = D : ***send to*** *F : Demande $Nb $Prod*
<recvdemande> = F : ***receive from*** *D : Demande $Nb $Prod*
<sendannonce> = F : ***send to*** *D : Annoncé pour $Cycle*
<recvannonce> = D : ***receive from*** *F : Annoncé pour $Cycle*
<sendlivraison> = F : ***send to*** *D : Livraison $Nb $Prod*
<recvlivraison> = D : ***receive from*** *F : Livraison $Nb $Prod*

Les variables échangées au travers des messages correspondent aux variables précisant le nombre de biens demandés, le type de ce bien et le cycle annoncé pour la livraison de la demande. Selon notre grammaire, la déclaration de ces variables s'exprime ainsi :

<Nb> = $Nb entier
<Prod> = $Prod chaîne de caractères
<Cycle> = $Cycle entier

Afin de permettre la synchronisation des interactions et de l'activité cyclique des agents, nous utilisons un mécanisme de barrière de synchronisation. Ce mécanisme consiste, au niveau des interactions, à activer une barrière lors de l'initiation d'une interaction. L'agent ayant fait cette initiation devra alors attendre que cette barrière soit franchie pour terminer son cycle d'activité. Le franchissement de cette barrière intervient au niveau du protocole juste avant le prédicat d'attente du cycle annoncé pour la livraison pour un agent fournisseur et avant la réception de la livraison pour un agent demandeur. Ceci se traduit au niveau du protocole d'interaction par les prédicats suivants :

<Barriere-F> = F : ***name*** *Barriere () → <Barriere-franchie>*
<Barriere-D> = D : ***name*** *Barriere () → <Barriere-franchie>*
<Barriere-franchie> = $Barriere-franchie booléen
<AttenteCycle> = F : ***name*** *AttenteCycle (<Cycle>) → <Finattente?>*
<Finattente?> = $Finattente? booléen

La déclaration d'une barrière s'effectue lors du noeud initial grâce aux actions :

<barriere-Fournisseur> = F : ***name*** *DeclareBarriere ()*
<barriere-Demandeur> = D : ***name*** *DeclareBarriere ()*

Les actions effectuées lors de ce noeud initial sont :

<Initialisation> = { <barriere-Fournisseur>,
<barriere-Demandeur>,
<produit-demande>,
<quantite-demandee> }[1]

Les actions *<quantite-demandee>* et *<produit-demande>* permettent de paramètrer la réalisation d'une interaction pour un agent demandeur. Elles fournissent la quantité de biens à demander, ainsi que la nature de ce bien :

<quantite-demandee> = D : ***name*** *QuantiteDemandee () → <Nb>*
<produit-demande> = D : ***name*** *Produit () → <Prod>*

1. Notation adoptée pour les suites d'actions

Le premier noeud intermédiaire (désigné par ***inter*** <2>) consiste à entreprendre les actions permettant à un agent fournisseur d'évaluer le nombre de cycles nécessaires pour fournir la quantité de biens demandée. Pour ces agents, cela consiste à réaliser deux actions : l'une pour évaluer le cycle à annoncer pour la livraison et l'autre pour mémoriser localement l'engagement de livrer à ce cycle.

<Engagement-CalculCycle> = { <calculcycle>,
<engagement>}
*<calculcycle> = F : **name** CalculCycle (<Nb>, <Prod>) → <Cycle>*
*<engagement> = F : **name** Engagement (<Nb>, <Prod>, <Cycle>)*

Les actions du noeud suivant correspondent aux actions entreprises par l'agent demandeur lorsqu'il a reçu l'annonce d'une livraison pour sa demande. Ces actions consistent alors à mémoriser cette promesse de livraison, et à juger le délai de livraison annoncé pour cette interaction (cette dernière action correspond au processus de réorganisation des interactions).

<Memoriser-JUGER> = { <memorisercycle>,
<JUGEMENT> }
*<memorisercycle> = D : **name** MemoriserCycle (<Cycle>, <Nb>, <Prod>)*
*<JUGEMENT> = D : **name** JugerDemande (<Cycle>, <Prod>)*

Après avoir franchi la barrière de synchronisation et l'attente du cycle pour lequel ils se sont engagés à livrer, les agents fournisseurs peuvent alors entreprendre les actions leur permettant de préparer cette livraison (mise à jour du stock sortant) et l'annulation de leur engagement.

<Livraison> = { <preparerlivraison>,
<desengagement>}
*<preparerlivraison> = F : **name** PreparerLivraison (<Nb>, <Prod>)*
*<desengagement> = F : **name** Desengagement ()*

Le dernier noeud correspond à la réception et à l'incorporation des biens livrés dans le stock entrant de l'agent demandeur. Les actions entreprises sont alors l'incorporation de ces biens et l'annulation de la mémorisation de la promesse à laquelle correspond cette livraison, puisque celle-ci a été tenue.

<Incorporer> = { <effacermemorisation>,
<incorporer>}
*<effacermemorisation> = D : **name** EffacerMemorisation ()*
*<incorporer> = D : **name** Incorporer (<Nb>, <Prod>)*

Pour la seconde application, les schémas d'interaction sont similaires à celui-ci. Nous en présentons uniquement les modifications qui permettent de prendre en compte une particularité de ces schémas : la récursivité des interactions. Dans cette application, nous utilisons deux schémas différents pour les deux types de biens échangés entre les agents :

<PA> = "Demander A" {D, F} <protocole-de-demande2>
<PB> = "Demander B" {D, F} <protocole-de-demande2>

Les modifications apparaissent dans la description du protocole d'interaction. Elles se traduisent par un nouveau noeud intermédiaire après la réception d'une demande où l'agent fournisseur initiera les interactions avec ses propres fournisseurs pour obtenir les biens entrant manquants et nécessaires pour produire les biens sortant qui lui demandés. La grammaire de ce protocole correspond alors à :

<protocole-de-demande2> = <noeud initial> <noeuds intermédiaires> <noeud final> <transitions>
*<noeud initial> = **initial** <0>*
*<noeud final> = **final** <6>*

<noeuds intermédiaires> = { ***inter*** *<1>*
inter *<2>*
inter *<3>*
inter *<4>*
inter *<5> }*

Où les suites d'actions sont :

<0> = <Initialisation>
<1> = <Recursivite>
<2> = <Engagement-CalculCycle>
<3> = <Memoriser-JUGER>
<4> = Λ
<5> = <Livraison>
<6> = <Incorporer>

Les transitions sont les suivantes :

Pour un agent demandeur :

initial *<0> <senddemande>* ***inter*** *<2>*
inter *<2> <recvannonce>* ***inter*** *<3>*
inter *<3> <Barriere-D>* ***inter*** *<4>*
inter *<4> <recvlivraison>* ***final*** *<6>*

Pour un agent fournisseur :

initial *<0> <recvdemande>* ***inter*** *<1>*
inter *<1> <AttenteFinRecusion>* ***inter*** *<2>*
inter *<2> <sendannonce>* ***inter*** *<3>*
inter *<3> <Barriere-F>* ***inter*** *<4>*
inter *<4> <AttenteCycle>* ***inter*** *<5>*
inter *<5> <sendlivraison>* ***final*** *<6>*

Nous avons donc ici une nouvelle suite d'actions *<Recursion>* et une nouvelle transition *<AttenteFinRecusion>*. Cette nouvelle suite d'actions consiste à effectuer des actions spécifiques suivant le niveau *(cf figure 17-2)* auquel appartient l'agent fournisseur. S'il fait partie du niveau intermédiaire, en fonction de ses stocks entrant et sortant disponibles, il pourra être amené à initier un certain nombre d'interactions avec les agents du niveau initial. Il devra alors mémoriser un indice supplémentaire relatif à son engagement à livrer : le nombre d'interactions qu'il a entreprises pour pouvoir satisfaire la demande. Par contre, si l'agent fait partie du niveau initial, il n'aura aucune demande récursive à effectuer. Au niveau de la déclaration du protocole, ceci correspond à :

<Recursion> = { *<recursion>,*
<engagementrecursif>};
<recursion> = F : ***name*** *DemandeRecursive (<Nb>, <Prod>) → <NbI>*
<NbI> = $NbI entier
<engagementrecursif> = F : ***name*** *Engagement (<Nb>, <Prod>, <NbI>)*

La nouvelle transition consiste alors à attendre que les interactions engagées par récursion annoncent un cycle de livraison qui va influer sur le calcul du cycle à annoncer pour l'agent fournisseur.

<AttenteFinRecursion> = F: ***name*** *AttenteFinRecursion (<NbI>) → <Finrecursion?>*
<Finrecursion?> = $Finrecursion? booléen

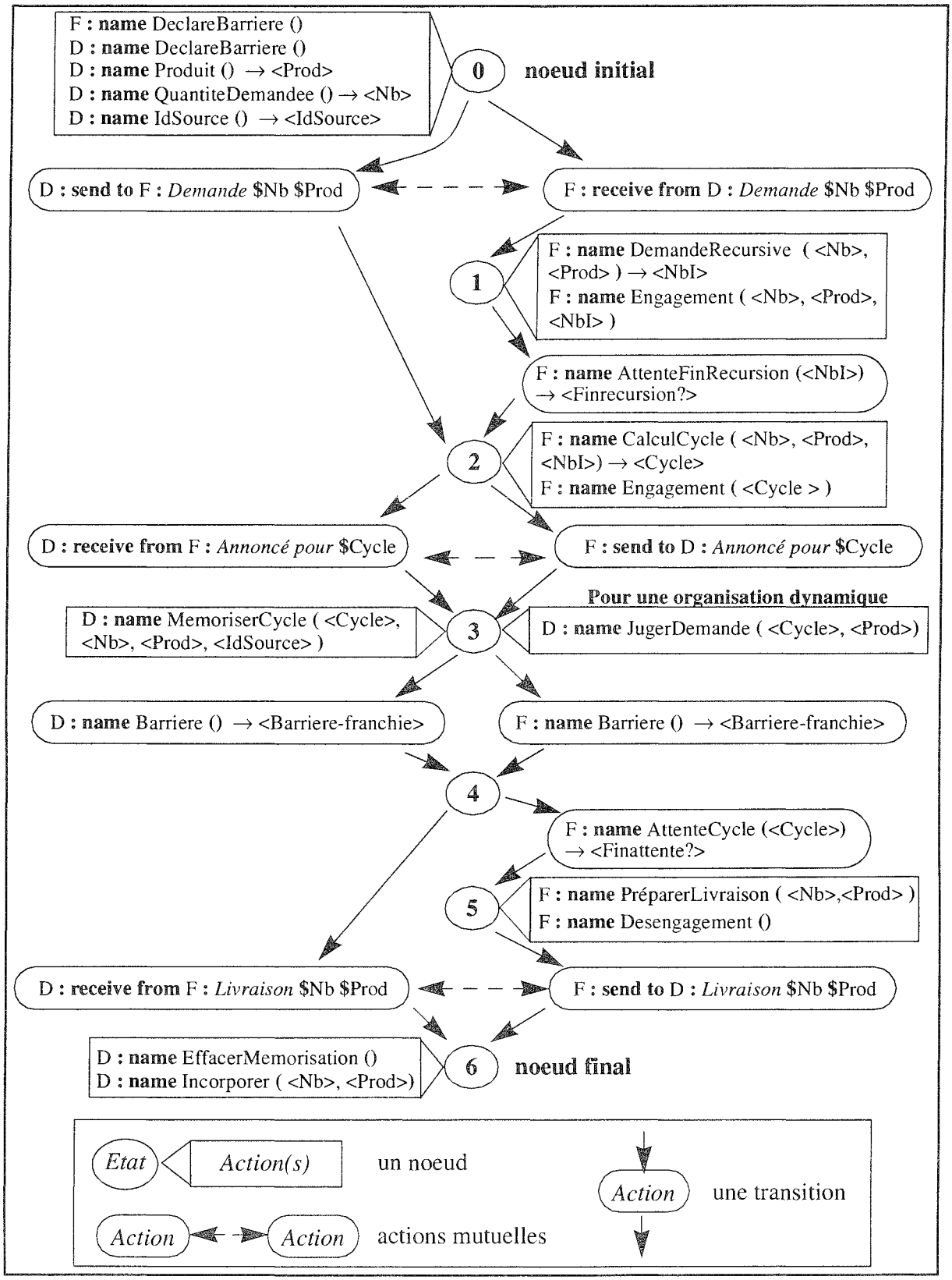


FIGURE 17-3 : Représentation graphique du protocole d'interaction.

La liaison entre l'interaction source de la récursion et les autres est assurée par les deux struc-

tures locales aux agents mémorisant les engagements et les promesses. Pour chaque engagement l'agent connaît le nombre d'interactions qu'il a entreprises par récursion et pour chaque promesse reçue, l'agent connaît l'identificateur de la conversation qui fut source de cette interaction. Il suffit alors à un agent fournisseur d'attendre qu'il ait reçu *NbI* promesses où l'identificateur de la conversation source correspond.

Nous ajoutons alors une fonction dans la suite d'actions à effectuer lors du noeud initial, pour connaître l'identificateur de la conversation source :

<Initialisation> = { ...,<idconvsource>};
*<idconvsource> = D : **name** IdSource () → <IdSource>*
<IdSource> = $IdSource chaîne de caractères

Si l'interaction n'est pas issue d'une récursion, comme c'est le cas pour les agents du niveau final *(cf figure 17-2)*, cette fonction retourne une chaîne de caractère vide.

Les actions d'engagement, de mémorisation des promesses et de calcul du cycle à annoncer présentées lors de l'étude du protocole pour la première application sont modifiées pour prendre en compte les nouveaux indices à mémoriser :

*<calculcycle> = F : **name** CalculCycle (<Nb>, <Prod>, <NbI>) → <Cycle>*
*<engagement> = F : **name** Engagement (<Cycle>)*
*<memorisercycle> = D : **name** MemoriserCycle (<Cycle>, <Nb>, <Prod>, <IdSource>)*

Afin de permettre aux lecteurs de disposer d'une vue d'ensemble de la déclaration de ce protocole d'interaction, nous en présentons une représentation graphique dans la figure 17-3.

2.2 Les modèles d'agents identifiés

Nous avons distingué trois types d'agents intervenant dans la deuxième application : les agents du niveau initial, du niveau final et des niveaux intermédiaires (voir figure 17-2). Ces types d'agents correspondent respectivement à des modèles explicitant le comportement des agents. Nous présentons le pseudo-algorithme de ces modèles qui constitue plus de 80% du code de la deuxième application (dont l'implantation est présentée en annexe A.II).

Nous adoptons pour ces différents modèles une partie commune dans leur description, qui correspond à :

```
<modèle d'agent> (    %input chaîne de caractères,
                      %output chaîne de caractères,
                      %necessary entier ,
                      %capacity entier )
{
      /* stocks des biens lors de l'initialisation */
      int input[%input] = 0;
      int output[%output] = 0;
      /* paramètres de la fonction de production */
      q[%input] = %necessary;
      q[%output] = %capacity;
...
}
```

Où *%input* et *%output* sont les paramètres du modèle désignant respectivement le type de bien entrant et sortant des agents, et où *%necessary* et *%capacity* sont les entiers désignant respectivement les biens entrants nécessaires et les biens sortants produits par la fonction de production cyclique des agents.

2.2.1 Le modèle pour les agents du niveau initial

Il s'agit du modèle correspondant aux agents alimentant la société en biens consommables. Ces agents n'utilisent que les schémas d'interaction motivés par les biens de type A et uniquement selon le point de vue du rôle de fournisseur.

Ce modèle d'agent correspond à :

AgentInitial (%input chaîne de caractères,
%output chaîne de caractères,
%necessary entier ,
%capacity entier)
{

Connaissances :

/ Partie commune aux différents modèles */*
...
/ Mémorisation locale des engagements pris par l'agent */*
commitments = {}

Les engagements sont mémorisés au moyen d'une liste pour chaque agent. Un engagement correspond à une structure regroupant l'identificateur de la conversation, le type du bien, la quantité manquante et disponible de ce bien, ainsi que le cycle annoncé pour la livraison. La figure 17-4 présente cette structure pour la première interaction initiée par l'agent Agent2_1 avec l'agent Agent1_1 comme fournisseur pour le bien de type A.

			Quantité		
	Identificateur	**Type**	**Manquante**	**Disponible**	**Date**
Un engagement	*'Agent2_1:1'*	*'bien A'*	*3*	*0*	*2*

FIGURE 17-4 : La structure d'un engagement

Nous supposons ici l'existence de certaines fonctions sur cette liste d'engagements :

add(identificateur, type, manquante, disponible) : ajoutant un engagement dans la liste,
add(identificateur, date) : précisant le date de livraison annoncé pour cet engagement,
del(identificateur) : supprimant un engagement de la liste,
Qmanquante(identificateur) : retournant la quantité de bien manquante pour un engagement,
SumQmanquante(prod) : retournant le cumul des quantités manquantes pour le bien prod.

Nous utilisons aussi certaines fonctions de gestion des interactions initiées fournies par la plate-forme :

Interaction() : le mécanisme d'initiation et de poursuite des interactions,
Start_Interaction (<motivation>) : le mécanisme d'initiation d'une interaction à partir d'une motivation,
IdentificateurCourant() : retournant l'identification de conversation de l'interaction initiées traitées.

Compétences :

```
void GenerateStockInput () {
        input[%input] = input[%input] + q[%input];
}
boolean Possible () {
        return (input[%input] >= q[%input]);
}
```

```
void Production () {
        input[%input] = input[%input] - q[%input];
        output[%output] = output[%output] + q[%output];
}
int DemandeRecursive (int nb, string prod) {
        return 0;
}
void Engagement (int nb, string prod, int nbi) {
        string id = IdentificateurCourant();
        int manquant;
        int dispo = output[prod] - commitments.SumQManquante(prod);
        si ( dispo >= nb)
                manquant = 0;
                dispo = nb;
        sinon
                si ( (dispo > 0) and (dispo < nb))
                        manquant = nb - dispo;
                sinon
                        manquant = nb;
                        dispo = 0;
                fsi
        fsi
        output[prod] = output[prod] - dispo;
        commitments.add(id, prod, manque, dispo);
}
boolean AttenteFinRecursion (int nbi) {
        return true;
}
int CalculCycle (int nb, string prod, int nbi) {
        int manque = commitments.SumQManquante(prod);
        return ( (manque div q[prod]) + Cycle-courant + 1);
}
void Engagement (int cycle) {
        string id = IdentificateurCourant();
        commitments.add(id, cycle);
}
void DeclareBarriere () {
        /* dépôt d'une barrière identifiée par l'identificateur de la conversation */
}
boolean Barriere () {
        /* retrait de la barrière correspondant à l'identificateur de la conversation*/
        return ( vrai si la liste des barrières est vide );
}
boolean AttenteCycle (int cycle) {
        return ( Cycle-courant == cycle );
}
void PreparerLivraison (int nb, string prod) {
        string id = IdentificateurCourant();
        int manque = commintments.Qmanquante(id);
        output[prod] = output[prod] - manque;
}
void Desengagement () {
        string id = IdentificateurCourant();
        commintments.del(id);
}
```

Comportement :

```
void run () {
        Pour chaque Cycle-courant de 1 à Cycle-fin
            GenerateStockInput();
            si (Possible())
                Production();
            fsi
            Tant que des messages sont dans la boîte aux lettres
                ou des interactions initiées ont encore une barrière à franchir
                ou des engagements sont prévus pour ce cycle
                faire
                Interaction ();
            ftant
        fpour
}
```

2.2.2 Le modèle pour les agents du niveau final

Ce modèle d'agent correspond aux agents ne réalisant qu'une consommation de biens : la fonction de production n'ayant alors pour seul but que de consommer le stock entrant à chaque cycle.

Ce modèle d'agent correspond à :

```
AgentFinal (    %input chaîne de caractères,
                %output chaîne de caractères,
                %necessary entier ,
                %capacity entier )
{
```

Connaissances :

```
    /* Partie commune aux différents modèles */
    ...
    /* Mémorisation locale des promesses : une liste de promesses */
        promesses = {};
```

Les promesses sont mémorisées, comme les engagements, au moyen d'une liste pour chaque agent. Une promesse correspond à une structure regroupant l'identificateur de la conversation (et celui de l'interaction qui fut source de récursion pour le modèle d'agent intermédiaire), le type du bien, la quantité, ainsi que le cycle annoncé pour la livraison. La figure 17-5 présente cette structure pour la première interaction initiée par l'agent Agent3_1 avec l'agent Agent2_1 comme fournisseur pour le bien de type A.

	Idcourant	Idsource	Quantité	Type	Date
Une promesse	*'Agent3_1:1'*		*10*	*'bien B'*	*5*

FIGURE 17-5 : La structure d'une promesse

Nous supposons ici l'existence de certaines fonctions sur cette liste de promesses :

add(idcourant, idsource, quantité, type, date) : ajoutant une promesse dans la liste,
del(idcourant) : enlevant une promesse de la liste.

Compétences :

```
boolean Possible () {
        return (input[%input] >= q[%input]);
}
void Production () {
        input[%input] = input[%input] - q[%input];
        output[%output] = output[%output] + q[%output];
}
string Produit() {
        return (%input);
}
int QuantiteDemandee() {
        return (q[%input]);
}
string IdSource() {
        return(");
}
void MemoriserCycle (int cycle, int nb, string prod, string ids) {
        string id = IdentificateurCourant();
        promesse.add(id, ids, nb, prod, cycle);
}
void EffacerMemorisation () {
        string id = IdentificateurCourant();
        promesse.del(id);
}
void DeclareBarriere () {
        /* dépôt d'une barrière identifiée par l'identificateur de la conversation */
}
boolean Barriere () {
        /* retrait de la barrière correspondante à l'identificateur de la conversation*/
        return ( vrai si la liste des barrières est vide );
}
void Incorporer (int nb, string prod) {
        input[prod] = input[prod]+ nb;
}
```

Comportement :

```
void run () {
        Pour chaque Cycle-courant de 1 à Cycle-fin
            si (non Possible())
                Start-Interaction("Demander %input");
            fsi
            Tant que des messages sont dans la boîte aux lettres
                ou des interactions initiées ont encore une barrière à franchir
                ou des livraisons ont été promises pour ce cycle
                faire
                Interaction ();
            ftant
            si (Possible())
                Production();
            fsi
        fpour
}
```

2.2.3 Le modèle pour les agents du niveau intermédiaire

Ce modèle d'agent est une composition des deux autres. Nous allons rapidement en donner le comportement et les modifications apportées qui correspondent à :

```
AgentInter (    %input chaîne de caractères,
                %output chaîne de caractères,
                %necessary entier ,
                %capacity entier )
(
```

Connaissances :

```
    /* Partie commune aux différents modèles */
    ...
    /* Mémorisation locale des promesses : une liste de promesses */
        promesses = {};
    /* Mémorisation locale des engagements pris par l'agent */
        commitments = {};
    /* Paramètres des demandes en cas de récursion */
        string idsource;      /* identificateur de l'interaction source */
        int coef;             /* coefficient multiplicateur des biens entrant nécessaires */
```

Compétences :

```
...
int QuantiteDemandee() {
        return (q[%input]*coef);
}
string IdSource() {
        return(idsource);
}
int DemandeRecursive (int nb, string prod) {
        string id = IdentificateurCourant();
        int manque = commintments.Qmanquante(id);
        si manque > 0
            idsource = id;
            coef= (manque div q[prod]) + (manque mod q[prod]);
            Start-Interaction("Demander %input");
            return 1;
        sinon
            return 0;
        fsi
}
boolean AttenteFinRecursion (int nbi) {
        si (nbi = 0)
            return true;
        sinon
            return (promesses.find_idsource(id));
        fsi
}
```

La fonction *find_idsource(id)* est définie sur la liste des promesses et retourne un booléen vrai si une promesse de cette liste a l'identificateur de la conversation courante comme source de la récursion.

```
int CalculCycle (int nb, string prod, int nbi) {
        int manque = commitments.SumQManquante(prod);
```

```
        string id = IdentificateurCourant();
        int cycle_recursion;
        si (nbi == 1)
                cycle_recursion = promesses.cycle_recursion(id);
        sinon
                cycle_recursion = 0;
        fsi
        return ( (manque div q[prod]) + Max(Cycle-courant, cycle_recursion)+ 1);
}
```

La fonction *cycle_recursion(id)* est définie sur la liste des promesses et retourne le cycle annoncé pour la demande récursive issue de l'interaction courante.

...

Comportement :

```
void run () {
        Pour chaque Cycle-courant de 1 à Cycle-fin
                si (Possible())
                        Production();
                sinon
                        Start-Interaction("Demander %input");
                fsi
                Tant que des messages sont dans la boîte aux lettres
                        ou des interactions initiées ont encore une barrière à franchir
                        ou des livraisons ont été promises pour ce cycle
                        ou des engagements sont prévus pour ce cycle
                        faire
                        Interaction ();
                ftant
        fpour
}
```

3 La construction de systèmes Applicatifs

Après la présentation de ces modèles d'agent et de ces schémas d'interaction, il reste à décrire la méthode de composition de la société d'agents que nous avons retenue pour construire un système multi-agents traitant l'application.

Nous allons tout d'abord présenter la déclaration d'une société organisée de manière statique pour notre seconde application, avant de discuter des modifications à apporter pour disposer d'une dynamique de réorganisation sur ce système.

3.1 Description de la société et de son organisation de départ

Nous présentons ici une organisation de départ des interactions, que nous désignons comme une mauvaise organisation, car les agents ont des préférences faibles pour les interactions qui permettent d'obtenir rapidement la quantité de biens nécessaire pour produire.

Nous utilisons ici une représentation du niveau de préférence basée sur l'intervalle [0; 9], où

9 est le meilleur niveau possible (et 0 le pire). La description de la société s'exprime alors par l'instanciation des modèles et des schémas présentée dans la figure 17-2, ce qui correspond à la grammaire suivante (dont l'implantation est présenté dans l'annexe A.III) :

<agt1> = Agent1_1 apply AgentInitial ("Bien x", "Bien A", 2, 10)
<agt2> = Agent1_2 apply AgentInitial ("Bien x", "Bien A", 2, 5)
<agt3> = Agent1_3 apply AgentInitial ("Bien x", "Bien A", 2, 3)
<agt4> =Agent2_1 apply AgentInter ("Bien A", "Bien B", 3, 3)
<agt5> = Agent2_2 apply AgentInter ("Bien A", "Bien B", 5, 5)
<agt6> = Agent2_3 apply AgentInter ("Bien A", "Bien B", 10, 10)
<agt7> = Agent3_1 apply AgentFinal ("Bien x", "Bien A", 10, 2)
<agt8> = Agent3_2 apply AgentFinal ("Bien x", "Bien A", 5, 2)
<agt9> = Agent3_3 apply AgentFinal ("Bien x", "Bien A", 3, 2)
<interaction1> = <PA> (Agent1_1 plays role F with preference level 5,
Agent2_1 plays role D with preference level 9)
<interaction2> = <PA> (Agent1_2 plays role F with preference level 5,
Agent2_1 plays role D with preference level 0)
...
<interaction9> = <PA> (Agent1_3 plays role F with preference level 5,
Agent2_3 plays role D with preference level 9)
<interaction10> = <PB> (Agent2_1 plays role F with preference level 5,
Agent3_1 plays role D with preference level 9)
<interaction11> = <PB> (Agent2_2 plays role F with preference level 5,
Agent3_1 plays role D with preference level 0)
...
<interaction18> = <PB> (Agent2_3 plays role F with preference level 5,
Agent3_3 plays role D with preference level 9)

Une simple modification des préférences permet rapidement d'obtenir un système ayant les mêmes types interactions, les mêmes agents, mais avec une organisation de départ différente. Cette organisation reste statique, selon les modèles d'agent que nous avons présentés, puisque nous n'avons pas défini dans ces modèles l'action de jugement qui intervenait dans le protocole d'interaction pour permettre la dynamique d'organisation des interactions.

3.2 Mise en place de l'organisation dynamique

La mise en place d'une organisation dynamique ne change nullement la déclaration de la société définissant le système, telle que nous l'avons présentée dans la section précédente. La dynamique se met en place dans les schémas d'interaction en précisant l'action de jugement (dans notre exemple (cf figure 17-3), elle se nomme JugerDemande), mais surtout dans les deux modèles d'agent *AgentInter* et *AgentFinal* en y précisant l'algorithme de cette action, dont voici un exemple :

Connaissances :

```
...
/* Un délai idéal possible selon le contexte du système */
        int reference = 0;
/* Une marge accordé par rapport à cette référence */
        int marge = 1;
```

Compétences :

```
...
float Qualite (int delai) {
        si ((reference + marge) > delai)
```

```
                return 1;
        sinon
                return ((reference + marge) / delai);
        fsi
}
JugerDemande (int cycle, string prod) {
        int delai = cycle - Cycle.Current();
        // Calcul de la qualité du délai annoncé
        float Q = Qualite(delai);
        // Jugement de cette qualité
        si (Q >= 0.9)
                // Renforcement de la préférence de la connaissance sociale courante
                Reinforce((int) (3*Q));
                // Apprentissage du contexte
                si ((PreferenceCourante() == 9) && ((reference + marge) > delai))
                        marge = delai - reference;
                fsi
        sinon
                // Affaiblissement de la préférence de la connaissance sociale courante
                Weaken((int) ((1 - Q)*3));
                // Apprentissage du contexte
                si (SumPreferenceKnowledges("Demande " + prod) == 0)
                        marge = marge + 1;
                        // Augmenter de 1 point le niveaux de préférence des autres connaissances
                        // sociale pour demander ce bien
                        ReinforcePreferenceOthersKnowledges("Demande " + prod, 1)
                fsi
        fsi
}
}
```

Où le renforcement et l'affaiblissement de la préférence (p) caractérisant la connaissance sociale courante correspondent à :

p = Min(p + x, 9) (pour *Reinforce(x)*),

p = Max(p - x, 0) (pour *Weaken(x)*).

Nous avons utilisé aussi des fonctionnalités de la plate-forme permettant aux agents de gérer leurs connaissances sociales (celles dont est issue l'interaction courante, aussi bien que d'autres connaissances sociales) :

PreferenceCourante() : retourne le niveau de préférence de la connaissance sociale courante,

SumPreferenceKnowledges(string motivation) : retourne la somme des préférences des connaissances sociales étant caractérisée par cette motivation

ReinforcePreferenceOthersKnowledges(string motivation, int x) : renforcement les préférences de toutes les connaissances sociales, autres que la connaissance sociale courante, qui sont caractérisées par cette motivation.

4 BILAN ET LIMITES CONSTATÉES

Les deux exemples de construction de systèmes nous permettent de dresser un premier bilan sur les objectifs que nous souhaitions atteindre avec notre plate-forme JAVAMA : modularité, conception incrémentale, réutilisation et modification aisée des descriptions de systèmes multi-agents.

La description de système à partir de JAVAMA s'exprime à différents niveaux (global, semi-global et local), ce qui permet naturellement d'aborder une construction modulaire de systèmes à partir des concepts de modèle d'agent et de schéma d'interaction. La taille du code correspondant à la description de la seconde application (organisation dynamique incluse) est présentée dans le tableau suivant :

TABLEAU 17-6 Taille du code décrivant la seconde application.

Niveaux de description	Nombre de lignes	Pourcentage
Modèles d'agent	2207	84,7%
Schémas d'interaction	193	7,4%
Société	206	7,9%
TOTAL	**2606**	**100%**

La description d'une application est aisément modifiable en ajoutant ou en enlevant un agent et les interactions auxquels il participe, de même qu'en incorporant ou non une dynamique d'organisation des interactions dans une description de système. La conception incrémentale d'une application est donc tout à fait possible.

Cette modularité permet également de réutiliser partiellement la description d'une application pour la description d'une autre à partir des modèles des agents et des schémas des interactions définies, même s'il s'avère nécessaire de réaliser quelques modifications dans ceux-ci (typage des variables contextuelles, etc.).

Les objectifs, que nous nous sommes fixés pour cette plate-forme sont donc atteint, mais à la lumière des exemples de construction présentés, nous pouvons envisager des améliorations pour introduire plus de souplesse pour la construction de systèmes multi-agents.

Par exemple pour la description de la seconde application, nous avons été amené à déclarer deux schémas d'interaction distincts (<*PA*> ou <*PB*>) qui se basent sur le même protocole d'interaction. Cette double déclaration est nécessaire dans l'état actuel de notre plate-forme, afin de différencier les connaissances sociales du potentiel d'un agent en fonction du type de bien qu'elles permettent de demander. Cette différenciation devrait néanmoins n'intervenir que lors de l'instanciation des interactions et non lors de la déclaration des schémas de ceux-ci, puisque dans les deux cas il s'agit d'une demande. Il serait alors plus judicieux d'utiliser la notion de type générique pour exprimer la motivation d'un schéma d'interaction et de préciser ce type générique uniquement lors de l'instanciation du schéma. Ceci devrait permettre d'accroître le pouvoir d'expression de notre plate-forme, en évitant des déclarations répétitives de schémas.

Un autre exemple de manque de souplesse correspond aux répétitions nécessaires pour l'instanciation des schémas. En effet dans notre application, tous les agents du niveau intermé-

diaire interagissent avec tous les agents des deux autres niveaux. Or les agents d'un même niveau forment un groupe d'agents instanciés à partir d'un même modèle. Il serait judicieux d'exprimer leurs interactions en se basant sur cette notion de groupe. Nous discuterons de cette possibilité à partir de l'évaluation de la plate-forme et de ses perspectives d'extension présentées dans le chapitre suivant.

Chapitre 18 Evaluation et perspectives

Ce chapitre présente une évaluation du formalisme description utilisé par notre plate-forme. Nous commençons par évaluer notre plate-forme vis-à-vis des buts que nous nous étions fixé lors de sa conception. Avant de présenter les perspectives d'évolution de notre modèle qui permettrait d'améliorer la puissance de description de la plate-forme.

1 EVALUATION DE LA PLATE-FORME JAVAMA

1.1 Modularité

Le modèle implanté dans JAVAMA propose différentes fonctionnalités facilitant la description de systèmes multi-agents. Ces fonctionnalités se sont révélées suffisantes pour nos expérimentations, en permettant de construire des agents et des interactions, et d'instancier ces dernières entre les agents pour décrire l'organisation de départ des interactions dans une société.

La description proposée par l'outil se base sur des modules élémentaires : les modèles d'agent et les schémas d'interaction. Ces modules se combinent grâce aux mécanismes d'instanciation et à la notion de rôle pour permettre de décrire l'organisation des interactions et la composition d'une société d'agents.

Notre utilisation de la notion de rôle se différencie de celle du modèle d'AALAADIN [Gutk98] par l'aspect statique des rôles que nous affectons aux agents. Contrairement à AALAADIN, nous n'avons pas envisagé d'affecter dynamiquement des rôles aux agents, mais seulement de leur permettre de jouer les rôles qui leur ont été affectés en fonction des motifs qui les poussent à

interagir.

Les types d'organisation qui peuvent être décrits à partir de notre plate-forme ne sont pas pour autant limités aux organisations statiques, puisque nous pouvons également décrire des organisations dynamiques grâce à la notion de préférence qu'intègre notre modèle.

1.2 Incrémentalité et réutilisation

La description modulaire d'un système et les outils associés permettent aux utilisateurs une conception incrémentale des applications et une réutilisation des éléments décrits.

Le concepteur d'un système peut aisément ajouter (ou enlever) un agent et ses interactions pour effectuer une mise au point incrémentale d'une application. De même, il peut ajouter (ou enlever) la dynamique d'organisation des interactions en exprimant des actions de jugement dans les protocoles d'interaction et dans les modèles des agents.

Le concepteur peut aussi réutiliser un modèle d'agent pour concevoir de nouveaux modèles, grâce à leur déclaration sous forme de classe Java et en développant ainsi un graphe d'héritage des modèles comme le propose LALO [Gauv97].

Il peut aussi réutiliser un ou plusieurs éléments de la description d'une application dans une autre, ce qui n'était pas le cas dans GTMAS [Chev94] qui permettait uniquement de réutiliser des modèles d'agent ou des agents. Les éléments réutilisables avec JAVAMA sont :

- les modèles d'agent,

- les instances de ces modèles (les agents identifiés),

- les schémas d'interaction,

- les instances de ces schémas et les agents concernés.

1.3 Evolutivité

L'implantation de JAVAMA est fondée actuellement sur un seul type de noyau pour les agents autonomes (basé sur les processus légers des API de JAVA) qui est utilisé pour opérationnaliser les systèmes décrits. Néanmoins, l'utilisateur peut tout à fait définir d'autres noyaux, pour lesquels il devra définir l'activité des agents (agents mobiles, processus), les mécanismes de communication nécessaires (envoi et réception par boîte aux lettres) et un mécanisme de synchronisation des cycles d'activités des agents. Dans ce cadre, le noyau microscopique des agents de la plate-forme MadKit [Gutk98] pourrait être utilisé par JAVAMA.

L'avantage de ce principe de noyau pour les agents autonomes est que notre plate-forme, tout comme MadKit, ne suppose pas de formalisme ou de modèle particulier pour décrire le comportement d'un agent. Le comportement d'un agent peut être décrit comme dans les exemples que nous avons présentés de manière procédurale ou encore comme dans le modèle d'agent de DIMA [Gues97] sous la forme d'un ATN (Augmented Transition Network).

Par ailleurs, les caractéristiques techniques des mécanismes de communication sont définies au niveau de ce noyau. Ce qui nous évite de spécifier dans notre description des protocoles d'interactions ces caractéristiques, comme c'est le cas dans le langage d'interaction proposé par MAGMA [Dema95] pour le mode de communication synchrone ou asynchrone. L'utilisation d'un mode de communication spécifique correspond dans le cadre de JAVAMA à utiliser un

noyau spécifique fournissant le mécanisme de communication adéquat.

2 BILAN, LIMITES CONSTATÉES ET PERSPECTIVES

JAVAMA implémente notre modèle de description d'un système multi-agents à partir des trois niveaux que nous avons identifiés : agent, interaction et organisation. Ce modèle intégre aussi la description de la réorganisation du système par la définition de processus d'évaluation, de jugement et de modification des préférences. Le formalisme ainsi défini permet d'obtenir à partir de telles descriptions des systèmes multi-agents opérationnelles.

Certes, l'utilisation de ce formalisme de description dans JAVAMA facilite la description des interactions liant les agents d'un système, mais cette description s'avère lourde et fastidieuse lorsque le système comporte beaucoup d'agents et beaucoup d'interactions entre ceux-ci (par exemple, l'utilisation de 2 schémas d'interaction et l'instanciation de 18 interactions entre les 9 agents composants l'application présentée dans le chapitre précédent).

Nous avons déjà évoqué les améliorations envisagées pour une description moins fastidieuse : type générique pour exprimer la motivation des schémas et notion de groupe pour l'instanciation des interactions. Nous présentons ici les perspectives qu'offre cette notion de groupe pour la description de système multi-agents avec notre approche, ainsi que les possibilités d'ouverture de notre description des interactions pour la prise en compte de communications indirectes.

2.1 La notion de groupe

Un groupe est un agrégat d'agents. Certes notre modèle permet déjà de désigner plusieurs agents pour jouer un même rôle dans un schéma d'interaction, ce qui permet ainsi d'exprimer et de réaliser la diffusion restreinte de message. Mais, ce type de définition n'évite nullement les redites que nous avons évoquées pour l'instanciation des schémas. La notion de groupe telle que nous avons commencé à l'introduire dans notre plate-forme se compose de deux approches de ce concept : une approche descriptive et une approche opérationnelle.

L'approche descriptive consiste à utiliser uniquement l'agrégat pour décrire plus facilement les interactions similaires (par exemple dans le chapitre 17: instancier le schéma d'interaction <*PA*> entre deux groupes G1 et G2, où G1 est composé des agents basés sur le modèle *AgentInitial* et G2 correspond au regroupement des agents basés sur le modèle *AgentInter*). Il serait alors possible d'introduire un mécanisme d'instanciation plus complexe des interactions permettant de décrire en une seule fois les 9 interactions qui prennent place entre ces agents dans l'application traitée.

L'approche opérationnelle consiste à représenter le groupe comme un agent particulier réalisant l'interfaçage entre ses membres et les autres agents du système. La notion de groupe revient alors à introduire un agent représentant les membres du groupe pour leurs interactions avec les autres agents du système (cette notion de représentation a aussi été adopté dans MadKit [Gutk98]). Ce représentant peut néanmoins posséder aussi des connaissances individuelles relatives au domaine et donc participer à la résolution au même titre que les autres agents (par exemple, si l'on considère cette notion de groupe pour le groupe G1 : l'agent Agent1_1 pourrait être

désigné comme représentant de ce groupe).

Cette double notion de groupe nécessite une remise en cause de la grammaire de description d'une société, où il faut distinguer les agents comme des composants élémentaires de la société et les groupes comme des agrégats d'agents. Nous parlons alors de composant et d'instanciation de composant plutôt que d'agent.

2.2 Du groupe au système

Un système peut alors être lui-même considéré comme une agrégation de composants incluant la description de ceux-ci et l'instanciation des interactions entre ceux-ci. Nous pouvons alors reprendre la double approche, que nous avons présentée pour les groupes, pour cette notion de système.

L'avantage de l'approche descriptive est de permettre la mise en place d'une réutilisation globale d'une description d'un système comme sous-système dans une nouvelle application et ainsi d'introduire la récursivité dans notre grammaire de description. Nous parlons alors de composant d'un système pour désigner ses agents, ses groupes et ses sous-systèmes.

De plus, cette approche fournit un support pour la mise en place de la diffusion globale dans un système en disposant de la liste de tous les membres du système (ce qui est assimilable à la notion d'alias), ou d'un agent particulier connaissant tous les membres de celui-ci (ce qui est assimilable à la notion de mailing-list).

2.3 Une dynamique pour le groupe

Une autre extension envisagée est d'introduire en plus des deux approches une dynamique dans la composition d'un groupe. Celle-ci consiste à permettre aux composants de rejoindre ou de quitter un groupe et même de migrer d'un groupe à l'autre durant la résolution.

Nous n'avons pas encore traité ce problème de la dynamique des groupes, mais, selon le cadre de description que nous venons de redéfinir, nous pensons qu'elle permettrait par la même occasion de prendre en compte la description et l'opérationnalisation de système ouvert selon notre approche. Toutes ces perspectives, nous amène à penser que la fusion de notre modèle avec celui d'AALAADIN [Gutk98] présenterait un grand intérêt.

2.4 Prise en compte des communications indirectes

Notre approche permet de décrire des systèmes en se basant sur la description de schémas d'interaction. La construction d'un système revient alors à un langage de description multi-agents basé sur l'équation SMA = A $\oplus$ I où A correspond à la déclaration des modèles d'agent, I à la déclaration de schémas d'interaction et où l'opérateur $\oplus$ correspond à l'instanciation des modèles d'agent, des schémas d'interaction et de la structure organisationnelle par la même occasion.

Les schémas d'interaction ont été présentés dans ce mémoire uniquement dans le cadre de communications directes entre les agents se basant sur l'envoi et la réception de messages. Si l'on considère maintenant l'utilisation d'une structure commune pour interagir que nous appellerons environnement, les communications indirectes permettant aux agents d'interagir via un environnement se basent sur deux mécanismes : un mécanisme d'action dans l'environnement ou un mécanisme de perception d'un événement dans celui-ci. Cela suppose une description de la structure, et un modèle pour le fonctionnement de l'environnement, que celui-ci soit une envi-

ronnement de simulation [Drog93], ou un environnement de type tableau noir [Chev93]. Si nous supposons qu'un tel modèle existe, nous pouvons considérer alors qu'il correspond à un agent particulier du système, qu'un événement est un message émanant de cet agent et qu'une action est un message envoyé à celui-ci.

Nous pourrions alors décrire des interactions utilisant des communications indirectes, en considérant le rôle particulier de cet agent dans les schémas d'interaction et en incorporant à notre grammaire de description des protocoles d'interaction, les actions de transition suivantes :

<action> = **act to** <environnement> **:** <type d'action>

<perception> = **perceive from** <environnement> **:** <événement>

Où <type d'action> peut être l'ajout, le retrait ou la modification d'un noeud d'un tableau noir, ou encore l'ajout, le retrait ou la modification d'un objet dans un environnement de simulation. De même que <événement> peut être considéré comme la description d'un signal de modification dans un tableau noir (ajout, retrait ou modification dans un niveau) ou encore comme la description d'une perception d'un type d'objet (éventuellement selon un rayon de perception caractérisant l'agent concerné par cette perception).

Nous incorporerions alors par ce biais l'environnement à notre équation de description de système : SMA = (A + E) $\oplus$ I. Il ne manquerait plus alors à notre approche qu'à intégrer un modèle de description de l'environnement (topologie d'un environnement de simulation ou niveaux pour un tableau noir), et de son fonctionnement (éventuellement avec des lois régissant celui-ci) [Magn96].

Chapitre 19 Conclusions et Perspectives

Dans ce chapitre, nous présentons une synthèse des problèmes abordés dans ce mémoire, ainsi que les solutions proposées pour les résoudre. Ensuite, nous précisons quelques perspectives de recherche de ces travaux.

1 Problématique abordée

Nous proposons un modèle de description et d'opérationnalisation automatique de systèmes composés d'agents communicants basé sur la notion d'interaction. Nous avons montré que ce modèle permet de définir des systèmes multi-agents clos pour lesquels une organisation des interactions peut être décrite selon l'aspect statique et dynamique.

Nous avons d'abord montré que la description et l'opérationnalisation de systèmes multi-agents nécessitaient de prendre en compte des concepts de différents niveaux : agent, interaction et organisation. Nous avons alors adopté différents niveaux de description pour ces concepts : le concept d'agent correspondant à un niveau local, celui d'interaction à un niveau semi-local et celui d'organisation à un niveau global. Néanmoins, nous avons également montré que, selon le principe d'autonomie de décision des agents, l'opérationnalisation d'une telle description nécessite la mise en place, la gestion des interactions et leur organisation au niveau des agents.

Nous avons alors bâti à partir de ceci une approche descriptive pour le prototypage de systèmes multi-agents à partir d'une description des agents, des interactions et de la société organisée composée par ces agents et ces interactions.

La dernière étape de la description consiste à décrire l'instanciation des modèles et des schémas pour spécifier une structure de départ pour les interactions et des alternatives à cette structuration au moyen de préférences ordonnant les possibilités d'interaction des agents. Nous décrivons ainsi l'aspect statique du concept d'organisation. Mais notre description permet également de prendre en compte l'aspect dynamique de l'organisation en décrivant les actions d'évaluation et de jugement permettant aux agents de modifier leurs préférences vis-à-vis des interactions en fonction d'heuristiques estimant leur satisfaction après l'utilisation de celles-ci.

Notre approche n'est pas uniquement descriptive, nous proposons également les mécanismes de liaison et d'instanciation permettant de rendre une telle description opérationnelle et cela de manière automatique en créant par instanciation les agents autonomes correspondant. Ces agents sont composés de trois couches : un noyau qui correspond à la couche primaire regroupant les fonctions de base d'un agent communicant autonome (lui permettant d'être une entité active qui peut communiquer et synchroniser son activité avec celle des autres agents), une couche comportementale qui est décrite par un modèle d'agent et enfin une couche sociale qui correspond aux représentations et aux mécanismes permettant aux agents de gérer par eux-même les interactions auxquelles ils participent.

Nous avons expérimenté notre approche décentralisée des interactions et de leur organisation à partir de la boîte à outils GTMAS, avant de développer une deuxième plate-forme JAVAMA qui permet de prototyper des systèmes multi-agents selon notre approche descriptive et opérationnelle. Ce type de prototypage permet de disposer d'un cadre de modulaire et incrémentale pour la conception de système multi-agents permettant une réutilisation des éléments de description.

2 PERSPECTIVES

Dans les sections suivantes, nous énumérons les évolutions possibles que nous envisageons pour les différents aspects abordés dans cette thèse.

2.1 Réorganisation des interactions

L'organisation des interactions, telle que nous la proposons dans les chapitres 11 et 12 de la deuxième partie, ne réalise qu'un réajustement des préférences attribuées aux connaissances sociales. Nous envisageons d'étendre cette réorganisation par les deux aspects suivants :

- nous voulons étudier et intégrer à notre approche des mécanismes de négociation qui permettraient de ne pas uniquement réorganiser un système en modifiant les préférences, mais aussi en négociant des modifications de l'ensemble des composants des connaissances sociales et en particulier au niveau des protocoles d'interaction ;

- nous voulons également étudier de plus près un aspect intéressant de notre gestion des préférences : la co-construction d'une interaction à partir de la notion de confiance et d'attention. En effet, cette co-construction nous semble présenter une approche intéressante de la notion de pouvoir social. Le pouvoir social n'est pas uniquement une notion intervenant au niveau de la force illocutoire des actes de communication. Il fait aussi intervenir le bon vouloir des agents. Par exemple, un ordre d'un lieutenant vers un simple soldat ne sera exécuté et donc pris en compte par celui-ci uniquement s'il reconnaît ce pouvoir au lieutenant. Le pouvoir n'est

pas un don inné, qu'un individu possède et exerce sur les autres (sinon aucune remise en cause du pouvoir n'est possible et la révolution française n'aurait jamais eu lieu). Le pouvoir social est attribué à un agent par les autres et c'est en lui attribuant ce pouvoir qu'ils acceptent d'obéir aux ordres émanant de celui-ci. Un agent peut donc attribuer un pouvoir à un autre, mais il peut également lui retirer ce pouvoir, ce qui ouvre de nouvelles perspectives pour la réorganisation.

2.2 Cohérence et validité d'une description de système

Notre approche met l'accent sur la réutilisation, la modularité et la souplesse des descriptions, et fournit un langage de description de système. Mais pour qu'elle constitue un langage de programmation multi-agents, il lui manque des mécanismes vérifiant la cohérence et la validité des descriptions :

- la cohérence correspond pour nous à envisager une vérification syntaxique et sémantique des systèmes décrits. Ceci revient, par exemple, à vérifier la correspondance des appels d'actions décrites dans les protocoles d'interactions et les actions sociales définies aux niveaux des modèles d'agent lors de l'instanciation des schémas d'interaction (de la même manière que les langages à objet permettent de vérifier la sémantique des appels de méthodes par rapport à leurs définitions) ;

- la validité correspond pour nous à envisager deux aspects : la validation de chaque protocole d'interaction comme nous l'avons évoqué dans le chapitre 13 de la deuxième partie, et la vérification de l'absence d'indéterminisme entre ces protocoles selon l'hypothèse adoptée au chapitre 11 de la deuxième partie.

2.3 Composants et réutilisation des composants

En ce qui concerne les composants élémentaires et leur réutilisation, nous envisageons de poursuivre nos travaux sur la notion de groupe (statique et dynamique), afin de permettre l'extension de notre approche à la description de systèmes ouverts et une réutilisation plus aisée des agrégats d'agent décrits (groupe ou système).

2.4 Développement d'applications

En ce qui concerne le développement d'applications à partir de notre plate-forme. Nous envisageons de réaliser un environnement de développement graphique permettant une conception plus conviviale des applications, en utilisant une construction graphique d'une société d'agents, des agents, et de leurs interactions. Nous voulons également développer une interface graphique temps-réel permettant de suivre l'évolution de l'activité du système et de la déboguer.

BIBLIOGRAPHIE

[Baei96] C. Baeijs and Y. Demazeau. "Les organisations dans les systèmes multi-agents." *Actes de la Journée sur les Systèmes Multi-Agents - PRC-GDR Intelligence Artificielle*, Toulouse, Février 1996.

[Barb95] M. Barbuceanu and M.S. Fox. "COOL: A language for describing coordination in Multi-Agent Systems." In V. Lesser, editor, *First International Conference On Multi-Agent Systems*, pages 17–24, San Francisco, June 1995. AAAI-Press.

[Bate84] G. Bateson. *La nature et la pensée*. Editions Seuil, 1984.

[Bela96] O. Belakhard and J. Ayel. "Modelling approach and tool for designing protocols for automated cooperation in multi-agent systems." In W. Van de Velde and J.W. Perram, editors, *Lecture Notes in Artificial Intelligence*, volume 1038, pages 100–115. Springer Verlag, 1996.

[Bour93a] T. Bouron. *Structures de communication et d'organisation pour la coopération dans un univers multi-agents : Une contribution à la définition d'un modèle de programmation agent basé sur les concepts d'engagement et d'acte de langage*. Thèse de doctorat, Université de Paris VI, 1993.

[Bour93b] T. Bouron. "Vers une approche plus systémique et évolutive de l'interaction." *Actes de la Journée sur les Systèmes Multi-Agents - PRC-GDR Intelligence Artificielle*, Montpelier, Décembre 1993.

[Bras92] C. Brassac. "Analyse de conversations et théorie des actes de langage." *Cahiers de linguistique française*, 13:62–75, 1992.

[Bras94] C. Brassac. "L'interaction inter-agents : non litteralité et processualité." *Actes des 2èmes Journées Francophones IAD et SMA*, pages 3–14, Voiron, Mai 1994.

[Bras96a] C. Brassac, J. Almeida, N. Grégori, and V. Saint-Dizier. "La théorie des actes de langage en IAD : utilisations et limites." *Actes des 4èmes Journées Francophones IAD et SMA*, pages 229–249, Port Camargue, Avril 1996.

[Bras96b] C. Brassac and V. Chevrier. "Vers un réxamen du statut de l'interaction en systèmes multi-agents." *Interaction et Cognitions*, 1(1):3–22, 1996.

[Burm95] B. Burmeister, A. Haddadi, and K. Sundermeyer. "Generic, configurable, cooperation protocols for multi-agent systems." In C. Castelfranchi and J.P. Müller, editors,

Lecture Notes in Artificial Intelligence, volume 957, pages 157–171. Springer Verlag, 1995.

[Camp90] J.A. Campbell and M.P. D'Inverno. "Knowledge interchange protocols." In Y. Demazeau and J.P. Müller, editors, *Decentralized A.I.*, pages 63–80. Elsevier Science Publishers B.V. (North-Holland), 1990.

[Camp95] V. Camps and M-P. Gleizes. "Principes et évaluation d'une méthode d'auto-organisation." *Actes des 3èmes Journées Francophones IAD et SMA*, pages 337–348, Chambéry, Mars 1995.

[Carl94] P. Carle, A. Collinot, and K. Zeghal. "Concevoir des organisations : la méthode cassiopée." *Actes de la Journée sur les Systèmes Multi-Agents - PRC-GDR Intelligence Artificielle*, Paris, Décembre 1994.

[Cast90] C. Castelfranchi. "Social power: A point missed in multi-agent, dai and hci." In Y. Demazeau and J.P. Müller, editors, *Decentralized A.I.*, pages 49–62. Elsevier Science Publishers B.V. (North-Holland), 1990.

[Chev93a] V. Chevrier. *Etude et mise en oeuvre du paradigme multi-agents : De ATOME à GTMAS*. Thèse de doctorat, Université Henri Poincaré, Nancy I, 1993.

[Chev93b] V. Chevrier. "GTMAS : un outil pour la construction et l'évaluation de systèmes multi-agents." *Actes des Premières Journées Francophones IAD et SMA*, Toulouse, April 1993.

[Chev94] V. Chevrier. "GTMAS: A Tool for Prototyping and Assessing Design Choices in Multi-Agent Systems." *Actes des Quatorzièmes Journées Internationales d'Avignon IA 94*, pages 161–170, Paris, Mai 1994. EC2.

[Chev95] V. Chevrier and C. Brassac. "Non littéralité et diffusion de l'information dans les systèmes multi-agents." *Actes de la Conférence Européenne sur les Sciences Cognitives*, Saint-Malo, Avril 1995.

[Cohe90a] P.R. Cohen and H.J. Levesque. "Persistence, Intention and Commitment." In P.R. Cohen, J. Morgan, and E.M. Pollack, editors, *Intentions in communication*, pages 33–69. MIT Press, Cambridge, 1990.

[Cohe90b] P.R. Cohen and H.J. Levesque. "Rational Interaction as the basis for communication." In P.R. Cohen, J. Morgan, and E.M. Pollack, editors, *Intentions in communication*, pages 221–225. MIT Press, Cambridge, 1990.

[Cohe90c] P.R. Cohen, J. Morgan, and E.M. Pollack. *Intentions in communication*. MIT Press, Cambridge, 1990.

[Cohe95] P.R. Cohen and H.J. Levesque. "Communicative actions for artificial agents." In V. Lesser, editor, *First International Conference On Multi-Agent Systems*, pages 65–72, San Francisco, June 1995. AAAI-Press.

[Cove95] P. Coveney and R. Highfield. *Frontiers of complexity: the search for order in a chaotic world.* 1995.

[Davi80] R. Davis. "Report on the workshop on distributed artificial intelligence." *Sigart Newsletter*, pages 42–52, October 1980.

[Davi81] R. Davis and R.G. Smith. "Frameworks for cooperation in distributed problem solving." *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-11(1):61–70, January 1981.

[Davi82] R. Davis. "Report on the second workshop on distributed artificial intelligence." *Sigart Newsletter*, pages 13–23, April 1982.

[Deck87] K. Decker. "Distributed problem solving techniques: A survey." *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-17(5):485–519, Sept./Oct. 1987.

[Deck89] K. Decker, E.H. Durfee, and V.R. Lesser. "Evaluating research in cooperative distributed problem solving." In L. Gasser and M. Huhns, editors, *Readings in Distributed Artificial Intelligence*, volume 2, pages 485–519. Pitman, London, 1989.

[Deck95] K.S. Decker and V.R. Lesser. "Designing a Family of Coordination Algorithms." In V. Lesser, editor, *First International Conference On Multi-Agent Systems*, pages 73–80, San Francisco, June 1995. AAAI-Press.

[Dema90] Y. Demazeau and J.P. Müller. "Decentralized artificial intelligence." In Y. Demazeau and J.P. Müller, editors, *Decentralized A.I.*, pages 3–13. Elsevier Science Publishers B.V. (North-Holland), 1990.

[Dema95] Y. Demazeau. "From interactions to collective behaviour in agent-based systems." In V. Lesser, editor, *First International Conference On Multi-Agent Systems*, pages 117–132, San Francisco, June 1995. AAAI-Press.

[Dema97] Y. Demazeau. "Towards Multi-Agent Oriented Programming." In *First International Workshop on Multi-Agent Systems (slides)*, Boston, October 1997.

[Drog93] A. Drogoul. *De la simulation multi-agents à la résolution collective de problèmes.* Thèse de doctorat, Université de Paris VI, 1993.

[Dupu92] J-P. Dupuy. *Introduction aux sciences sociales. (Logique des phénomènes collectifs).* Ellipses, 1992.

[Durf87] E.H. Durfee, V.R. Lesser, and D. D. Corkill. "Coherent cooperation among communicating problem solvers." *IEEE Transactions on Computers*, C-36(11):1275–1291, November 1987.

[Durf91] E.H. Durfee and V.R. Lesser. "Partial global planning: A coordination framework for distributed hypothesis formation." *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 21(5):1167–1184, Sept./Oct. 1991.

[Erce91] J. Erceau and J. Ferber. "L'intelligence artificielle distribuée." *La recherche*, 22(233):750–758, Juin 1991.

[Erce93] J. Erceau and J. Ferber. "Introduction aux premières journées francophones d'intelligence artificielle distribuée et systèmes multi-agents." *Actes des Premières Journées Francophones IAD et SMA*, pages 1–8, Toulouse, Avril 1993.

[Færg94] O. Færgemand and A. Olsen. "Introduction to SDL'92." In *Computer Networks and ISDN Systems 26*, pages 1143–1167, 1994.

[Ferb89] J. Ferber. *Objets et agents : une étude des structures de représentations et de communications en Intelligence Artificielle*. Thèse de doctorat d'état, Université Pierre et Marie Curie de Paris VI, 1989.

[Ferb95] J. Ferber. *Les systèmes Multi-Agents (vers une intelligence collective)*. InterEditions, Paris, 1995.

[Ferb97] J. Ferber. "Les systèmes multi-agents : un aperçu général." *Technique et science informatiques*, Vol. 16(8):979–1012, 1997.

[Fiel97] R. Fielding, J. Gettys, J. Mogul, and T. Berners-Lee H. Frystyk. "Hypertext transfer protocol – http/1.1." Technical Report RFC 2068, Information Sciences Institute (ISI) of the University of Southern California (USC), January 1997. (ftp://ftp.ibp.fr/pub/rfc/rfc/rfc2068.txt.gz).

[Fini94] T. Finin, R. Fritzson, D. McKay, and R. McEntire. "KQML as an agent communication language." In ACM Press, editor, *Proceedings of the Third International Conference on Information and Knowledge Management*, November 1994.

[Fois96a] R. Foisel, V. Chevrier, and J.-P. Haton. "De l'organisation d'une société à sa réorganisation." *Actes de la Journée sur les Systèmes Multi-Agents - PRC-GDR Intelligence Artificielle*, pages 121–128, Toulouse, Février 1996.

[Fois96b] R. Foisel and L. Chapelier. "Architecture Multi-Agents d'un système de dialogue d'assistance." *Actes de la rencontre des étudiants chercheurs en informatique pour le traitement automatique de la langue (poster)*, pages 165–168, Gif-sur-Yvette, Septembre 1996.

[Fois96c] R. Foisel, V. Chevrier, and J.-P. Haton. "Improving global coherence by adaptive organization in a multi-agent system." *Second International Conference On Multi-Agent Systems (poster)*, page 435, Kyoto, Japan, December 1996. AAAI Press / MIT Press.

[Fois96d] R. Foisel, V. Chevrier, and J.-P. Haton. "Modélisation de l'organisation dans les systèmes multi-agents." *Actes des 4èmes Journées Francophones IAD et SMA (poster)*, Port Camargue, Avril 1996.

[Fois97] R. Foisel, V. Chevrier, and J.-P. Haton. "Un modèle pour la réorganisation de système multi-agents." *Actes des 5èmes Journées Francophones IAD et SMA*, pages 261-277, Colle sur Loup, Avril 1997.

[Fois98a] R. Foisel, V. Chevrier, and J.-P. Haton. "Modeling Adaptive Organizations." *Third International Conference On Multi-Agent Systems (poster)*, page 427-428, Paris, July 1998. IEEE Press.

[Fois98b] R. Foisel. "Construire des systèmes multi-agents à partir de schémas d'interactions." *Actes des 6èmes Journées Francophones IAD et SMA*, Pont-à-Mousson, Novembre 1998. (à paraître)

[Fox81] M.S. Fox. "An organizational view of distributed systems." *IEEE Transactions on Systems, Man and Cybernetics*, SMC-11(1):70–80, January 1981.

[Galb73] J. Galbraith. *Designing complex organizations*. Addision-Wesley, 1973.

[Gall90] J. R. Galliers. "The positive role of conflict in cooperative multi-agent systems." In Y. Demazeau and J.P. Müller, editors, *Decentralized A.I.*, pages 33–46. Elsevier Science Publishers B.V. (North-Holland), 1990.

[Gasp91] G. Gaspar. "Communication and belief changes in a society of agents: towards a formal model of an autonomous agent." In Y. Demazeau and J.P. Muller, editors, *Decentralized Artificial Intelligence*, volume 2, pages 245–255. Elsevier Science Publishers B.V. (North-Holland), 1991.

[Gass87] L. Gasser. "The 1985 workshop on distributed artificial intelligence." *AI Magazine*, pages 91–97, Summer 1987.

[Gass88] L. Gasser, C. Braganza, and N. Herman. "Implementing distributed ai systems using mace." In A. H. Bond and L. Gasser, editors, *Readings in Distributed Artificial Intelligence*, pages 445–450. Morgan Kaufmann, 1988.

[Gass92] L. Gasser. "An overview of D.A.I." In N. M. Avouris and L. Gasser, editors, *Distributed Artificial Intelligence: Theory and Praxis*, pages 9–30. Kluwer Academic Publishers, 1992.

[Gauv97] D. Gauvin, H. Marchal, and C. Saldanha. "LALO : un environnement de programmation ouvert pour des systèmes multi-agents." *Actes des 5èmes Journées Francophones IAD et SMA*, pages 91–92, La Colle sur Loup, Avril 1997. *Http://www.crim.ca/sbc/francais/lalo/architecture.html.*

[Geor84] M. Georgeff. "A theory of action for multi-agent planning." In *Proceedings of the National Conference on A.I.*, AAAI, August 1984.

[Glei94] M-P. Gleizes, P. Glize, and S. Trouilhet. "Etude des lois de la conversation entre agents autonomes." *Revue Internationale de Systémique*, 8(1):39–50, 1994.

[Gric75] H.P. Grice. "Logic and conversation in syntax and semantics." *Speech acts*, 3:41–58, 1975.

[Gric89] P. Grice. *Studies in the way of words*. Harvard University Press, Cambridge, 1989.

[Gues96] Z. Guessoum. *Un environnement opérationnel de conception et de réalisation de systèmes multi-agents*. Thèse de doctorat, Université de Paris VI, 1996.

[Gues97] Z. Guessoum, J-P. Briot, and M. Dojat. "Des objets concurrents aux agents autonomes." *Actes des 5èmes Journées Francophones IAD et SMA*, pages 93–107, La Colle sur Loup, Avril 1997.

[Gutk98] O. Gutknecht and J. Ferber. "A meta-model for the analysis and design of organizations in multi-agent systems." In Y. Demazeau, editor, *Third International Conference On Multi-Agent Systems*, pages 128–135, Paris, July 1998. IEEE.

[Hato91] J-P. Haton, N. Bouzid, F. Charpillet, M-C. Haton, B. Lâasri, H. Lâasri, P. Marquis, T. Mondot, and A. Napoli. *Le Raisonnement en Intelligence Artificielle*. InterEditions, Paris, 1991.

[Hewi77] C. Hewitt. "Viewing control structures as patterns of passing messages." *Artificial Intelligence*, 8(3):323–364, 1977.

[Juan95] G. Juanole, A. Serhrouchni, and D. Seret. *Réseaux de communication et conception de protocoles*. Hermés, Paris, 1995.

[Labi93] S. Labidi and W. Lejouad. "De l'intelligence artificielle distribuée aux systèmes multi-agents." Technical Report 2004, INRIA, Sophia Antipolis, Août 1993.

[Lena94] C. Lenay. "Organisation émergente dans les populations : biologie, éthologie, systèmes artificiels." *Intellectica*, 19(2):9–17, 1994.

[Ludw93] M. Ludwig and M. Courant. "Modèle de communication dans un système multi-agents." *Actes des 1ères Journées Francophones IAD et SMA*, pages 265–276, Toulouse, Avril 1993.

[Magn96] L. Magnin. *Modélisation et simulation de l'environnement dans les systèmes multi-agents*. Thèse de doctorat, Université de Paris VI, 1996.

[MARC96] Groupe MARCIA. "Auto-organisation = évolution de structures." *Actes de la Journée sur les Systèmes Multi-Agents - PRC-GDR Intelligence Artificielle*, pages 139–152, Toulouse, Février 1996.

[Marc97] P. Marcenac. "Modélisation de systèmes complexes par agent." *Technique et science informatiques*, 16(8):1013–1037, Octobre 1997.

[Mart92] F. Von Martial. "Coordinating plans of autonomous agents." In F. Von Martial, editor, *Lecture Notes in Artificial Intelligence*, volume 610. Springer Verlag, 1992.

[Masi89] G. Masini, D. Colnet, A. Napoli, D. Léonard, and K. Tombre. *Les langages à objets*. InterEditions, Paris, 1989.

[Mayf96] J. Mayfield, Y. Labrou, and T. Finin. "Evaluation of kqml as an agent communication language." In J-P. Müller M. J. Wooldridge and M. Tambe, editors, *Lecture Notes in Artificial Intelligence*, volume 1037, pages 347–360. Springer Verlag, 1996.

[Mint79] H. Mintzberg. *The Structuring of Organizations: a synthesis of the research*. Prentice-Hall, 1979.

[Monc94] F-R. Monclar. "Rôles et modes de coopération dans les systèmes à base de connaissances coopératifs." *Actes des Secondes Rencontre Nationales des Jeunes Chercheurs en Intelligence Artificielle*, pages 215–222, Marseille, Septembre 1994.

[Mori77] E. Morin. *La méthode (1) : La nature de la Nature*. Le seuil, 1977.

[Müll95] T. Müllen and M.P. Wellman. "A simple computational market for network information services." In V. Lesser, editor, *First International Conference On Multi-Agent Systems*, pages 283–289, San Francisco, June 1995. AAAI-Press.

[Nii86] H.P. Nii. "Blackboard Systems (Part I): The Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures." *AI Magazine*, 7(2):38–53, 1986.

[Peti84] F. Petit. *Introduction à la psychosociologie des organisations*. Privat, 1984.

[Popu93] P. Populaire, Y. Demazeau, O. Boissier, and J. Sichman. "Description et implémentation de protocoles de communication en univers multi-agents." *Actes des 1ères Journées Francophones IAD et SMA*, pages 241–252, Toulouse, Avril 1993.

[Quin92] J. Quinqueton, P. Reitz, and J. Sallantin. "Une approche multi-agents des environnements d'acquisition de connaissance." *Actes de la Journée sur les Systèmes Multi-Agents - PRC-GDR Intelligence Artificielle*, Nancy, Décembre 1992.

[Riva92] N. Cevallos Rivas, M-P. Gleizes, P. Glize, C. Piquemal-Baluard, and S. Trouilhet. "Du rôle des agents à la typologie des accointances." *Actes de la Journée sur les Systèmes Multi-Agents - PRC-GDR Intelligence Artificielle*, Nancy, Décembre 1992.

[Rous94] D. Rouseau, B. Moulin, and G. Lapalme. "Une approche multi-agents pour modéliser les conversations." *Actes des 2èmes Journées Francophones IAD et SMA*, pages 3–14, Voiron, Mai 1994.

[Schm91] K. Schmidt. "Cooperative work: a conceptual framework." In B. Brehmer and J. Leplat, editors, *Distributed decision making: cognitive models for cooperative work*, pages 75–110. J. Wiley & Sons, 1991.

[Sear72] J.R. Searle. *Les actes de langage*. Hermann, Paris, 1972.

[Sear79] J.R. Searle. *Expressions and Meaning, chapter 1: Taxinomy of Illocutionary Acts*. Cambridge University Press, 1979.

[Sear85] J.R. Searle and D. Vanderveken. *Foundations of illocutionary logic*. Cambridge University Press, Cambridge, 1985.

[Shoh93] Y. Shoham. "Agent-oriented programming." *Journal of Artificial Intelligence*, 60:51–92, November/December 1993.

[Sich92] S. Sichman, Y. Demazeau, and O. Boissier. "When can knowledge-based systems be called agents?" In *Brazilian Symposium on A.I.*, pages 172–185, Rio de Janeiro, 1992.

[Sich95a] S. Sichman. *Du raisonnement social chez les agents*. Thèse de doctorat, Université de l'Institut National Polytechnique de Grenoble, 1995.

[Sich95b] S. Sichman and Y. Demazeau. "Exploiting social reasoning to deal with agency level inconsistency." In V. Lesser, editor, *First International Conference On Multi-Agent Systems*, pages 352–359, San Francisco, June 1995. AAAI-Press.

[Smit80] R. G. Smith. "The contract net protocol: High-level communication and control in a distributed problem solver." *IEEE Transactions on Systems, Man, and Cybernetics*, C-29(12):1104–1113, December 1980.

[Stru95] E. Le Strugeon. *Une méthodologie d'auto-adaptation d'un système multi-agents cognitifs*. Thèse de doctorat, Université de Valenciennes et du Hainaut-Cambrésis, 1995.

[Tata97] S. Tata. *Modélisation d'agent OSI en SDL'92*. Université Henri Poincaré Nancy I, Nancy, Juillet 1997. Mémoire de DEA Informatique.

[Trou93] S. Trouilhet. *Représentation et traitement des connaissances sociales chez l'agent : application à l'environnement multi-agent SYNERGIC.* Thèse de doctorat, Université Paul Sabatier de Toulouse, 1993.

[Trou94] S. Trouilhet. "Méthode d'acquisition des connaissances sociales pour l'organisation d'un société d'agents." *Actes des 2èmes Journées Francophones IAD et SMA*, pages 27–38, Voiron, Mai 1994.

[Vand88] D. Vanderveken. *Les actes de discours.* Mardaga, Bruxelles, 1988.

[Well93] M.P. Wellman. "A market-oriented programming environment and its application to distributed multi-commodity flow problems." *Journal of Artificial Intelligence Research*, 1(1):1–23, 1993.

Annexe A Description d'une application multi-agents

Nous présentons ici un extrait du code définissant une application multi-agents avec notre plateforme JAVAMA, afin d'illustrer l'utilisation de la plateforme. Cette application a été présentée et analysée dans le chapitre 17 de la partie III. Nous ne traduirons pas ici l'intégralité des pseudo-algorithmes de cette application, mais seulement les éléments explicitant le principe de leur traduction.

.I LES SCHÉMAS D'INTERACTION

La déclaration de schémas d'interaction prend place dans la classe JAVA décrivant la société et son organisation que nous présenterons dans la section III. Elle se base sur l'utilisation de la méthode *InteractionPattern* de la classe *MultiAgentSystem* (module *MultiAgentSystem*) qui crée un instance de la classe *InteractionDescriptor* (module *Interaction*) correspondant à la représentation d'un schéma d'interaction. Ceci correspond dans notre application à l'extrait suivant :

```
InteractionDescriptor PA, PB;
PA = SMA.InteractionPattern (new ProtocolDemandGeneric(), "Demander A");
PB = SMA.InteractionPattern (new ProtocolDemandGeneric(), "Demander B");
```

Une caractèristique essentielle de la déclaration de ces schémas est l'utilisation d'une instance d'un protocole d'interaction (classe *ProtocolDemandGeneric*). La déclaration d'un protocole d'interaction correspond à une classe héritant des propriétés générales à tout protocole (classe *Protocol*).

Voici un extrait de la classe de déclaration du protocole d'interaction de demande :

```
/** DÉFINITION D'UN PROTOCOLE DE DEMANDE "GÉNÉRIQUE" */

public class ProtocolDemandGeneric extends Protocol {

protected void Define () {                          ⇐ Définition du protocole

/* Déclarations des variables contextuelles */
DeclaredContext = new ContextualVariables();
DeclaredContext.CreateVariable("Nb", new ParamInteger());
DeclaredContext.CreateVariable("Prod", new ParamString());
DeclaredContext.CreateVariable("Cycle", new ParamInteger());
DeclaredContext.CreateVariable("IdSource", new ParamString());
DeclaredContext.CreateVariable("NbI", new ParamInteger());
DeclaredContext.CreateVariable("Finrecursion?", new ParamBoolean());
DeclaredContext.CreateVariable("Finattente?", new ParamBoolean());
DeclaredContext.CreateVariable("Barriere-franchie", new ParamBoolean());
```

```
/* Déclaration des appels d'actions */
                /* Fonctions et procédures */
Procedure barriere-Fournisseur = new Procedure("DeclareBarriere", "F", {}[1]);
Procedure barriere-Demandeur = new Procedure("DeclareBarriere", "D", {});
Function quantite-demandee = new Function("QuantiteDemandee", "Demandeur", {"Prod"}, "Nb");
Function produit-demande= new Function("Produit", "Demandeur", {}, "Prod");
...

/* Agrégation d'appels d'actions*/
Vector Initialisation = {barriere-Fournisseur, barriere-Demandeur, produit-demande, quantite-demandee,
idconvsource};
Vector EngageRecursion = {recursion, engagementrecursif};
Vector EngageCalculCycle = {calculcycle, engagement};
Vector Livraison = {preparerlivraison, desengagement};
Vector Mémoriser-et-JUGER = {memorisercycle, JUGEMENT};
Vector Incorporer = {effacermemorisation, incorporer};

/* Déclaration des actions de transition */
                /* Prédicats */
Function barriereF = new Function("Barriere", "Fournisseur", {}, "Barriere-franchie");
Event Barriere-F = new Event(barriereF);
...
                /* Envoi et réception de message */
SendedAction senddemande = new SendedAction("Demande $Nb $Prod", "Demandeur", "Fournisseur");
Event recvdemande = new ReceivedEvent("Demande $Nb $Prod", "Demandeur", "Fournisseur");
SendedAction sendannonce = new SendedAction("Annoncé pour $Cycle", "Fournisseur", "Demandeur");
Event recvannonce = new ReceivedEvent("Annoncé pour $Cycle", "Fournisseur", "Demandeur");
SendedAction sendlivraison = new SendedAction("Livraison $Nb $Prod", "Fournisseur", "Demandeur");
Event recvlivraison = new ReceivedEvent("Livraison $Nb $Prod", "Fournisseur", "Demandeur");

/* Déclaration des étapes et les actions associées */
State state0 = new State("0", Initialisation);
State state1 = new State("1", Recursion);
State state2 = new State("2", Engage-CalculCycle);
State state3 = new State("3", Memoriser-JUGER);
State state4 = new State("4");
State state5 = new State("5", Livraison);
State state6 = new State("6", Incorporer);

/* Déclaration des noeuds et de leur enchaînement*/
protocol = new ProtocoleDescriptor();

/* Déclaration du noeud Initial et Final */
protocol.addinitialstate (state0);
protocol.addterminatedstate (state6);
/* par défaut les noeuds intermédiaires correspondent aux étapes qui ne sont pas initial et final */

/* Déclaration des transitions entre les noeuds */
                /* qui concerne le Demandeur */
protocol.addtransition (state0, senddemande, state2);
protocol.addtransition (state2, recvannonce, state3);
```

1. Notation adoptée pour les listes. Les listes sont gérées et représentées au moyen de la classe Vector de JAVA.

```
protocol.addtransition (state3, Barriere-D, state4);
protocol.addtransition (state4, recvlivraison, state6);
            /* qui concerne le Fournisseur */
protocol.addtransition (state0, recvdemande, state1);
protocol.addtransition (state1, AttenteFinRecursion, state2);
protocol.addtransition (state2, sendannonce, state3);
protocol.addtransition (state3, Barriere-F, state4);
protocol.addtransition (state4, AttenteCycle, state5);
protocol.addtransition (state5, sendlivraison, state6);
}
}
```

.II LES MODÈLES D'AGENTS

La déclaration d'un modèle d'agent s'effectue en définissant une classe JAVA décrivant ce modèle. Cette classe hérite de la classe *AgentInteraction* (module *LogicalAgentModel*) qui fournit les mécanismes gérant les interactions et les communications.

D'autres classes sont utilisées pour cette application spécifique, mais nous ne les détaillerons pas puisque nous avons déjà présenté les principes de leur algorithmique. Il s'agit des classes *Commitments* et *Stock* pour de la classe *AgentInitial* de notre application, dont voici un extrait :

```
public class AgentInitial extends AgentInteraction{

// Gestion locale des engagements pris par l'agent
protected Commitments commitments;

// Les stocks
protected Stock input;
protected Stock output;

/** Mécanisme de liaison pour les appels de procédure définis dans le(s) protocole(s) d'interaction */
public void executeProcedure (String name, ContextualVariables parameter, InteractionKnowledge
current_knowledge) {
// Où name est le nom de la procédure appelée,
// parameter est la liste des variables (nom, et valeur (typée)), ainsi que l'identifiant de la conversation,
// current_knowledge est la connaissance sociale dont est issue l'interaction.
if (name.equals("Desengagement")) {
            commitments.del(parametre.id());
} else if (name.equals("PreparerLivraison")) {
...
} else
System.out.println (">>Procedure " + name + " : Not yet implanted");
}

/** Mécanisme de liaison pour les appels de fonction définis dans le(s) protocole(s) d'interaction */
public Param executeFunction (String name, ContextualVariables parameter, InteractionKnowledge
current_knowledge) {
if (name.equals("AttenteFinRecursion")) {
            Param b = new ParamBoolean();
            b.Modify(new Boolean(true));
            return b;
```

```
} else if (name.equals("AttenteCycle")) {
            Integer cycle = (Integer) parameter.Variable("Cycle").Value();
            Param b = new ParamBoolean();
            b.Modify(new Boolean((Cycle.Current() == cycle.intValue())));
            return b;
} else if (name.equals("Barriere")) {
...
} else {
System.out.println (">>Fonction " + name + ": Not yet implanted");
return null;
}
}

...

/** DÉFINITION DU COMPORTEMENT DE CE MODÈLE D'AGENT */
public void run () {
            /** Initialisation du comportement */

            // Création d'une liste d'engagement vide
            commitments = new Commitments();
            // Création des stocks initiaux de biens
            input = new Stocks();
            output = new Stocks();

            // Différentiation de l'initiation en fonction de l'identifiant de l'agent des stocks
            if (_name.equals("Agent1_1")) {
                  input.addproduit (new Produit("Bien x", 2));
                  output.addproduit (new Produit("Bien A", 10));
            }
            else if (_name.equals("Agent1_2")) {
                  input.addproduit (new Produit("Bien x", 2));
                  output.addproduit (new Produit("Bien A", 5));
            }
            else if (_name.equals("Agent1_3")) {
                  input.addproduit (new Produit("Bien x", 2));
                  output.addproduit (new Produit("Bien A", 3));
            }

            /** Corps du comportement */
            for (; ((Cycle.Current() >= 0) && (Cycle.Current() < Cycle.Finish()));) {
            ...


            for(; (     (Box().Count() > 0) ||
                        ( CountInteractionInitiatedConcernedBy (Cycle.Current()) > 0 ) ||
                        ( commitments.CountConcernedBy(Cycle.Current()) > 0 )
                  ) ; ) {
            Interaction ();
            }
            } // End Comportement
} // End run
} // Class AgentInitial
```

.III DESCRIPTION DU SYSTÈME APPLICATIF

Après la conception de ces modèles d'agents et de ces schémas d'interaction, il reste encore à préciser comment nous proposons de décrire une société d'agents avec notre plate-forme. Cette description se base sur les classes *InteractionDescriptor* (du module *Interaction*) et *MultiAgentSystemDescriptor* (module *MultiAgentSystem*).

Nous commençons par montrer comment nous exprimons l'instanciation de nos modèles d'agents et de nos schémas d'interaction, puis nous donnons l'exemple de la déclaration d'une société organisée de manière statique en présentant le pseudo-code correspondant.

1 Instanciation d'un modèle d'agent

La composition d'un système s'explicite grâce à la méthode *addAgent* de la classe *MultiAgentSystemDescriptor* qui permet d'instancier la description d'un agent en attribuant un identifiant à cet agent et en désignant le modèle sur lequel se basera sont comportement. Le pseudo-code suivant illustre cette instanciation :

```
MultiAgentSystemDescriptor SMA;
SMA.addAgent("Agent1_1", new AgentInitial());
```

2 Instanciation d'un schéma d'interaction

Les interactions s'explicitent à partir des schémas d'interaction en précisant les participants aux interactions. Préciser ces participants correspond à utiliser la classe *AgentsInvolved* (module *Interaction*) pour désigner le nom des participants, leur rôle et leur préférence vis-à-vis d'une interaction, avant d'instancier celle-ci entre ces participants selon le schéma d'interaction choisi. Ceci correspond au pseudo-code suivant :

```
AgentsInvolved participants = new AgentsInvolved();
participants.DefineRole("Agent2_1","Fournisseur", 5);
participants.DefineRole("Agent3_1","Demandeur", 9);
SMA.Instanciated_Interaction(PB, participants);
```

Ce qui signifie qu'une possibilité d'interaction de demande de biens de type B existe entre les agents Agent2_1 et Agent3_1 qui tiennent respectivement les rôles de fournisseur et de demandeur. L'entier précisé lors de l'affectation d'un rôle à un participant correspond à la déclaration d'un niveau de préférence (dans l'intervalle [0; 9]) de ce participant pour cette interaction.

3 Description d'une société d'agents

Nous présentons ici un exemple de la déclaration d'une société d'agents. La déclaration d'une société correspond à la définition d'une classe héritant de la classe *SystemBasic* (module *MultiAgentSystem*), qui fournit les fonctionnalités permettant de préciser la composition du système ainsi défini et de rendre celle-ci opérationnelle.

```
/** DÉFINITION D'UNE SOCIÈTÉ */

public class Application extends SystemBasic {

public void init () {                          ⇐ Définition de la société

/* Les schémas d'interactions */
InteractionDescriptor PA, PB;
```

```
PA = SMA.InteractionPattern (new ProtocolDemandGeneric(), "Demander A");
PB = SMA.InteractionPattern (new ProtocolDemandGeneric(), "Demander B");

/* Les instanciations des schémas */
/* interaction de demande de bien de type B de Agent3_1 vers Agent2_1 */
AgentsInvolved participants = new AgentsInvolved();
participants.DefineRole("Agent2_1","Fournisseur", 5);
participants.DefineRole("Agent3_1","Demandeur", 9);
SMA.Instanciated_Interaction(PB, participants);

/* interaction de demande de bien de type B de Agent3_1 vers Agent2_2 */
participants = new AgentsInvolved();
participants.DefineRole("Agent2_2","Fournisseur", 5);
participants.DefineRole("Agent3_1","Demandeur", 0);
SMA.Instanciated_Interaction(PB, participants);
...
/* interaction de demande de bien de type B de Agent3_3 vers Agent2_3 */
participants = new AgentsInvolved();
participants.DefineRole("Agent2_3","Fournisseur", 5);
participants.DefineRole("Agent3_3","Demandeur", 9);
SMA.Instanciated_Interaction(PB, participants);

/* interactions de demande de bien de type A de Agent2_1 vers Agent1_1 */
participants = new AgentsInvolved();
participants.DefineRole("Agent1_1","Fournisseur", 5);
participants.DefineRole("Agent2_1","Demandeur", 9);
SMA.Instanciated_Interaction(MA, participants);
...

/* Les instanciations des modèles d'agents */

/* Les agents du niveau initial */
SMA.addAgent("Agent1_1", new AgentInitial());
SMA.addAgent("Agent1_2", new AgentInitial());
SMA.addAgent("Agent1_3", new AgentInitial());

/* Les agents du niveau intermédiaire */
SMA.addAgent("Agent2_1", new AgentIntermediaire());
SMA.addAgent("Agent2_2", new AgentIntermediaire());
SMA.addAgent("Agent2_3", new AgentIntermediaire());

/* Les agents du niveau terminal */
SMA.addAgent("Agent3_1", new AgentTerminal());
SMA.addAgent("Agent3_2", new AgentTerminal());
SMA.addAgent("Agent3_3", new AgentTerminal());
}
}
```

Université HENRI POINCARÉ NANCY 1

FACULTÉ DES SCIENCES

Monsieur **FOISEL Rémy**

DOCTORAT de l'UNIVERSITE HENRI POINCARE, NANCY-I

en INFORMATIQUE

VU, APPROUVÉ ET PERMIS D'IMPRIMER

Nancy, le 23 novembre 1998 - n° 146

Le Président de l'Université

UNIVERSITÉ HENRI POINCARÉ NANCY 1 LE PRÉSIDENT

J.P. FINANCE

3C-F9

Résumé

Cette thèse, effectuée au sein de l'équipe RFIA (Reconnaissance des Formes et Intelligence Artificielle) du LORIA, présente le résultat de notre travail sur l'étude des interactions et de leur réorganisation pour la construction de systèmes multi-agents.

Notre travail s'est articulé autour de deux phases complémentaires qui correspondent à la proposition d'un modèle de réorganisation des interactions et la définition d'un formalisme de description de systèmes. Ce formalisme de description est évidemment couplé à des mécanismes permettant une opérationnalisation des systèmes décrits.

La première partie présente les différents concepts relatifs aux systèmes multi-agents. Nous y montrons la diversité des concepts (agents, interaction et organisation) et la nécessité de leur formalisation à différents niveaux de description (locale, semi-globale et globale). Une description de système est considérée comme l'agrégation de ces niveaux de description.

La deuxième partie propose un modèle pour la formalisation de description de système en prenant en compte ces concepts, ainsi que les mécanismes permettant d'opérationnaliser cette description. Nous définissons ainsi un cadre de description de systèmes multi-agents permettant la définition de société d'agents hétérogènes et son opérationnalisation sous la forme d'un ensemble d'agents autonomes. Ce cadre de description met l'accent sur la modularité et la réutilisation des concepts décrits tout en proposant un modèle de réorganisation pour la gestion des interactions entre des agents autonomes. Ce modèle de réorganisation se fonde sur une adaptation locale des croyances de chaque agent. Cette adaptation utilise une notion de préférence associée à des schémas d'interaction. Les schémas expriment la réalisation des interactions ainsi que l'évaluation du bénéfice lié à leurs utilisations, ce qui permet éventuellement de modifier la préférence qui caractérise chacun de ces schémas.

La troisième partie débute par une présentation des expérimentations préliminaires que nous avons effectuées à partir de GTMAS sur le modèle de réorganisation. Ces expérimentations montrent ainsi les propriétés d'adaptation qu'offre notre modèle à des systèmes multi-agents. Nous présentons ensuite une plate-forme, basée sur notre approche descriptive et opérationnelle pour la construction de systèmes, JAVAMA. Nous illustrons l'utilisation de cette plate-forme sur deux exemples à partir desquels nous proposons une évaluation de la plate-forme par rapport à d'autres outils existant et envisageons diverses améliorations.

Mots clés :

Intelligence artificielle distribuée, système multi-agents, schéma d'interaction, organisation, adaptation, description