# Conception et mise en oeuvre de systèmes multi-agents ouverts et distribués

Laurent Vercouter

# THÈSE

présentée par

Laurent Vercouter

pour obtenir le grade de Docteur
de l'Université Jean Monnet et
de l'Ecole des Mines de Saint-Etienne
(Arrêté ministériel du 30 mars 1992)

Spécialité : informatique

# Conception et mise en œuvre de systèmes multi-agents ouverts et distribués

Soutenue le 20 décembre 2000

| | |
|---|---|
| Président : | M. Bertrand David |
| Rapporteurs : | M. Pierre Glize |
| | M. Jean-Pierre Müller |
| Examinateurs : | M. Philippe Beaune |
| | M. Colin de la Higuera |
| | Mme. Claudette Sayettat-Fau |

Thèse préparée au sein du Département **S**ystèmes **I**ndustriels **C**oopératifs

# THÈSE

présentée par

Laurent Vercouter

pour obtenir le grade de Docteur
de l'Université Jean Monnet et
de l'Ecole des Mines de Saint-Etienne
(Arrêté ministériel du 30 mars 1992)

Spécialité : informatique

# Conception et mise en œuvre de systèmes multi-agents ouverts et distribués

Soutenue le 20 décembre 2000

| | |
|---|---|
| Président : | M. Bertrand David |
| Rapporteurs : | M. Pierre Glize |
| | M. Jean-Pierre Müller |
| Examinateurs : | M. Philippe Beaune |
| | M. Colin de la Higuera |
| | Mme. Claudette Sayettat-Fau |

Thèse préparée au sein du Département Systèmes Industriels Coopératifs

**SCIDEM**
**Espace Fauriel**

# Remerciements

Je souhaite remercier Madame **Claudette Sayettat** pour la confiance qu'elle m'a accordée en acceptant de diriger mon travail. Ses conseils et son aide scientifique ont largement contribués à la réalisation de cette thèse.

Je remercie également Monsieur **Philippe Beaune** d'avoir encadré ma thèse. Son apport est à la fois scientifique par ses points de vue critiques sur mes travaux, et humain par ses nombreux encouragements.

Je remercie Messieurs **Pierre Glize** et **Jean-Pierre Müller** pour avoir accepté d'être rapporteurs de ma thèse. Leur examen attentif de mon mémoire et leurs critiques très constructives ont contribués à l'amélioration de ce document. Je remercie aussi Messieurs **Bertrand David** et **Colin de la Higuera** d'avoir accepté de faire partie de mon jury.

Je tiens à remercier :

Monsieur **Olivier Boissier** qui s'est montré très disponible face à mes nombreuses sollicitations. Ses remarques et critiques éclairées m'ont été particulièrement utiles tout au long de mon travail. J'associe également sa femme **Marie-Dominique**, ainsi que leurs filles, à ces remerciements, pour leur hospitalité et l'accueil très chaleureux qu'ils m'ont toujours accordés.

Tout ceux qui m'ont accompagné tout au long de ces trois ans, et plus particulièrement **Fernando**, **Franck**, **Hubert**, **Jean-Claude**, **Jean-Luc**, **Johann** et **Stéphane**. Merci aussi à **Nicolas** de m'avoir cité dans les remerciements de sa thèse (et pour beaucoup d'autres choses). Merci à l'équipe du Foyer du Personnel pour les indispensables moments de détente du Lundi soir.

Tous les membres des centres **SIMMO** et **SITE** de l'Ecole des Mines pour leur sympathique accueil et l'ambiance chaleureuse qu'ils m'ont réservés.

Les cristolliens **Benoît**, **Eric**, **Jean-Christophe**, **Wilfrid** et **Yves** ainsi

**SCIDEM**

Espace Fauriel

que **Adrien**, **Hélène**, **Jérôme** et **Stéphane** pour leur amitié et pour tout ce que nous partageons depuis notre plus jeune âge. Merci aussi à **Sonia** pour son soutien, plus que nécessaire, pendant la rédaction de ce mémoire.

Tout au long d'un cursus universitaire, il arrive que certains enseignants aient une influence prépondérante sur les choix que doit effectuer un étudiant. A ce titre, je souhaite remercier vivement Monsieur **Jacques Guignot** qui est à l'origine de ma passion pour la recherche.

Merci à **Fabienne** et à **Birgit** pour leurs corrections sur mes documents en anglais.

Mes **parents** et mes **frères** m'ont également apporté un soutien constant. Je les en remercie ainsi que pour tout ce qu'ils ont fait pour moi durant toutes ces années.

Enfin, je voudrais adresser mes plus vifs remerciements à **Franck Benayoun**. Il ne sait sûrement pas combien je lui suis redevable pour l'influence indirecte qu'il a eue sur mon travail pendant ces trois années de thèse. Je ne le remercierais jamais assez pour son amitié si précieuse.

## Résumé

Un Système Multi-Agents (SMA) ouvert est un système extensible et évolutif. Son extensibilité se traduit par la possibilité d'ajouter de nouveaux agents mais aussi de gérer leurs retraits. De plus, un SMA ouvert doit permettre l'évolution de ses agents. Nous nous sommes intéressés à la conception de SMA ouverts du point de vue de la représentation des autres maintenue par chaque agent du système. L'ouverture peut conduire la représentation qu'un agent a des autres, à devenir fausse ou incomplète. Dans les travaux existants, l'ouverture d'un SMA est gérée de manière centrale par une entité regroupant une représentation de chaque agent du système. Cette entité, qui selon les cas est un "broker", un faciliteur ou des pages jaunes, fournit sur demande la connaissance nécessaire à un agent pour raisonner sur les autres agents de son système. Outre le coût de l'accès à ces informations, cette approche présente l'inconvénient de dépendre du bon fonctionnement de cette entité critique et indispensable à l'ouverture et à la coopération.
La première étape de notre travail a été d'analyser dans quelle mesure on peut distribuer la tâche consistant à intégrer de nouveaux agents. Cette analyse nous a conduit à définir un modèle social partiel d'agent que nous appelons sa facette accueillante. Un agent accueillant est un agent ayant la faculté d'aider un nouvel agent à intégrer le système auquel il appartient. Une société composée d'agents accueillants est un SMA ouvert où l'intégration de nouveaux agents est assurée par une activité collective et coopérative faisant intervenir plusieurs agents accueillants. Le développement d'une approche distribuée de la conception de SMA ouverts nous a ensuite amené à généraliser le problème de l'ouverture. Nous avons repris notre définition des tâches associées à la gestion de l'ouverture d'un SMA pour proposer un modèle général de conception de SMA ouvert autorisant le choix d'une approche centralisée ou distribuée.

**Mots clés :** Systèmes multi-agents, Systèmes ouverts, Systèmes distribués, Représentation des autres, Stéréotypes d'agents logiciels

**SCIDEM**
Espace Fauriel

## Abstract

An open multi-agent system (MAS) is an extensible and evolutive system. Extensibility means the possibility of adding new agents but also of managing their withdrawals. Moreover, an open MAS must allow the evolution of the agents' abilities. We were interested in the design of open MAS from the perspective of the representation of others maintained by each agent of the system. In an open MAS, the representation that an agent has about others, may become false or incomplete.
In existing works, this problem is managed by a central entity gathering a representation of each agent of the system. This entity, which may be a broker, a facilitor or yellow pages, provides some knowledge about others each time an agent requests it. This approach has two disadvantages. First, access to knowledge about others requires some message exchanges, which may be costly. The second one is that the whole system is dependent on the correct functionning of the central entity which is a critical and essential component of an open MAS for co-operation between agents.
The first stage of our work was to analyze how the task consisting in integrating new agents can be distributed. This analysis led us to define a partial social model of an agent which we call its welcoming facet. A welcoming agent is an agent having the ability to help a new agent to integrate the system to which it belongs. In an open MAS made up of welcoming agents, the integration of new agents is ensured by a collective and co-operative activity involving several welcoming agents.
The development of a distributed approach to the design of open MAS led us to generalize this problem. With the help of our definition of welcoming tasks, we propose a general model to design open MAS such that a conceptor may choose either a centralized or a distributed approach.

**Keywords :** Multi-agent systems, Open systems, Distributed systems, Agent modeling, Stereotypes

# Table des matières

# Liste des figures

# Liste des tables

# Chapitre 1

# Introduction

La croissance de taille et d'hétérogénéité des systèmes informatiques, observées depuis plus d'une vingtaine d'années, a conduit la recherche en informatique à s'intéresser et à développer des systèmes composés de plusieurs entités parmi lesquelles sont réparties des tâches complexes à exécuter. Cette approche de conception de systèmes, appelés approche distribuée, s'est développée dans les différentes branches de l'informatique. Ainsi, à partir de l'Intelligence Artificielle (IA) est née l'Intelligence Artificielle Distribuée (IAD) qui a elle-même engendrée les Systèmes Multi-Agents (SMA).

Un système multi-agent est composé de plusieurs sous-systèmes autonomes appelés agents dont chacun a une activité et des informations propres. Le comportement général du SMA est alors lié à l'activité combinée de l'ensemble de ses agents et la réalisation d'une tâche peut impliquer plusieurs agents. Cette répartition nécessite que chaque agent puisse effectuer localement la tâche (ou sous-tâche) qui lui est assignée, mais aussi qu'il puisse se coordonner avec d'autres agents s'il faut gérer des dépendances entre les sous-tâches ou s'il a besoin de fonctions assurées par d'autres agents.

On identifie ces activités conjointes par le terme de *coopération* signifiant que plusieurs agents s'entraident dans la réalisation de leurs activités. Cela nécessite des capacités de raisonnement locales à chaque agent pour qu'il puisse déterminer de quelle aide il a besoin et quel agent peut la lui fournir. Ce raisonnement se fonde sur les informations dont dispose l'agent sur ses propres fonctions, sur son environnement et sur les autres agents.

Les informations d'un agent sur les autres agents du SMA est appelée sa *représentation des autres* et tient un rôle primordial dans les interactions et coopérations entre agents. La structure et le contenu d'une représentation des autres varient suivant les besoins d'un agent au niveau de ses interactions avec les autres. Ainsi, s'il cherche une coopération, un agent doit connaître les fonctions des autres pour s'adresser aux agents pertinents alors que s'il doit surveiller et éviter des conflits potentiels dûs à des interférences d'activités, il doit connaître leurs activités et états courants.

La représentation des autres n'est pas une connaissance statique et figée. Si elle décrit l'activité courante des autres, elle doit être fréquemment remise à jour pour être utilisable. Même dans le cas où elle représente les fonctions des autres, il faut pouvoir répercuter les modifications touchant les agents du SMA. Ces modifications peuvent être d'ordre locales (c'est-à-dire toucher les caractéristiques et fonctions propres d'un agent) ou globales (le partitionnement du système en agent est altéré soit en ajoutant ou en retirant des agents, soit en modifiant l'organisation des agents entre eux). Les SMA autorisant et gérant ces modifications sont dits *ouverts* et nécessitent plus que tout autre un modèle de représentation des autres précis ainsi que des mécanismes d'acquisition et de mise à jour de cette représentation des autres.

Ce mémoire présente nos travaux sur la **conception de systèmes multi-agents ouverts en proposant un modèle de représentation des autres adapté à l'ouverture du système** et facilitant son acquisition et sa mise à jour par les agents.

Pour atteindre cet objectif, nous commençons par une étude générale de la représentation des autres agents et de soi dans laquelle nous la positionnons par rapport aux diverses représentations contenues dans la connaissance d'un agent. Différentes approches de la représentation des autres, mettant l'accent sur les fonctions, les préférences ou les états des agents, sont présentées dans le chapitre 2.

Le chapitre 3 est consacré aux systèmes multi-agents ouverts. Nous y donnons notre point de vue sur l'ouverture d'un SMA et ce que cela implique comme propriétés du système et comme fonctionnalités qui doivent être accomplies. L'ouverture d'un SMA a aussi donné naissance à plusieurs approches que nous classons dans deux catégories: l'approche centralisée confiant l'ouverture à une entité et l'approche distribuée où tous les agents du SMA peuvent participer aux tâches liées à l'ouverture.

Après ces deux chapitres d'état de l'art, nous proposons une nouvelle approche distribuée de l'ouverture d'un SMA. Elle implique que chaque agent participe à l'ouverture du SMA et donc bénéficie de connaissances et de capacités de raisonnement appropriées à cette tâche. Notre proposition prend la forme d'un modèle partiel d'agent englobant ces connaissances et ce raisonnement que nous appelons la *facette accueillante* d'un agent. Dans le chapitre 4, nous définissons un modèle de représentation des autres centré sur les fonctions et compétences des autres agents. Ce modèle permet de sélectionner les informations pertinentes à connaître à propos des autres ainsi que celles nécessaires à l'agent pour accomplir sa part dans l'ouverture du SMA.

Le chapitre 5 présente les mécanismes de raisonnement associés à l'ouverture du SMA. Nous y montrons comment se passe l'intégration d'un agent dans le SMA ainsi que les mécanismes attachés au retrait d'un agent ou à la propagation d'une modification interne.

Enfin, les deux derniers chapitres traitent des propriétés et des applications de notre proposition de SMA ouvert. Le chapitre 6 donne notre point de vue critique sur les différentes approches de l'ouverture en les comparant en termes de coût de communication, de raisonnement et de robustesse du SMA. Nous y proposons également des variantes de notre approche pour passer à une gestion centralisée de l'ouverture ou pour renforcer la robustesse ou la sécurité dans le SMA.

Dans le chapitre 7, nous utilisons nos agents accueillants pour "ouvrir" un système distribué en encapsulant ses entités dans des agents. Un système d'information distribué est pris comme exemple pour illustrer cet apport.

# Première partie

# Etat de l'art

# Chapitre 2

# La représentation d'autrui et de soi

Sans se limiter au domaine des systèmes distribués, un logiciel est toujours conçu de manière à interagir avec d'autres éléments qui lui sont externes. Une fonction fréquente d'un logiciel étant le traitement d'informations, il doit se procurer ces informations dans une source indépendante du programme de traitement et répercuter les résultats obtenus dans d'autres entités externes (telles des interfaces utilisateurs, d'autres logiciels, des périphériques de sortie, ...). A titre d'exemple, on peut citer les systèmes d'exploitation qui doivent interagir avec la machine sur laquelle ils se situent, de même que les logiciels applicatifs interagissent avec leur système d'exploitation. On a coutume d'appeler ces interactions, les entrées/sorties d'un logiciel. Celles-ci présupposent donc l'existence d'objets extérieurs au logiciel nécessaires à son utilité voire à son bon fonctionnement.

Pour un agent logiciel, ces objets extérieurs prennent des formes diverses et peuvent être des objets physiques ou informatiques, des utilisateurs humains, et, dans le cas qui nous intéresse, d'autres agents logiciels. Le bon fonctionnement d'un agent étant lié à ses actions sur ces objets ou à sa perception de ceux-ci, il doit savoir comment les utiliser ou interpréter les signaux perçus et donc disposer d'une certaine connaissance de ces objets. Cette connaissance peut être vue comme un ensemble de *représentations* dont chacune établit un modèle plus ou moins précis du substrat associé.

Dans ce chapitre, nous décrivons quelques types d'entités à représenter, sans chercher à être exhaustifs. Puis nous présentons un état de l'art de la représentation des autres agents logiciels.

## 2.1 Différents substrats de représentation

La connaissance d'un agent est composée d'un ensemble de représentations portant sur les éléments de son "monde" et sur certaines de ses

propres parties. Une représentation est toujours associée à un objet ou un concept, proposant une "image" partielle du substrat, celle-ci décrivant un sous-ensemble de ses caractéristiques perçues. La notion de représentation n'est pas nouvelle et était déjà évoquée comme un élément de la connaissance humaine : "j'ai souvent remarqué, en beaucoup d'exemples, qu'il y avait une grande différence entre l'objet et son idée" [Descartes, 1647]. Ainsi, en plus d'être incomplète, une représentation peut se révéler incorrecte en ne décrivant pas les propriétés de son objet de manière exacte. Dans le domaine de l'Intelligence Artificielle, D. Kayser [Kayser, 1997] définit une représentation comme une approximation, justifiant ce point par le fait qu'une représentation complète d'une entité (possédant toutes ses propriétés) ne peut être que l'entité elle-même (ce point de vue est également défendu par R. Brooks [Brooks, 1991] pour lequel la meilleure représentation du monde est le monde lui-même).

D. Kayser nous dit également qu'une représentation est une structure de symboles qui s'interprète sur un modèle du monde. Pour doter les agents de représentations, il faut donc dans un premier temps cerner les différentes catégories d'entités de leur monde pour définir des modèles appropriés.

L'approche *Voyelles* [Demazeau, 1997] propose de distinguer quatre axes dans un système multi-agent : les Agents, l'Environnement, les Interactions et l'Organisation (AEIO). Un agent du système, évoluant dans ce système, construit et interprète des représentations du contenu de ces axes. Certains agents, amenés à interagir avec des utilisateurs humains, maintiennent en plus des représentations d'utilisateurs. Nous évoquons cette connaissance supplémentaire, non pas pour ajouter une cinquième voyelle à l'approche AEIO car elle catégorise un SMA et l'utilisateur est extérieur au système, mais parce que les modèles utilisés pour représenter un utilisateur nous ont intéressés dans le cadre de la représentation des autres agents.

Nous décrivons ici la représentation de l'Organisation, de l'Interaction, de l'Environnement, de l'Utilisateur et des Agents en citant quelques modèles associés avant de détailler dans les prochaines sections différentes approches de la représentation des autres.

### 2.1.1 Organisation, Interaction et Environnement

L'**Organisation** structure une société d'agents par des relations entre agents ou par la décomposition d'une tâche globale du système qu'elle assigne aux agents. Les modèles d'Organisation sont très divers : certains s'appuient sur des relations d'autorité ([Hannoun et al., 1998]), d'autres sur des rôles définissant des comportements à adopter ([Cavedon and Sonenberg, 1998], [Veloso and Stone, 1998], [Gutknecht and Ferber, 1999]), des règles sociales à respecter ([Adam et al., 1999]) ou la formation d'équipes structurant les coopérations d'agents ([Tidhar et al., 1998]).

Du côté d'un agent, l'Organisation est représentée d'après le modèle

utilisé et souvent comme un sous-ensemble de relations, de rôles, de règles ou d'équipes dans lesquels l'agent est impliqué. Dans les méthodologies de conception de systèmes multi-agents orientées-organisation, on commence même par décrire globalement l'organisation souhaitée avant de découper le modèle global en plusieurs parties dont chacune est assignée à un agent ([Collinot et al., 1996], [Gutknecht and Ferber, 1999]).

La forme la plus courante d'**interaction** entre agents est la communication par messages. Pour communiquer les agents utilisent un langage d'interaction spécifiant la sémantique des différents messages disponibles. La théorie des actes de langage [Austin, 1962] [Searle, 1969] distingue les effets *illocutoires* (conditions requises pour l'émission d'un message) des effets *perlocutoires* (conséquences réelles du message) d'un acte de langage. Un message est émis quand sa portée illocutoire est vérifiée, celle-ci faisant souvent intervenir la représentation de l'agent destinataire du message (par exemple, pour demander à un agent d'effectuer une action, il faut croire que celui-ci peut faire cette action). L'interprétation faite du message par l'interlocuteur dépend de sa propre représentation de cet acte et représente l'effet perlocutoire. Dès lors que des agents utilisent un certain langage d'interaction, leur représentation de l'interaction consiste en un ensemble de fiches descriptives spécifiant la sémantique des actes de langages (voir par exemple [Labrou and Finin, 1994], [Carron et al., 1999]).

Une communication entre agents demande souvent une séquence de plusieurs messages. On peut alors définir des structures ordonnançant et contraignant un ensemble d'actes de langages. Ces structures, appelées "protocoles d'interaction" (cf. [Barbuceanu and Fox, 1995], [FIPA, 1997]), impliquent plusieurs agents où chacun peut être invité à émettre un message parmi ceux proposés suivant l'état dans lequel se trouve le protocole. Cette structuration des interactions se rapproche de la notion d'organisation. Ainsi, à l'instar de la représentation de l'organisation, un protocole global existe (explicitement défini ou non) et il est réparti dans la connaissance des agents d'après le rôle qu'ils sont susceptibles d'y tenir sous la forme de représentations des protocoles d'interaction. Chaque agent n'a besoin de connaître qu'une partie d'un protocole, celle qui le concerne.

L'**Environnement** est un axe particulier du SMA. En effet, alors que les représentations de l'organisation et des interactions désignent des objets abstraits, les substrats des représentations de l'environnement sont des objets concrets et réels ayant une existence propre indépendamment des agents. [Boissier, 2000] définit "l'environnement comme le medium commun aux agents du système, que l'agent peut contrôler totalement, ou partiellement, au travers de ses actions, à partir d'une perception locale, incomplète, voire incorrecte". Le contrôle partiel, ainsi que l'incomplétude ou l'erreur dans la perception sont les caractéristiques d'une représentation. Que l'environnement soit physique ou virtuel (ex. des ressources informatiques), un agent se représente une partie de l'environnement soit parce qu'elle lui a été assignée

ou qu'il la "possède" ([Franc and Sanders, 1998], [Vercouter, 1997]) soit par son état dynamique vis-à-vis de l'environnement (par exemple une position spatiale dans un environnement situé) ([Bakam, 1998], [Ramat et al., 1998]).

Même s'ils sont parfois considérés comme appartenant à l'Environnement des agents, les utilisateurs font l'objet de représentations particulières que nous décrivons maintenant.

### 2.1.2 Les utilisateurs

Certains agents doivent raisonner sur un utilisateur, prendre en compte ses actions ou comprendre les commandes qu'il émet. Pour cela, ils maintiennent une représentation des utilisateurs regroupant les connaissances et croyances nécessaires à leur fonction. Ces représentations sont souvent appelées des *profils d'utilisateurs.*

Nous nous sommes intéressés aux modèles existants de profil utilisateur car ce domaine de recherche présente des similarités intéressantes avec notre problématique de représentation des autres agents. En effet, contrairement aux représentations de l'Organisation, de l'Environnement et des Interactions dont les substrats sont des concepts abstraits ou des objets inanimés[1], les sujets désignés par une représentation d'un utilisateur ou d'un agent sont actifs et autonomes du point de vue de l'agent se les représentant. C'est-à-dire que le sujet peut agir sans qu'un événement particulier provenant de l'agent soit la cause directe de cette action.

La grande majorité des cas d'interaction entre un agent et un utilisateur considère que l'agent est au service de l'utilisateur. Un tel agent doit agir en fonction des demandes de son utilisateur ou en fonction de l'objectif qu'il lui attribue. L'interprétation d'une commande ou l'attribution de buts sont les informations maintenues dans un profil utilisateur.

A titre d'exemple, on peut citer le système Letizia [Lieberman, 1995] fournissant à un utilisateur un agent l'assistant pendant une navigation sur Internet. En observant le comportement de navigation de l'utilisateur *par dessus son épaule*, l'agent Letizia construit un profil lui correspondant. Cette construction nécessite que les pages parcourues soient étiquetées par des mots-clés. Suivant l'action de l'utilisateur, Letizia accorde un degré d'importance aux mots-clés de cette page (par exemple, la sauvegarde de l'adresse d'une page est plus significative que sa simple lecture, de même que le rapport entre le temps passé sur une page et la taille de cette page donne une indication de l'intérêt). Le profil utilisateur est ici constitué d'un ensemble pondéré de mots-clé dont Letizia va se servir pendant les navigations futures pour recommander à l'utilisateur certains liens à suivre.

[1] Les représentations de l'Environnement constituent néanmoins un cas particulier quand il s'agit de modéliser certains objets dotés d'une dynamique propre indépendant des agents

Même si c'est sa forme la plus courante, le contenu d'un profil utilisateur n'est pas systématiquement un ensemble de mots-clés. Dans un système proposant des agents d'interface triant automatiquement les messages électroniques d'un utilisateur, les agents peuvent effectuer plusieurs actions (effacer le message, le ranger dans un répertoire approprié, l'afficher en priorité, ...) et le profil utilisateur doit préciser d'après un contexte donné quelle est l'action appropriée. Dans le système de [Lashkari et al., 1998], les agents utilisent un raisonnement basé sur la mémoire pour construire un profil de leur utilisateur sous la forme de couples $< \mathcal{M}, \mathcal{A} >$ où $\mathcal{M}$ définit des caractéristiques de messages (émetteur, récepteur, date, priorité, ...) et $\mathcal{A}$ une action associée à entreprendre, dès qu'un message présentant les caractéristiques $\mathcal{M}$ est reçu.

Les profils sont une manière de représenter individuellemnt chaque utilisateur. Il est néanmoins possible de représenter des groupes d'utilisateurs sous la forme de stéréotype. Nous décrivons ces techniques en section 2.2.

### 2.1.3 Les agents logiciels

La tentation est grande de rapprocher la représentation d'un utilisateur et la représentation des autres agents à cause de la nature "agissante" de leurs sujets et de l'anthropomorphisme fréquemment utilisé dans les systèmes multi-agents. Néanmoins, ce serait une erreur d'appliquer aveuglément un modèle de profil utilisateur à la représentation d'un agent (et vice-versa) car il existe des différences importantes entre ces représentations:

- Il semble évident de dire qu'un agent suit un fonctionnement beaucoup plus simple qu'un utilisateur. Il est imaginable qu'une représentation ait le même modèle que l'agent représenté alors que la question ne se pose même pas pour un utilisateur dès lors qu'il n'existe pas de modèles d'humains. Alors que des techniques d'apprentissage permettent d'acquérir un profil approximatif d'un utilisateur, l'apprentissage d'une représentation d'un autre agent dépend du contexte coopératif du système et/ou de son homogénéité permettant l'utilisation de techniques plus simples avec des résultats plus précis.

- Un agent gérant un profil utilisateur est la plupart du temps au service de cet utilisateur. C'est-à-dire qu'il va tenter de répondre favorablement à toutes ses commandes et requêtes sans s'interroger sur ses motivations. Dans un système multi-agent, cette servitude n'est qu'un cas particulier d'interactions entre agents et la décision de répondre à une requête d'un autre agent prend parfois en considération des relations d'autorité entre agents ou le but motivant cette requête.

- Les interactions entre agents sont plus fréquentes. L'assignation d'un agent à un utilisateur a généralement pour objectif l'automatisation de

certaines des tâches qu'il effectuait jusque-là manuellement. Il est donc souhaitable de minimiser les sollicitations émanant de l'agent vers son utilisateur pour ne pas allourdir le système et ennuyer l'utilisateur avec de fréquents messages ou questions. Si une surcharge d'occupation peut amener les agents à ne pas répondre aux requêtes de leurs pairs, ils ne sont, fort heureusement, pas encore capables d'éprouver de l'ennui ou de l'agacement!

Comme nous l'avons évoqué précédemment, une représentation est exprimée en fonction d'un modèle qu'un agent utilise pour implanter et interpréter cette représentation. Le modèle choisi est alors dépendant du raisonnement pratiqué sur la représentation des autres et doit ressortir les propriétés adéquates. De manière générale, on peut dire qu'un agent $i$ doit se représenter un autre agent $j$ par ses caractéristiques susceptibles d'altérer l'état interne de $i$. Cet état interne peut être des croyances sur son environnement, sur les autres agents, une satisfaction de buts, une récompense, ...

Le raisonnement sur les autres doit prendre en compte plusieurs agents simultanément. Ceux-ci doivent être aussi représentés de manière à prendre en compte leurs *interactions* car deux agents agissant en même temps peuvent avoir une influence différente sur les états de l'agent $i$ que si l'on considérait séparément chaque action.

Considérons par exemple trois agents $i$, $j$ et $k$ partageant un accès à une ressource $r$ utilisable par un seul agent à la fois. L'agent $i$ compte utiliser $r$ de 15h à 15h30, $j$ de 15h15 à 16h et $k$ de 16h à 17h. Il y a un conflit entre les prévisions d'utilisation de $i$ et de $j$ qu'ils peuvent détecter et résoudre (par une négociation, ou par une autre méthode interne) en décalant la période à laquelle $j$ utilise $r$ à l'intervalle 15h30 $\rightarrow$ 16h15. Cette solution créé néanmoins un nouveau conflit entre $j$ et $k$ qui doit être détecté avant de commencer sa résolution. Pour cela il faut que l'agent $k$ ait connaissance du nouvel horaire de $j$ et ait pris en compte les interactions entre $i$ et $j$, soit pour simuler en interne leurs raisonnements et prédire les modifications d'horaires, soit pour savoir qu'un des horaires est susceptible d'avoir été modifié et observer le nouveau ou le demander si la communication est possible.

Dans cet exemple, on peut remarquer que les agents doivent être représentés suivant un même modèle (des périodes d'utilisation d'une ressource). En effet, pour interpréter les interactions entre agents, il est souhaitable d'avoir un modèle commun à chaque représentation d'un autre. Ce modèle décrivant les influences et relations entre agents, un agent peut raisonner sur un ensemble d'agents plutôt que sur chaque agent séparément. Dans certains cas, un agent doit raisonner *globalement* sur un ensemble d'agents auquel il appartient. Il maintient alors une représentation de lui-même suivant le même modèle que celui utilisé pour représenter ses accointances, de manière à ce qu'il puisse raisonner sur ses interactions avec les autres agents [Giroux, 1995]. Nous appelons une représentation des autres incluant une

représentation de soi-même, une représentation *réflexive*.

Dans les sections suivantes, nous revenons d'abord sur la représentation des utilisateurs pour évoquer le cas particulier de représentations groupées avant de présenter différentes approches de la représentation des autres, en débutant par les modèles de représentation des préférences pour la prédiction du comportement des autres. Nous nous intéressons ensuite aux représentation des états mentaux des autres (modèle qui peut également servir à la prédiction). Enfin nous décrivons des systèmes centrés sur les capacités des agents pour évaluer un potentiel de coopération ou d'interférence.

## 2.2 Représentation groupée d'utilisateurs

La connaissance représentée par un profil utilisateur est composée d'informations dites *sur le contenu*. C'est-à-dire qu'elle porte sur les caractéristiques propres d'un utilisateur telles que ses centres d'intérêt, préférences ou habitudes. Dans un contexte multi-utilisateurs, la connaissance disponible est plus riche que la simple addition des profils de chaque utilisateur car un agent peut y ajouter des informations *sociales* [Basu et al., 1998]. Celles-ci proviennent de la mise en relation des différents profils utilisateurs et de la déduction de points communs ou de différences.

Un profil utilisateur est alors enrichi par ces informations sociales obtenues par comparaison avec les profils des autres utilisateurs. Un exemple de cette approche est le filtrage collaboratif de documents ([Resnick et al., 1994], [Delgado and Ishii, 1999]) : plusieurs utilisateurs ont accès à un ensemble de documents et notent chaque document d'après l'intérêt qu'ils leur accordent. Une note unique est attribuée à chaque document, calculée en fonction des évaluations de chacun. Ce sont ensuite les utilisateurs qui sont notés par le logiciel d'après une distance entre leurs propres évaluations de documents et l'évaluation globale retenue. A chaque profil utilisateur est assignée une valeur exprimant la confiance accordée à la notation de l'utilisateur correspondant. Par exemple un utilisateur donnant des notes éloignées de la note générale d'un document sera peu pris en compte par la suite à l'inverse d'un utilisateur proche de la note générale. Dans le cadre de cette application, les notes des documents sont des informations sur le contenu et les notes des utilisateurs des informations sociales.

Le filtrage collaboratif met en œuvre un système de recommandation de documents, affiné par la prise en compte d'informations sociales. La recommandation peut être vue comme une fonction $f(X) \rightarrow V$ où $X$ est un document et $V$ une estimation de l'intérêt du document si l'on considère qu'un document a une valeur d'intérêt générale quelque soit l'utilisateur. Elle peut être néanmoins complexifiée par l'introduction de classes d'utilisateurs sous la forme $f(U, X) \rightarrow V$ [Basu et al., 1998] où $U$ est une classe d'utilisateurs. La notion de classes d'utilisateur est désignée par le terme

*stéréotype*. Il existe deux approches différentes de formation de stéréotypes décrites par la figure 2.1.

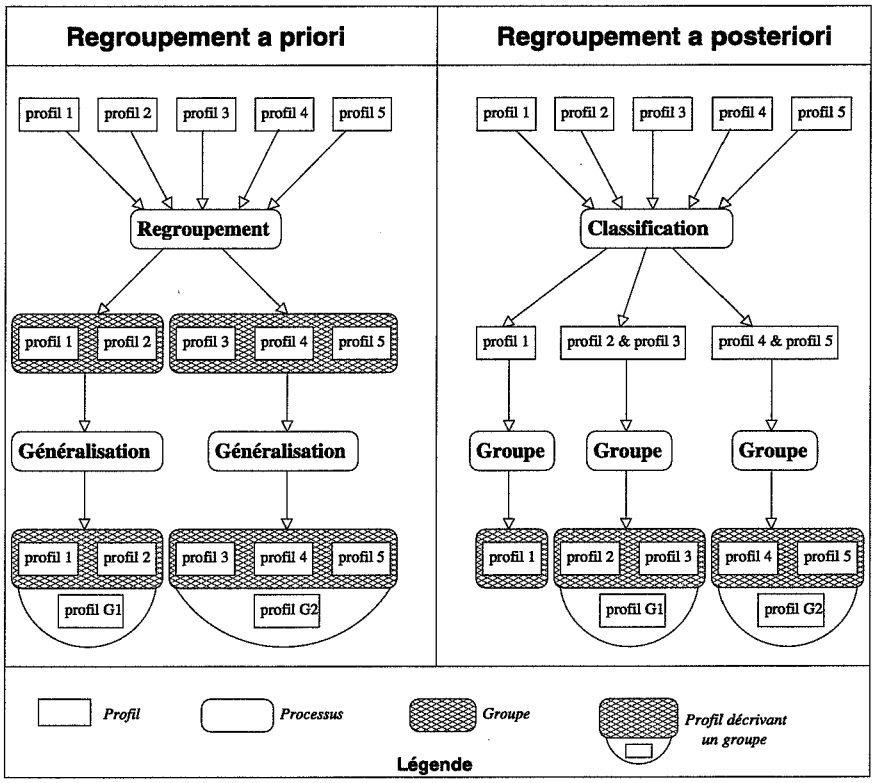


**Figure 2.1** : Formations de stéréotypes

### 2.2.1 Regroupement a priori

Le système Ricochet [Bothorel, 1998] propose un système de recommandation en regroupant les utilisateurs dans des communautés d'intérêts communs. A chaque utilisateur est assigné un agent disposant d'un profil de son utilisateur. Celui-ci est chargé de sélectionner quelles communautés il va intégrer en évaluant la distance entre le profil de son utilisateur et celui de la communauté. Ce profil de communauté est l'équivalent d'un stéréotype car c'est un profil caractérisant l'ensemble des utilisateurs y appartenant.

La formation de stéréotypes est dans ce cas un processus de généralisation prenant en entrée l'ensemble des profils utilisateurs d'une communauté pour en ressortir un profil général décrivant la communauté. Ces profils communautaires évoluent à chaque ajout ou retrait d'un profil utilisateur pour représenter au mieux leurs membres. La caractéristique de cette approche est que des groupes sont d'abord imposés et que les profils communautaires

sont une conséquence de ce regroupement.

### 2.2.2 Regroupement a posteriori

Une plus grande importance est accordée ici aux stéréotypes. Les profils utilisateurs ne sont pas assignés initialement à un groupe. Ils sont pris comme entrée d'un processus de classification qui produit un ensemble de stéréotypes d'utilisateurs. Cette classification est souvent incrémentale, c'est-à-dire qu'à chaque insertion d'un nouveau profil, les stéréotypes sont mis à jour. La structure résultant de cette classification peut prendre différentes formes. Si l'on souhaite classer chaque utilisateur dans un stéréotype, il est préferrable de disposer d'un arbre de décision. Par contre, si chaque utilisateur peut se retrouver dans plusieurs stéréotypes, on peut garder un ensemble inordonné ou, si plusieurs niveaux de généralité sont requis, une hiérarchie.

Des groupes d'utilisateurs sont ensuite formés automatiquement pour correspondre à chaque stéréotype. Ici, les stéréotypes sont la cause du regroupement. On peut trouver une illustration de cette approche dans les travaux de [Rich, 1979] proposant un système de classification de profils utilisateurs dans des stéréotypes prédéfinis. Le système recommande à l'utilisateur la lecture de certaines nouvelles associées à son stéréotype et, suivant les réponses de l'utilisateur (son évaluation de la recommandation), celui-ci est classé dans des stéréotypes de plus en plus précis. On trouve une approche similaire utilisant une construction incrémentale de stéréotype dans [Bendou, 1999].

### 2.2.3 Classement et recommandation

Les stéréotypes formés, selon l'une ou l'autre approche, présentent une double utilité. La première est le classement de nouveaux profils dans les stéréotypes les plus proches. Les profils des stéréotypes sont considérés comme des profils par défaut pour leurs utilisateurs. La nature humaine du sujet amplifiant le caractère approximatif de la représentation, il est très probable que pour un profil appris par un agent, l'utilisateur n'ait pas exprimé tous ses intérêts et pour un profil donné par l'utilisateur, qu'il ait oublié de mentionner certains de ses intérêts. Par exemple si un utilisateur se définit dans un profil par les caractéristiques suivantes: "profession: doctorant", "domaine: informatique", "mots-clés: systèmes multi-agents, systèmes ouverts, apprentissage automatique", il peut être classé dans un stéréotype regroupant des profils d'autres doctorants en informatique présentant certaines de ces caractéristiques et d'autres différentes. Si dans ce stéréotype est présent un mot-clé supplémentaire comme "enseignement de l'informatique", il sera attribué par défaut au profil de cet utilisateur et pourra combler un de ses oublis. Bien sûr, si par la suite il rejette cet ajout (explicitement ou par un comportement contraire), une exception sera précisée dans son profil, son

profil sera retiré du stéréotype ou le stéréotype sera modifié.

Dans ce domaine de connaissances partielles sur les utilisateurs, le second avantage des stéréotypes est de contenir une information, parfois inexprimable, favorisant les interactions entre les utilisateurs d'un même stéréotype. L'idée est que si des utilisateurs partagent certaines caractéristiques, il est très probable qu'ils en partagent d'autres, inconnues du système et absentes de leurs profils. On revient alors au principe du filtrage collaboratif où des utilisateurs d'une même communauté attribuent une valeur d'intérêt à certains documents, utilisée par la suite pour recommander les documents bien notés aux autres utilisateurs de la communauté. Le système ne sait pas forcément pourquoi ils sont jugés intéressants, il sait juste que cette communauté s'intéresse à ces documents ou types de documents et que tout nouveau membre est susceptible de s'y intéresser.

Nous nous sommes intéressés aux stéréotypes d'utilisateur pour cette possibilité de recommandation. Nous proposons dans la section 4.3 du chapitre 4 de l'adapter à des agents logiciels dans leur représentation d'autres agents. Cette représentation n'est cependant pas basée sur un profil utilisateur mais sur un modèle d'agent, domaine dont nous parcourons maintenant l'existant.

## 2.3 Représentation des préférences des agents

Les agents connaissent différents états possibles du monde et les ordonnent suivant un degré de préférence accordé à chacun d'entre eux. Un agent qui ne bénéficierait pas de représentations des autres choisirait une ou plusieurs actions l'amenant à l'état atteignable le plus satisfaisant. Cependant, la transition d'un état vers un autre prend en compte les actions de plusieurs agents. Ainsi, un agent doit prédire les actions de ses accointances pour choisir sa propre action en fonction des états atteints par les combinaisons d'action correspondantes. Cette prédiction nécessite une représentation des préférences des autres pour savoir quel état ils veulent atteindre.

### 2.3.1 En théorie des jeux

La théorie des jeux est par excellence un domaine compétitif où chaque participant agit selon son propre intérêt pouvant être incompatible avec les intérêts locaux de ses partenaires. Dans ce contexte, les agents ont un ensemble d'actions parmi lesquelles ils doivent en choisir une à effectuer. Quand chaque agent impliqué dans un jeu a choisi une action à accomplir, les agents en retirent un bénéfice (ou subissent un coût) propre à chacun et calculé en fonction de la combinaison d'actions de la société. Un agent suit une *stratégie* gérant sa prise de décision d'action à entreprendre en espérant maximiser ses bénéfices (ou minimiser ses coûts). Par exemple, il peut suivre une stratégie fixe (à une situation donnée correspond une action

à accomplir) ou il peut apprendre sa stratégie en fonction des retours perçus précédemment.

Nous allons nous intéresser ici à une autre approche impliquant la représentation des autres. En effet, le retour perçu par un agent est conséquence de son action mais aussi de celles des autres agents. Si un agent possède un modèle pertinent des autres, il peut alors anticiper leurs actions et, en connaissant ou en ayant appris la fonction calculant les retours qu'il perçoît, il peut déduire laquelle de ses actions lui procurera le meilleur retour.

On peut citer comme exemple le *dilemme du prisonnier* [Axelrod, 1984] qui est un jeu impliquant deux joueurs. Chacun a le choix entre deux actions à effectuer: coopérer (C) avec son partenaire ou le trahir (T). Le tableau 2.1 présente la matrice de gains associée à ce jeu.

**Table 2.1 :** Matrice de gains du dilemme des prisonniers

| | | Joueur 2 C | Joueur 2 T |
|---|---|---|---|
| Joueur 1 | C | 3 / 3 | 5 / 0 |
| | T | 0 / 5 | 1 / 1 |

Si un joueur raisonne sur son seul intérêt personnel, il ne fera que trahir car, quelque soit l'action choisie par son partenaire, le bénéfice est supérieur que s'il avait coopéré. Le problème est que si les deux joueurs se trahissent, le gain est plus faible que s'ils s'étaient mis d'accord pour coopérer. Le *dilemme des prisonniers itéré* répète ce jeu plusieurs fois permettant d'évaluer sur la durée l'efficacité d'une stratégie. Cette répétition créé un historique de parties à partir duquel un agent peut apprendre la stratégie de l'adversaire et jouer de la manière qui lui est la plus profitable.

Pour [Carmel and Markovitch, 1998], les croyances d'un agent se divisent en trois parties dans ce contexte:

- Les actions que peut effectuer l'adversaire;
- La stratégie de l'adversaire qui est composée de données *privées*, c'est-à-dire qu'elle n'est pas directement accessible;
- L'historique des parties qui est une connaissance commune aux deux agents.

Pour généraliser le cas du dilemme des prisonniers itéré, on peut considérer les deux premiers points comme une représentation d'un autre. Les actions que peut effectuer un agent représentent une croyance statique car

elles sont initialement connues et invariantes. La partie la plus intéressante, du point de vue de la théorie des jeux, est la stratégie de l'adversaire car, étant inaccessible, il faut trouver un moyen de l'apprendre avec comme donnée d'apprentissage l'historique des parties déjà jouées. Ainsi, quelque soit la véritable forme de cette stratégie, l'agent en construit une approximation dans sa représentation de l'autre sous la forme de règles *situation* $\rightarrow$ *action* où une situation est un ou plusieurs états précédents (un état est une combinaison où chacun a joué un coup). Si des règles concurrentes (par exemple deux règles référençant une même situation mais ayant chacune deux actions différentes) existent, des probabilités y sont assignées pour les prendre en compte à des degrés différents.

### 2.3.2 The Recursive Modeling Method (RMM)

Dans un cadre proche de la théorie des jeux, la *Recursive Modeling Method* [Gmytrasiewicz and Durfee, 1995] (en français, Méthode de Modélisation Récursive) définit un modèle d'agent doté d'une représentation récursive des autres (un agent se représente un agent qui se représente un agent...) et d'un raisonnement utilitaire. Les agents sont rationnels (au sens des auteurs), c'est-à-dire qu'ils agissent en fonction de leur propre intérêt. Dans le modèle RMM, cet intérêt est exprimé par une fonction d'utilité locale à chaque agent dont les paramètres sont les actions entreprises par tous les agents de la société. Grâce à cette fonction, un agent décide quelle action il doit effectuer pour tirer le plus grand bénéfice de la combinaison des actions des agents. De plus, un agent dispose d'un modèle homogène de représentation des autres par lequel il modélise chaque accointance ainsi que lui-même. Etant donné que l'utilité d'un agent dépend de ses actions mais aussi des actions des autres, le processus de prise de décision d'action des autres agents doit également être représenté pour anticiper leur comportement. Ce processus faisant intervenir l'utilité de l'agent représenté ainsi que son anticipation des actions des autres, le modèle est récursif et la représentation d'un agent contient aussi sa représentation des autres, et ainsi de suite... Enfin, le raisonnement utilitaire des agents les place potentiellement dans un univers compétitif (chaque agent cherche à maximiser son utilité au dépend de l'utilité des autres agents si nécessaire) où les croyances portant sur les autres peuvent être incertaines, car souvent issues d'un apprentissage à l'insu de l'agent sur lequel portent ces croyances. RMM permet plusieurs représentations différentes d'un même agent en assignant à chacune d'entre elles une probabilité.

Formellement, un agent $R_i$ est défini par une structure de modèle récursif :

$$RMS_{R_i} = (P_{R_i}, RM_{R_i}) \tag{2.1}$$

$P_{R_i}$ est la matrice de gains de l'agent $R_i$ définie par un triplet $(R, A, U)$

où $R$ est un ensemble d'agents $(R_1, \ldots, R_n)$, $A$ est un ensemble d'ensembles d'actions $(A_1, \ldots, A_n)$ tel qu'un ensemble $A_j = (a_1^j, \ldots, a_k^j, \ldots)$ représente les actions que peut effectuer l'agent $R_j$, et $U$ est la fonction d'utilité assignant une valeur (l'utilité) à une combinaison d'actions de chaque agent : $U : A_1 \times A_2 \times \ldots \times A_n \longrightarrow \Re$ où $\Re$ est l'ensemble des réels.

$RM_{R_i}$ est un modèle récursif de l'ensemble des accointances de l'agent $R_i$. Il est composé d'un ensemble de modèles alternatifs :

$$RM_{R_i} = ((p_1^{R_i}, M_{\{-R_i\}}^{(R_i,1)}) .. (p_\alpha^{R_i}, M_{\{-R_i\}}^{(R_i,\alpha)}) .. (p_m^{R_i}, M_{\{-R_i\}}^{(R_i,m)})) \qquad (2.2)$$

auxquels sont assignés des probabilités dont la somme est égale à 1. Chaque modèle $M_{\{-R_i\}}^{(R_i,\alpha)}$ avec une probabilité d'existence $p_\alpha^{R_i}$ regroupe les modèles de chaque accointance :

$$M_{\{-R_i\}}^{(R_i,\alpha)} = (M_{R_1}^{(R_i,\alpha)} .. M_{R_{i-1}}^{(R_i,\alpha)}, M_{R_{i+1}}^{(R_i,\alpha)} .. M_{R_n}^{(R_i,\alpha)}) \qquad (2.3)$$

Une accointance $R_j$ y est représentée par un modèle $M_{R_j}^{(R_i,\alpha)}$ qui peut prendre trois formes différentes :

- Le modèle intentionel $M_{R_j}^{(R_i,\alpha)} = IM_{R_j}^{(R_i,\alpha)}$. Il représente l'agent $R_j$ comme un agent rationnel par le biais d'une structure de modèle récursif ($IM_{R_j}^{(R_i,\alpha)} = RMS_{R_j}^{(R_i,\alpha)}$) lui attribuant une matrice de gain et un modèle récursif de ses accointances.

- Le modèle sub-intentionnel $M_{R_j}^{(R_i,\alpha)} = SM_{R_j}^{(R_i,\alpha)}$ est une distribution de probabilités sur l'ensemble des actions de $R_j$.

- Le modèle sans information $M_{R_j}^{(R_i,\alpha)} = NM_{R_j}^{(R_i,\alpha)}$ dans le cas où $R_i$ ne connaît rien des croyances de $R_j$. $R_i$ considère alors que la décision d'agir de $R_j$ est aléatoire et assigne une distribution équiprobable à l'ensemble des actions $A_j$.

Pour déterminer quelle action il doit effectuer, un agent $R_i$ doit assigner une valeur, estimant l'utilité probable, à chacune de ses actions. Pour une de ses actions $a_m^i$, il calcule la probabilité de chaque combinaison d'actions que peut effectuer l'ensemble de ses accointances (le produit des probabilités individuelles de chaque action) et multiplie chaque probabilité de combinaison par l'utilité espérée de la combinaison correspondante (en y incluant sa propre action $a_m^i$) notée $u^{R_i}_{a_k^1 \ldots a_m^i \ldots a_p^n}$. La somme de ces valeurs donne une estimation de l'utilité que l'agent $R_i$ peut attendre s'il effectue l'action $a_m^i$. Si $R_i$ dispose de plusieurs modèles alternatifs (équation 2.2), il faut estimer l'utilité dans chacun de ces modèles et faire leur somme pondérée par les probabilités correspondantes pour avoir l'utilité espérée de l'action.

Les probabilités selon lesquelles un agent effectue telle ou telle action ne sont pas directement disponibles s'il est représenté par un modèle intentionnel. Dans ce cas, l'agent $R_i$ propage l'information de manière ascendante en partant des feuilles de l'arbre (les modèles sub-intentionnels ou sans information) et remonte les divers modèles intentionnels en calculant les probabilités d'actions des agents concernés en considérant que ceux-ci cherchent aussi à maximiser leur utilité.

Plusieurs travaux ont enrichi ce raisonnement des agents. On peut citer l'introduction du raisonnement limité [Vidal and Durfee, 1995] restreignant le parcours de l'arbre aux branches critiques d'après des contraintes de temps ou de gains, ainsi que la justification de la communication et de la coopération [Gmytrasiewicz, 1995] comme moyen de réduction de l'incertitude sur les autres.

## 2.4 Représentation des états mentaux

Un des héritages que l'Intelligence Artificielle a transmis aux Systèmes Multi-Agents est l'utilisation de formalismes logiques à la manière des Systèmes à Base de Connaissances (pour plus de détails voir [Werner, 1996]). Suivant cette approche, un agent dispose d'une base de connaissances représentant son état courant et d'un mécanisme d'inférence modélisant son raisonnement. Un ensemble d'opérateurs modaux (correspondant aux états possibles d'un agent) sont définis et appliqués à des propositions représentant un état de l'environnement de l'agent, alors que des axiomes fixent la sémantique de ces opérateurs modaux.

Dans beaucoup de formalismes logiques développés pour des agents, on trouve un opérateur modal pour exprimer la croyance. L'ensemble des croyances d'un agent correspond à l'ensemble des propositions associées à l'opérateur modal de croyances. La représentation des autres est enfouie au sein de ces croyances comme un ensemble de propositions portant sur les états des autres agents et donc utilisant ces mêmes opérateurs modaux pour les exprimer.

Par exemple, si pour signifier que l'agent $i$ croit la proposition $\alpha$, on écrit $B_i\alpha$, on notera, si l'agent $i$ croît que l'agent $j$ croît la proposition $\alpha$, $B_iB_j\alpha$. La seconde composante de la représentation des autres est contenue dans les mécanismes d'inférence de l'agent $i$. Considérons que nos deux agents $i$ et$j$ raisonnent grâce à des règles d'inférence. L'agent $j$ possède ses propres règles d'inférence que l'agent $i$ doit se représenter s'il souhaite suivre l'évolution des états de $j$. Ces règles sont alors transposées dans la base de connaissances de $i$ pour y ajouter son opérateur de croyances. Ainsi une règle $B_j\alpha \rightarrow B_j\phi$ se retrouve chez $i$ sous la forme : $B_iB_j\alpha \rightarrow B_iB_j\phi$.

La croyance est un état commun aux différents formalismes existants qui ont chacun développé d'autres opérateurs spécifiques. Shoham [Shoham, 1990]

a centré son formalisme sur l'activité des agents en insérant des opérateurs exprimant la capacité des agents, $CAN^t(a, \alpha)$ (l'agent $a$ peut, à l'instant $t$, rendre la proposition $\alpha$ vraie) et l'engagement d'un agent vis-à-vis d'un autre ou de lui-même $CMT^t(a, b, \alpha)$ (l'agent $a$ s'est engagé à l'instant $t$ auprès de l'agent $b$ à rendre la proposition $\alpha$ vraie).

Le formalisme BDI de [Rao and Georgeff, 1991] s'appuie sur une logique arborescente où un agent connaît un ensemble de mondes possibles (chaque monde possible est un ensemble d'états mentaux de l'agent). Ceux-ci sont attachés par des relations d'accessibilité pour représenter le fait qu'un agent passe d'un monde possible à un autre (et donc que les états mentaux de cet agent évoluent) formant une structure arborescente dont le monde courant de l'agent est la racine. Les opérateurs modaux définissant les états mentaux d'un agent sont des croyances, des désirs et des intentions. Les désirs (ou buts) représentent l'ensemble des mondes dans lesquels l'agent désire être ainsi que les chemins pour y accéder. Les intentions sont les choix qu'a effectué l'agent parmi ses désirs (l'ensemble des intentions est un sous-ensemble des désirs). A l'instar de la représentation des croyances d'autrui, un agent $i$ se représente dans ses propres croyances, les capacités et engagements d'un agent $j$ pour Shoham et leurs désirs et intentions dans une logique BDI.

## 2.5 Représentation des capacités

Dans les systèmes coopératifs, les agents agissent conjointement pour la réalisation d'un but commun ou se coordonnent pour éviter les conflits potentiels. La coopération peut être atteinte en raisonnant sur les autres et/ou en communiquant avec eux et peut faire intervenir des représentations de leurs préférences ou états mentaux. Cependant, il faut savoir à la base ce que les agents peuvent faire pour en déduire quels sont les objets ou les moyens d'une coopération.

Un des premiers systèmes incluant explicitement une représentation des capacités des autres est le système MACE [Gasser et al., 1987]. Chaque agent s'y représente ses accointances par des plans, des buts et des compétences permettant d'atteindre un de ces buts. Cette connaissance sur les autres est utilisée pour établir des coopérations avec les agents bénéficiant des compétences souhaitées.

Ce type de représentation peut aussi s'avérer nécessaire dans des contextes autres que coopératifs. Dans les travaux que nous présentons ci-dessous, le système IMBBOP l'utilise pour que chaque agent sache comment ses adversaires peuvent agir en sa défaveur alors que le modèle de description externe défini par Sichman est tourné vers la détection du potentiel de coopération et le système STEAM vers la formation et la supervision d'équipes d'agents.

### 2.5.1 Le système IMBBOP

Dans un contexte hautement compétitif où les buts des agents sont opposés, on considère que, quelque soit l'action que décide d'entreprendre un agent $i$ pour atteindre un but $g$, un agent $j$, mis en compétition avec $i$, décide de ses actions de manière à empêcher la satisfaction de $g$. Dans ce cas $i$ ne décide pas de l'action à accomplir en fonction des préférences de $j$. Il sélectionne une action qui l'approche le plus possible de $g$ et qui ne peut être contre-carrée par aucune action de $j$. Le système IMBBOP (*Ideal-Model-Based Behavior Outcome Production*) [Stone et al., 2000] est un module intégré au raisonnement d'agents footballeurs dans le cadre de simulations de football robotique, qui modélise les adversaires d'un agent par les actions qu'ils peuvent effectuer.

Un agent $i$ connaît ses propres capacités (i.e. les actions qu'il peut faire et leurs effets) et dispose de connaissances incomplètes sur la dynamique du monde (par exemple le comportement du ballon) et sur les capacités des autres agents. $i$ a un but $g$ (par exemple marquer un but) et doit sélectionner une action l'en approchant. Soit $j$ un adversaire de $i$, l'agent $i$ sait que $j$ va tenter d'agir de manière à l'empêcher d'atteindre $g$. Il choisit alors son action de telle sorte qu'aucune action de $j$ ne puisse l'empêcher de s'approcher de $g$, en prenant en compte le temps requis pour son action et pour celles de $j$ (la différence de temps doit dépasser un certain seuil pour que $i$ puisse effectuer son action en toute tranquilité).

Les partenaires de l'agent $i$ sont représentés suivant le même modèle et sont sujets à un raisonnement similaire, la différence étant qu'un partenaire $k$ œuvre pour la satisfaction de $g$. L'agent $i$ utilise ses connaissances sur les autres et peut estimer que l'agent $k$ est le plus apte à satisfaire (ou à s'approcher le plus près de) $g$. Dans ce cas, il choisit une action (passer le ballon) permettant à $k$ d'agir en vérifiant que celle-ci ne peut pas être bloquée par un adversaire. Pour estimer que l'agent $k$ est mieux placé pour agir, l'agent $i$ utilise son modèle des capacités de $k$ et évalue la différence de temps avec les actions des adversaires en élevant le seuil de sécurité pour prendre en compte le temps requis par l'action de $i$.

### 2.5.2 Description externe pour le raisonnement social

Un ensemble d'agents coopérant pour la réalisation d'un but commun est appelé une *coalition*. Pour [Wooldridge and Jennings, 1994], la formation de coalitions nécessite plusieurs étapes dont la première est la reconnaissance du potentiel de coopération. Celle-ci s'effectue, au sein d'un agent, en raisonnant sur les propriétés des autres agents et en détectant les complémentarités de chacun.

C'est dans cette problématique que J. S. Sichman [Sichman, 1995] propose un mécanisme de raisonnement social utilisant la théorie des dépen-

dances [Castelfranchi et al., 1992] et la notion de description externe, qui est une représentation d'un agent. Une description externe définit un agent de manière fonctionnelle. Il ne s'agit pas ici de représenter les intentions ou les préférences de ses accointances mais de connaître leurs capacités ou leur savoir-faire en vue de reconnaître leur potentiel de coopération. Une description externe $ED$ est composée d'un quintuplet $< id, G, A, R, P >$ où :

- $id$ est l'identifiant de l'agent décrit;
- $G$ est l'ensemble des buts que l'agent cherche à atteindre;
- $A$ est l'ensemble des actions que l'agent peut entreprendre;
- $R$ est l'ensemble des ressources sur lesquelles l'agent a un certain contrôle;
- $P$ est l'ensemble des plans dont dispose l'agent pour atteindre un de ses buts. Un plan étant une séquence d'actions instanciées par des ressources.

La notion de plan dans une description externe entraine directement la nécessité d'une coopération des agents. Alors que les buts, actions et ressources sont des éléments indépendants, un plan satisfait un but et requiert l'accomplissement d'actions et l'utilisation de ressources (respectivement définis par les prédicats $achieves(p, g)$, $uses_a(p, a)$ et $uses_r(p, r)$ où $p$ est un plan, $g$ est un but, $a$ est une action et $r$ est une ressource). Ainsi on peut distinguer les plans *autonomes* et les plans *dépendants* au niveau des actions, selon que l'agent puisse ou non effectuer toutes les actions spécifiées dans le plan. On dira également qu'un plan est autonome ou dépendant au niveau des ressources, selon que l'agent puisse ou non utiliser toutes les ressources spécifiées dans le plan. Ces situations de plans sont utilisées pour caractériser les relations entre agents, d'abord sous la forme de *dépendances de base* définies formellement par :

$$basic\_dep_a(i, j, g, p, a) \Leftrightarrow achieves(p, g) \wedge needs_a(i, p, a) \wedge is_a(j, a) \quad (2.4)$$

$$basic\_dep_r(i, j, g, p, r) \Leftrightarrow achieves(p, g) \wedge needs_r(i, p, r) \wedge is_r(j, r) \quad (2.5)$$

Une dépendance de base en action $basic\_dep_a(i, j, g, p, a)$ exprime le fait que $p$ est un plan satisfaisant $g$, que l'agent $i$ a besoin d'une action $a$ pour effectuer $g$ (prédicat $needs_a(i, p, a)$) et que l'action $a$ peut être effectuée par l'agent $j$. Une dépendance de base en ressource $basic\_dep_r(i, j, g, p, r)$ exprime le fait que $p$ est un plan satisfaisant $g$, que l'agent $i$ a besoin d'une ressource $r$ pour effectuer $g$ (prédicat $needs_r(i, p, r)$) et que la ressource $r$ peut être utilisée par l'agent $j$.

Sichman étend par la suite la dépendance aux buts (d'après la situation des plans les satisfaisant) puis par des relations de dépendances plus générales entre agents, tissant ainsi un réseau de dépendances et de situations de dépendances. Celles-ci sont utilisées par un agent pour enrichir les processus et les critères de choix de buts et de plans par un raisonnement social et pour influer sur ses choix de partenaires dans la formation d'une coalition.

### 2.5.3 Représentation d'équipes

Jusqu'ici, nous avons présenté des approches de la représentation des autres centrées sur les agents dans le sens où une représentation peut être isolée et se réfère à un seul agent. Les représentations d'équipes regroupent plusieurs agents sous une seule structure définissant ses propriétés sans être obligées de préciser si une caractéristique est l'apanage de tel ou tel agent (ou de la coopération d'une partie de l'équipe).

Les travaux les plus importants dans ce domaine sont ceux de Milind Tambe implantés dans le système STEAM [Tambe, 1997] (a Shell for TEAMwork) proposant un cadre de conception d'équipes d'agents. Une équipe se définit par deux hiérarchies :

- Une hiérarchie pour l'organisation de l'équipe et des rôles : une équipe peut être composée de plusieurs équipes lui conférant alors une organisation hiérarchique. Chaque membre d'une équipe est un rôle qui peut être soit persistant (assigné à long terme), soit spécifique à une tâche (il n'est valable que pendant le temps d'exécution d'une tâche). Ces rôles peuvent être remplis par des agents ou pour des sous-équipes.

- Une hiérarchie d'activités : l'activité d'une équipe est exprimée par des *opérateurs d'équipe* qui sont des plans réactifs. Chaque opérateur a un ensemble de préconditions qu'il faut vérifer pour l'activer, des règles d'application (ce qu'il faut faire quand il est activé) et des règles de terminaisons. Une équipe *agit* quand les préconditions d'un de ses opérateurs sont vérifiées.

La coordination entre agents est régie par des équipes en assignant les rôles d'une équipe à des agents. La connaissance d'un agent sur les autres agents et sur leurs cadres de coopération est en fait une connaissance des équipes auxquelles il participe. Dans STEAM, un agent connaît intégralement ses équipes, dans le sens où il connaît les hiérarchies d'activité et de rôles ainsi que les agents qui remplissent ces rôles. C'est une connaissance mutuelle, c'est à dire qu'en plus de connaître ces informations, il sait que les autres agents de l'équipe les connaissent aussi et il sait qu'ils savent que chacun les connaît, ...

En complément de STEAM, la technique de modélisation des autres agents SAM (*Socially Attentive Monitoring*) [Kaminka et al., 1998] intro-

duit un aspect dynamique en observant le comportement des autres agents. En utilisant la connaissance commune de deux agents impliqués dans une même équipe, l'un d'entre eux peut émettre des hypothèses sur le plan que déroule l'autre agent en reconnaissant l'action en cours comme une étape de ce plan. La reconnaissance de plan est utilisée pour détecter des incohérences dans le déroulement d'un plan commun ou dans des croyances jugées communes.

## 2.6 Synthèse

Les modèles de représentation des autres que nous avons présentés sont très variés car ils sont très dépendants du contexte et de la catégorie de systèmes multi-agents traitée. Un agent ne peut pas se représenter de la même manière ses accointances selon qu'il doive coopérer ou non avec elles, selon qu'elles soient bienveillantes ou non, ...On peut dégager plusieurs critères caractérisant ces modèles de représentation des autres :

- **la disponibilité des informations :** une représentation d'un agent fait référence à des informations internes à cet agent. Quand ces informations ne sont pas directement accessibles (parce qu'elles sont cachées ou pour éviter la multiplication des communications entre agents), les agents ont la possibilité d'observer l'agent à représenter ou de demander ces informations à un tiers. La précision et la confiance dans ces données sur les autres sont plus faibles lorsque l'acquisition d'information par observation fait appel à un processus d'apprentissage (par exemple en théorie des jeux) et un autre agent peut aussi avoir de fausses informations. Dans ce cas, un agent maintient souvent une alternative de représentation d'un autre pour considérer plusieurs possibilités (comme les probabilités de modèle dans RMM). Quand les caractéristiques des autres agents sont disponibles, les agents doivent avoir des modèles homogènes de représentation des autres (ex : modèle de description externe). Si chaque agent communique des informations personnelles ou les rend accessibles, il doit utiliser un modèle compris par son interlocuteur, et pour ne pas multiplier les représentations réflexives d'un agent, on utilise un modèle commun aux agents.

- **Le niveau du raisonnement :** cette distinction regroupe d'un côté les représentations des préférences et des états mentaux et de l'autre la représentation des capacités. Ce critère différencie la représentation de ce qu'un agent *veut* faire et de ce qu'il *peut* faire. Le premier cas est centré sur la décision d'un agent pour prévoir son comportement futur. Un agent se représente les critères de décision des autres sous la forme de préférences ou d'intentions. Il faut dans ce cas savoir aussi ce qu'un agent peut faire mais c'est un pré-requis plus qu'une fin en

soi. Dans le cas où un agent se représente les capacités des autres (ce qu'ils peuvent faire), un agent ne se préoccupe pas de l'activité future des autres.

- **La réflexivité :** comme nous l'avons vu précédemment, une représentation réflexive est une représentation qu'un agent a de ses propres caractéristiques. Une représentation réflexive permet à un agent de considérer le système dans son ensemble et d'interpréter ses interactions avec les autres de la même manière qu'il interprète les interactions entre deux autres agents (représentation des états mentaux, système RMM, description externe). Si une telle représentation n'existe pas, un agent ne se met pas au même niveau que les autres, ce qui permet d'obtenir une société d'agents hétérogènes (il n'a pas à être exprimé dans le même modèle que les autres). C'est notamment le cas en théorie des jeux.

- **L'auto-suffisance des agents :** un agent auto-suffisant n'a pas besoin de coopérer avec d'autres agents pour atteindre ses objectifs. S'il doit se représenter et coopérer avec d'autres agents, c'est pour éviter les conflits ou interférences potentiels. On est alors dans un contexte compétitif où les agents tentent d'empêcher les autres d'atteindre leurs objectifs, soit volontairement soit suite à une incompatibilité avec leurs propres objectifs (ex : certains cas en théorie des jeux), ou bien les agents évoluent dans un environnement partagé où les actions de l'un peuvent influer sur les états des autres (ex: RMM, ou d'autres cas de théorie des jeux). Si les agents doivent coopérer, ils doivent se représenter le potentiel de coopération de leurs accointances. On trouve dans ce cas les représentations des capacités des autres.

Ces critères sont assez généraux et permettent de dégager les caractéristiques essentielles des contextes des travaux présentés d'où découlent les différents modèles. En combinant les différents critères on peut généraliser les travaux sur la représentation des autres en deux grandes catégories de contexte :

- **Les contextes compétitifs :** les agents sont en concurrence avec les autres par leurs buts ou préférences. Chaque agent peut agir seul (*auto-suffisant*) et doit prévoir ce que les autres *vont* faire. Les agents n'ont *pas de représentation réflexive* et évidemment les informations les concernant sont *inaccessibles.*

- **Les contextes coopératifs :** les agents ont des buts communs ou s'entraident dans la réalisation de leurs buts individuels. Ils ne sont *pas auto-suffisants* et se représentent ce que les autres *peuvent* faire. Ils rendent les informations sur eux-mêmes *disponibles* et disposent par conséquent d'une *représentation réflexive.*

Dans nos travaux, nous nous sommes particulièrement intéressés à la conception de systèmes multi-agents ouverts (que nous décrivons chapitre 3) dans un contexte coopératif. Notre approche se fonde sur la détection du potentiel de coopération entre agents ce qui nous a naturellement guidé vers un modèle proche de la description externe de Sichman qui lui accorde une grande importance. De plus, le potentiel que nous exprimons en terme de *besoins* d'un agent est une notion assez proche des dépendances calculées à partir des descriptions externes.

Dans notre volonté de distribuer la tâche d'intégration de nouveaux agents, nous avons assigné à chaque agent de la société les mécanismes nécessaires à la détection du potentiel de coopération entre lui-même et un nouvel agent ainsi que la tâche de faciliter l'intégration de cet agent avec les autres agents du système. Un agent doit pouvoir recommander certaines de ses accointances. Nous avons présenté en début de ce chapitre des techniques de recommandation, utilisés dans la gestion de profils utilisateurs, par le biais de stéréotypes. Nous nous sommes également inspirés des travaux sur les stéréotypes utilisateurs pour les adapter aux systèmes multi-agents et construire des stéréotypes d'agents (voir section 4.3).

Le chapitre suivant est consacré à un état de l'art sur les systèmes multi-agents ouverts, en appuyant plus particulièrement notre propos sur les agents. Notre analyse nous amènera à considérer le modèle de représentation des autres, son acquisition et sa mise à jour comme le cœur du problème de l'ouverture des systèmes multi-agents.

# Chapitre 3

# les Systèmes Multi-Agents Ouverts

L'intérêt croissant porté aux systèmes multi-agents vient principalement du fait que ceux-ci offrent une solution flexible et évolutive aux problèmes auxquels ils sont confrontés. La décomposition d'un système complexe en sous-systèmes autonomes interagissants permet de se focaliser sur une partie de ces sous-systèmes au lieu d'être obligé de considérer le système dans sa globalité.

Si on peut aisément imaginer qu'un tel système est plus flexible et évolutif en limitant l'adaptation à de nouvelles situations aux entités concernées, il faut néanmoins que soient développées des techniques d'adaptation. Le paradigme multi-agent offre un cadre favorable à ces développements mais ne propose pas encore de solutions standards.

Dans le cadre de l'évolutivité d'un système multi-agent, nous nous intéressons plus particulièrement à son ouverture. Un système ouvert est un système auquel on peut ajouter ou enlever de nouvelles fonctionnalités en cours d'exécution. Dans un système multi-agent, ces fonctionnalités sont assurées par les agents le composant. Ainsi un système multi-agent ouvert doit pouvoir assurer l'intégration de nouveaux agents et permettre le départ d'agents présents.

## 3.1 Systèmes ouverts et Systèmes Multi-Agents

Pour pouvoir définir ce qu'est un système multi-agent ouvert, nous commençons, dans cette section, par rappeler les propriétés d'un système ouvert avant d'analyser leurs relations et recouvrements avec celles d'un système multi-agent pour déterminer de quelle manière on peut "ouvrir" un SMA.

### 3.1.1 Les Systèmes Ouverts

Les systèmes ouverts peuvent se caractériser par plusieurs propriétés [Hewitt and Jong, 1982] [Agha and Hewitt, 1985]. *L'hypothèse de monde fermé est levée* car elle est contraire à la nature même de ces systèmes et en inadéquation avec une situation réelle. Ainsi, l'environnement d'un système ne peut pas être complètement représenté et il doit prendre en compte le fait qu'il existe des informations qui lui sont inconnues et inaccessibles.

Une autre caractéristique des systèmes ouverts est la *modularité*. Un système ouvert est composé de plusieurs sous-systèmes, ayant chacun leur fonctionnement propre, et de leur mise en relation. Celle-ci s'effectue par *dissémination* de l'information en dupliquant certaines informations d'un sous-système dans les autres (c'est l'équivalent d'une représentation des autres). Une conséquence de la modularité est *l'absence d'objet global* partagé par plusieurs sous-systèmes. Les relations entre sous-systèmes se limitent à des communications.

Enfin, la dernière propriété des systèmes ouverts qui retient plus particulièrement notre attention est leur extensibilité. Un système ouvert est un système incrémental, c'est-à-dire qu'il est possible de construire un système ouvert par composition de plusieurs systèmes ouverts.

### 3.1.2 L'ouverture dans les Systèmes Multi-Agents

Il découle logiquement, des trois caractéristiques évoquées ci-dessus, que les systèmes ouverts sont des systèmes distribués [Hewitt, 1986]. Le développement des modèles acteurs concrétisant la conception de systèmes ouverts en introduisant ces propriétés, C. Hewitt [Hewitt and Inman, 1991] a même proposé de considérer le domaine des systèmes ouverts comme un fondement pour l'Intelligence Artificielle Distribuée (Les Gasser a critiqué cette position dans [Gasser, 1991]), son principal argument étant que la résolution de conflits est un point essentiel et critique dans les deux domaines. La décentralisation du contrôle dans ces systèmes crée de multiples centres de décision, chacun ayant des intérêts et des connaissances qui lui sont propres, faisant apparaître la possibilité de micro-décisions contradictoires ou incompatibles et donc un conflit à résoudre pour maintenir la cohérence globale du système.

Pour qu'un système multi-agent puisse être considéré comme ouvert, celui-ci doit respecter les propriétés d'un système ouvert. Nous nous sommes plus particulièrement intéressés à la gestion de l'extensibilité d'un système multi-agent. Dans un système ouvert, on considère un service (ou une fonction) comme l'élément de base du système. On dit que le système est extensible dès lors que l'on peut ajouter de nouveaux services à ceux initialement prévus. Dans un système multi-agent, ces services sont assurés par des agents et sont obtenus par l'activité d'un agent ou par l'activité conjointe de plu-

sieurs d'entre eux. L'agent représente ici l'élément de base. Pour ajouter de nouveaux services au système, il faut modifier la projection existante entre l'ensemble des agents et l'ensemble des services, ce qui peut être effectué de deux manières :

1. En créant un agent encapsulant ce nouveau service et en ajoutant l'agent à la société existante;

2. En ajoutant ce service à un ou plusieurs agents déjà présents dans le système.

Ces deux approches doivent être distinguées car elles impliquent des mécanismes très différents. Dans le premier cas, on parlera de *Système Multi-Agent Ouvert* car l'ajout d'un agent au SMA modifie sa composition en terme d'agents (ses éléments de base). Un système multi-agent étant généralement coopératif, le mécanisme d'intégration d'un nouvel agent doit établir des liens entre le nouvel agent et certains agents du système, la base de ces liens étant une connaissance réciproque (un modèle de l'autre).

La seconde approche consiste à modifier le "contenu" d'un agent. On parlera alors de *Système d'agents évolutifs* car ceux-ci peuvent évoluer dans le temps. Il faut que le système prévoit des mécanismes de propagation des changements effectués pour maintenir la cohérence globale dans les liens entre agents. Les agents peuvent être considérés comme évolutifs parce qu'ils permettent une modification "manuelle" des services qu'ils proposent ou parce qu'ils sont apprenants (pour un état de l'art sur les agents apprenants voir [Vercouter et al., 1998]) et les modifient automatiquement.

Pour les deux approches, on peut parler d'évolutivité en précisant que dans le premier cas c'est *le système* qui est évolutif alors que dans le second ce sont *les agents* qui sont évolutifs. On accorde une préférence à l'une de ces approches en fonction des contraintes ou souhaits de conception. Par exemple si le développement d'un agent ou la migration d'agents sont coûteux, on optera pour un système d'agents évolutifs alors que si l'on préfère une approche plus modulaire (pour gérer par exemple des accès sécurisés au contenu des agents), un système multi-agent ouvert est approprié.

Pour laisser au concepteur une certaine liberté de choix entre SMA ouvert et système d'agents évolutifs, la plupart des systèmes existants présentant l'un de ces aspects prévoit également l'autre. Nous présentons dans la section suivante plusieurs travaux existants en focalisant notre propos sur les mécanismes d'ouverture qui leur sont associés.

## 3.2 Les approches existantes

L'ajout d'un nouveau service (dans un agent existant ou à encapsuler dans un nouvel agent) provoque une phase d'instabilité du système global.

L'inter-dépendance des entités nécessite en effet qu'une nouvelle entité soit mise en relation avec les entités concernées pour être utilisable et/ou opérationelle. On peut faire l'analogie avec un étudiant entrant à l'université. On ne peut pas dire qu'il est intégré à l'université uniquement en y entrant physiquement. Il doit en plus connaître les horaires des cours magistraux, s'inscrire à des groupes de travaux dirigés, se faire connaître des services de scolarité, faire la connaissance des autres étudiants pour choisir au mieux ses partenaires lors de travaux collectifs... Dans cette énumération d'exemples, on peut remarquer qu'il est toujours question de *connaissances* sur les différents composants du système (de l'université dans notre exemple).

L'ouverture d'un système multi-agent nécessite que le système permette l'ajout physique d'un nouvel agent mais aussi qu'il fournisse des mécanismes de mise à jour de la connaissance sur les propriétés des agents. Pour qu'un agent interagisse avec un autre, il choisit son partenaire en fonction de ce qu'il connaît de lui, et cette connaissance doit donc lui être accessible. Celle-ci est généralement acquise par communication avec les autres agents.

### 3.2.1 Protocoles de présentation

Dans un contexte de conception de SMA ouvert, les agents sont coopératifs et mettent à disposition des autres agents des informations sur eux-mêmes. Cette mise à disposition se traduit par des échanges de messages où deux agents s'envoient des informations sur eux-mêmes dans un protocole de présentation.

La présentation n'est cependant pas l'apanage de l'ouverture des SMA. Il est également nécessaire dans d'autres contextes de communiquer des informations sur soi-même (par exemple dans un protocole de négociation [Sian, 1990] [Chang and Woo, 1991]). Le problème qui se pose, d'une manière générale, dans une présentation est de déterminer quelles informations échanger et comment évaluer leur pertinence. Par exemple, la coordination distribuée de plusieurs agents nécessite que chacun connaisse quelques informations sur les activités des autres pour se coordonner localement avec eux. Le *Partial Global Planning* [Decker and Lesser, 1992] se fonde sur l'échange de tâches générales à accomplir par les différents agents. Chaque agent effectue une tâche complexe qui se décompose en sous-tâches. Le problème qui se pose est d'ordonnancer l'exécution de ces sous-tâches, ou de choisir lesquelles exécuter si elles sont concurrentes. Les tâches sont inter-dépendantes, c'est-à-dire qu'une tâche peut faciliter ou au contraire nuire à l'exécution d'une autre tâche.

En partant du principe que les agents connaissent ces relations entre tâches, le contenu des messages de présentation peut se limiter aux tâches générales. Ainsi, les agents s'échangent leur tâche générale et chacun peut déduire si, parmi les sous-tâches qu'il peut effectuer, il y en a qui facilitent l'exécution de la tâche générale d'un autre ou non. Si de telles relations sont

détectées, l'agent effectuant cette sous-tâche en informe celui qui a la tâche générale concernée. Ces informations sur les activités des autres donnent à chaque agent des critères supplémentaires dans leur choix de sélection ou d'ordonnancement de sous-tâches (par exemple si un agent doit choisir entre deux sous-tâches concurrentes sans préférences locales, il priviligiera celle qui facilite le plus de tâches des autres agents).

Pour l'ouverture du SMA, un agent utilise une présentation pour informer les autres de ses propres caractéristiques, et pour acquérir des informations sur les autres. Il est également souhaitable dans ce cas de sélectionner les informations à envoyer, ou les agents à qui les envoyer, pour ne pas encombrer inutilement les canaux de communications avec des informations non pertinentes. S. Berthet [Berthet et al., 1992] propose un protocole de présentation basé sur une représentation de soi-même hiérarchisée en plusieurs niveaux d'abstraction. Un agent se présente à tous les autres par une énumération de *rôles* qu'il peut remplir et d'*actions* qu'il peut effectuer. Seuls les agents qui sont intéressés par ces caractéristiques lui répondent. Les présentations se poursuivent avec les agents intéressés en portant sur des connaissances plus précises comme l'utilisation des actions (par exemple dans des plans) ou l'explication de rôles inconnus d'un agent.

Deux agents prennent part à une présentation car ils ont besoin d'avoir une représentation des autres. La présence de cette connaissance ainsi que la participation à des présentations sont des critères nous permettant de distinguer deux grandes approches de SMA ouvert dans les travaux existants : l'approche **centralisée** dédiant une entité à la gestion de cette connaissance et l'approche **distribuée** la répartissant en représentation des autres au sein des agents du système.

### 3.2.2 Approche centralisée

La centralisation de la connaissance sur les agents est à l'origine un problème d'indexation. Il s'agit en effet de représenter les éléments d'un ensemble en regroupant leurs caractéristiques descriptives dans une entité centrale. On peut trouver une problématique similaire dans d'autres domaines. Par exemple, la recherche d'informations sur Internet nécessite une structuration des documents accessibles en les représentant dans une ou plusieurs entités. Ces entités sont alors utilisées pour traiter des requêtes et trouver un ensemble de documents correspondant aux caractéristiques définies par la requête considérée (voir par exemple [Doan and Beigbeder, 1999]). L'adoption d'un index central gagne aussi les grandes entreprises. Celles-ci sont confrontées à des problèmes de cohérence générés par la duplication de données et la multiplication d'annuaires locaux à certaines applications. Cette multiplication est flagrante pour les données concernant le personnel que l'on peut trouver dans diverses applications (base de donnée des ressources humaines, messagerie, comptes utilisateurs, ...) et il n'est pas rare de constater des

incohérences ou des retards de mises à jour. Pour ces grandes entreprises, la centralisation des données est une solution fréquemment employée (cf. [Rognon, 2000]) via la création d'un *meta-annuaire* fédérant les différents annuaires existants.

Une indexation centrale est également une solution possible dans un système multi-agent. Cependant, on lui préferre parfois un agent particulier, le *broker* (courtier en français), proposant un ensemble de services pouvant s'étendre au-delà de la gestion d'un index.

### Le courtage

En général une coopération est initiée par un agent détectant un manque dans ses compétences et requiérant l'aide d'autres agents. Il doit alors commencer par communiquer avec d'autres agents pour leur demander s'ils souhaitent coopérer. Les destinataires de ce message sont choisis d'après certaines caractéristiques démontrant leur aptitude à coopérer dans le cas considéré. Cependant, une conséquence de la centralisation de la connaissance sur les agents est que l'agent en question ne dispose pas directement des informations nécessaires pour effectuer le choix du destinataire. Le *broker* connaissant les caractéristiques de tous les agents du système doit combler ce manque et peut aider l'agent émetteur du message de deux manières différentes :

1. Il remplit le rôle de *pages jaunes*, c'est-à-dire qu'il effectue la correspondance entre l'adresse des agents et leurs caractéristiques. Les agents peuvent lui demander de leur communiquer une liste d'agents correspondant à un ensemble de caractéristiques données.

2. Il se rapproche d'un rôle de routeur. Les agents lui délèguent le transport de leurs messages en lui indiquant quelles sont les caractéristiques auxquelles doivent répondre les agents destinataires. Il les sélectionne et leur envoie directement le message. Il sert également d'intermédiaire pour véhiculer les réponses à ce message.

Le service fourni par le broker est appelé le *courtage*. En schématisant cette approche, on peut dire que le broker est un passage obligé de la communication entre agents. Chaque agent, pour être intégré à la société, n'a besoin que de connaître le broker et de se faire connaître auprès de lui, produisant une architecture en étoile représentée par la figure 3.1.

L'architecture en étoile est une représentation très générale de l'utilisation d'un broker. En effet des liens transversaux entre agents peuvent être créés dans le cas où un agent demande au broker de l'information sur les autres et où il retient cette information. C'est le même rapport qui existe entre un carnet d'adresse et un annuaire. Si, par exemple, je cherche l'adresse de garages automobiles, je vais dans un premier temps consulter les

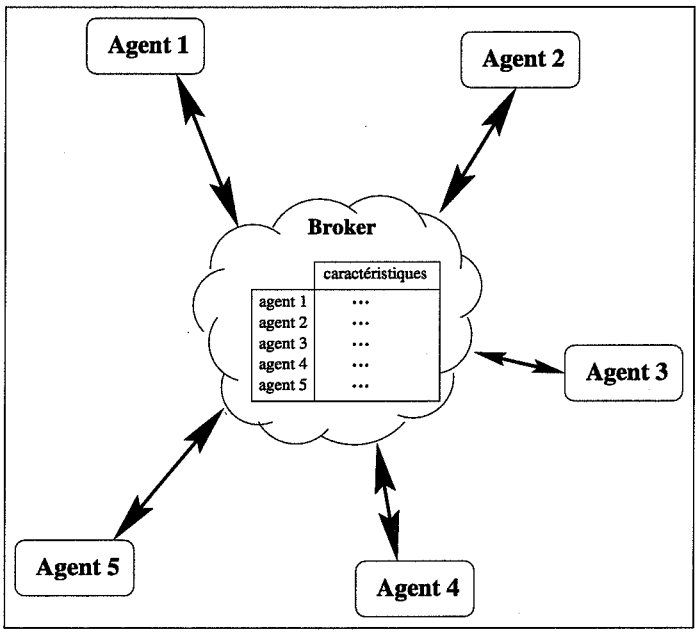


**Figure 3.1 :** Courtage par une architecture en étoile

pages jaunes d'un annuaire pour avoir un numéro de téléphone. Si je pense avoir souvent besoin de les rappeler, je noterai le numéro dans mon carnet d'adresse m'évitant d'avoir à réeffectuer une recherche dans l'annuaire ultérieurement. De la même manière deux agents coopérant peuvent d'abord avoir été mis en relation par le broker puis, à la suite d'une coopération réussie, retenir leurs adresses et caractéristiques pour ne plus avoir à passer par le broker si un cas similaire se reproduit.

Le système ABROSE [Gleizes and Glize, 2000] fonctionne selon ce principe. Quand un agent recherche un autre agent présentant certaines caractéristiques, il consulte d'abord sa propre représentation des autres. Si celle-ci est insuffisante pour répondre à ses besoins, il contacte le broker (appelé agent de médiation) pour acquérir la connaissance nécessaire. Sa représentation des autres est alors mise à jour pour éviter de reproduire la même demande à l'agent de médiation ultérieurement.

Le nœud central correspondant au broker peut également avoir une structure plus complexe. Dans des grands systèmes incluant de nombreux agents avec des caractéristiques variées, il peut être souhaitable de structurer le service de courtage pour accélérer la recherche d'agents pertinents. Dans ce cas l'architecture devient hiérarchique par la définition de plusieurs brokers d'un certain niveau de spécialisation. Cette spécialisation se mesure d'après les caractéristiques des agents que chaque broker indexe. La figure 3.2 donne un exemple de brokers organisés hierarchiquement.

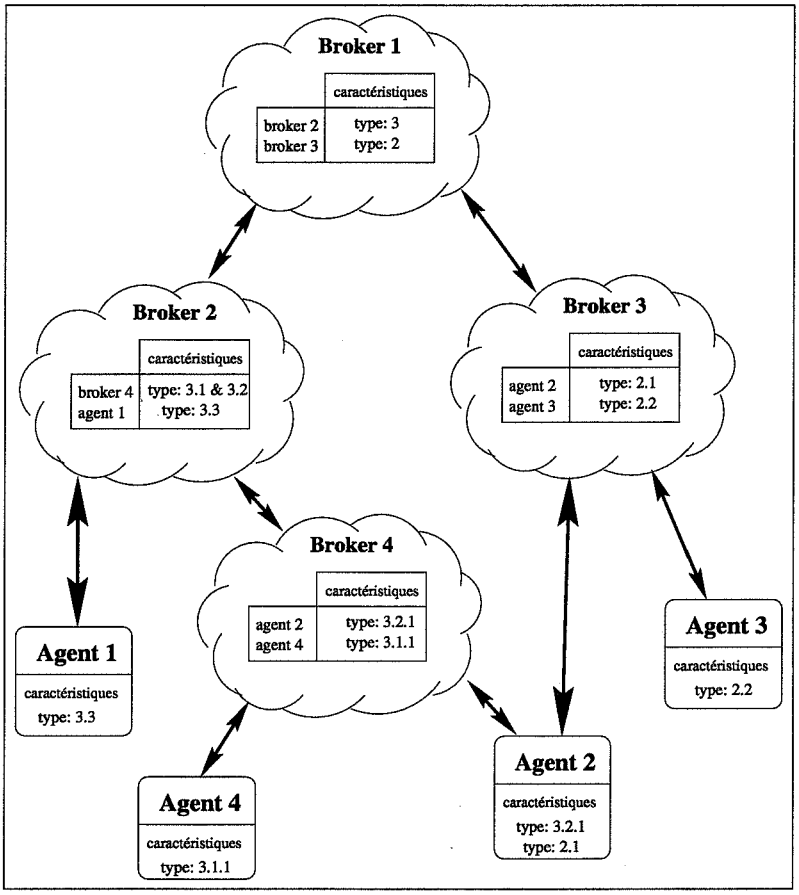


**Figure 3.2 :** Courtage par une architecture hiérarchique

Les agents sont répertoriés dans des brokers correspondant à leurs spécificité. Pour trouver les destinataires appropriés à un message, les brokers sont amenés à interagir et nécessitent un broker plus général de la même manière que les agents ont besoin d'eux. Imaginons, par exemple que l'agent 3 cherche un agent pouvant effectuer une action de type 3.2. Il demande à son broker (broker 3) quels agents ont cette caractéristique. Celui-ci ne peut pas lui indiquer directement l'agent 2 car le broker 3 est spécialisé dans le recensement des agents ayant une caractéristique de type 2.1 ou 2.2 et n'a pas connaissance des autres propriétés de l'agent 2. Il transmet alors la requête au broker plus général qui la renvoie au broker 2 sachant que celui-ci peut connaître les agents ayant une caractéristique de type 3. Le broker 2 reconnaît la spécialisation et passe la requête au broker 4 qui répond avec les adresses des agents 2 et 4. La réponse est renvoyée à l'agent 3 en faisant le chemin inverse.

L'ouverture est une propriété que l'on peut retrouver à différents niveaux dans un SMA. Par exemple, dans MadKit [Gutknecht et al., 2000] les agents sont organisés en groupes qui sont eux-même ouverts. Ainsi un agent peut sortir d'un groupe pour entrer dans un autre groupe, tout cela restant au sein d'un même SMA. Les problèmes qui se posent alors pour gérer l'ouverture sont similaires à ceux présentés précédemment. Dans MadKit, il existe, dans chaque groupe, un unique responsable du groupe qui y joue le rôle de broker.

Nous avons limité, dans nos exemples, le rôle du broker à celui d'un annuaire, voire d'un routeur, ce qui n'en fait pas nécessairement un agent. Par la place privilégiée qu'il occupe au niveau des interactions entre agents, il est possible d'étendre ses attributions [Foss, 1998]. En plus de rechercher des informations sur les autres agents, un broker peut gérer les droits d'accès de chacun, envoyer de lui-même des informations qu'il juge intéressantes, construire un profil de ses utilisateurs (ou agents), pratiquer du data-mining sur des données auxquelles il a accès, gérer lui-même une négociation... C'est alors au concepteur de déterminer quelles tâches il attribue au broker et ce qui est du ressort des agents. Il est même possible de mettre un utilisateur directement en contact avec un broker si celui-ci automatise suffisamment de tâches.

La présence d'un broker dans un système multi-agent facilite grandement son ouverture. Pour intégrer un nouvel agent, nous avons précisé précédemment qu'il fallait établir des liens entre cet agent et les autres agents du système dont il se trouve dépendant ou avec lesquels il peut coopérer. Or, chaque agent n'a ici besoin que de connaître un broker pour pouvoir accéder aux agents l'intéressant. La phase d'intégration comprend uniquement une présentation du nouvel agent à un broker pour que celui-ci l'enregistre dans sa liste d'agents ou trouve les brokers spécifiques auxquels il doit être rattaché.

Cette approche est également adaptée à des agents évolutifs. Chaque agent n'étant représenté que dans un seul broker, il lui suffit de l'avertir si ses caractéristiques évoluent. Dans une architecture hiérarchique, il peut se trouver redirigé simplement vers d'autres brokers. Pour avoir des exemples de systèmes utilisant des brokers, le lecteur peut consulter [Cohen et al., 1998] pour une architecture en étoile, et [Bradshaw, 1997] pour une architecture hiérarchique.

### Les spécifications FIPA

Dans ses spécifications d'une plate-forme d'agents, la FIPA (*Foundation for Intelligent Physical Agents*) utilise une approche de type broker pour référencer à plusieurs niveaux et sous différents aspects les agents de la plate-forme. C'est une extension de la notion de faciliteur proposée par [Labrou and Finin, 1994]. Le modèle de référence pour la gestion des agents [FIPA, 2000a] est représenté par la figure 3.3.

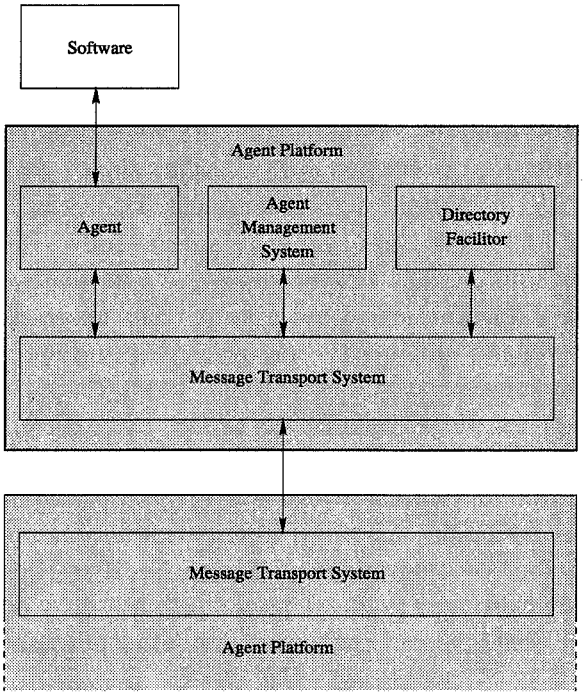


**Figure 3.3 :** Modèle de référence FIPA d'une plate-forme d'agents

Nous ne détaillons pas ici le composant *Message Transport System* (MTS), qui gère les communications entre les agents de la plate-forme et avec d'autres plate-formes, pour se focaliser sur les *Agents*, le module *Agent Management System* (AMS) et le *Directory Facilator* (DF) qui peuvent chacun remplir un rôle proche du broker.

Le composant AMS, unique dans une plate-forme d'agents, est un outil de supervision et de contrôle des agents. Les informations dont il dispose sur les agents de la plate-forme ont trait à leur activité informatique. Il connaît leur nom, adresse et leur état courant dans le cycle de vie d'un agent (actif, suspendu, ...). Chaque agent est censé s'enregister auprès de l'AMS pour avoir accès aux fonctions de la plate-forme (communiquer, consulter le DF, ...).

Le *Directory Facilitor* remplit réellement le rôle de broker dans la plate-forme. Un agent de la plate-forme s'enregistre auprès du DF en décrivant les *services* qu'il souhaite mettre à la disposition des autres agents. Chaque *service* est identifié par un *type* permettant au DF de les regrouper en catégories et de servir de pages jaunes. Un agent de la plate-forme cherchant quels autres agents peuvent effectuer un service d'un type donné, envoie sa requête au DF qui lui répond par la liste des adresses des agents y correspondant. En plus d'un ensemble de *services*, le DF connaît pour chaque agent

quels *protocoles d'interaction* et quelles *ontologies* il connaît, ainsi que les *langages* qu'il comprend. Plusieurs DF peuvent exister dans une plate-forme, chacun répertoriant certains type de services à l'instar des architectures hiérarchiques de courtage. Chaque DF connaît un ensemble d'autres DF dont la description contient un service de type *fipa-df*.

Ces deux composants sont suffisants pour établir des relations entre agents. Cependant la FIPA s'est également intéressée à l'intégration de logiciels non-agent [FIPA, 2000b], et a spécifié, pour atteindre cet objectif, des rôles d'agent spécifiques:

- des agents *wrapper* qui encapsulent un ou plusieurs logiciels. Cette fonction est représentée par un service de type *fipa-wrapper* et étiquetée par une description du logiciel.
- des *Agent Resource Broker* qui sont l'équivalent des DF mais pour les logiciels non-agents. Leur service de pages jaunes effectue une correspondance entre une description de logiciel et un ensemble d'adresses d'agents wrapper. Les ARB sont identifiés auprès des DF comme proposant un service de type *fipa-arb*.

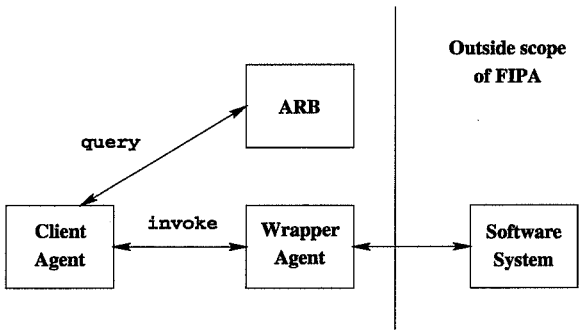


**Figure 3.4 :** Scénario général d'une intégration de logiciel

La figure 3.4 illustre le déroulement d'une recherche d'un logiciel par un agent client. Un agent client contacte un ARB (dont il a pu avoir l'adresse en demandant à un DF) par l'envoi d'une requête spécifiant la description du logiciel qu'il recherche. Dans ce scenario, l'ARB lui donne l'adresse d'un agent wrapper correspondant de manière à ce que le client puisse l'utiliser pour accéder au logiciel.

Ces spécifications permettent de définir un système ouvert au niveau des agents et des services. L'intégration d'un nouvel agent suit deux étapes: (i) il s'enregistre auprès de l'AMS, ce qui correspond à son ajout "physique" au système, (ii) il envoie une description des services qu'il met à la disposition des autres au DF, ce qui permettra par la suite à d'autres agents de le

contacter. L'intégration d'un nouveau service logiciel suit les deux mêmes étapes, excepté qu'il faut commencer par créer un agent wrapper décrivant le logiciel et que l'envoi de la description se fait à un ARB au lieu d'un DF.

### 3.2.3 Approche distribuée

La centralisation des informations sur les agents du système dans une entité ou un petit groupe d'entités semble adaptée à l'intégration de nouveaux agents. Néanmoins, cela implique l'existence d'entités critiques pour la cohérence globale ce qui peut fragiliser le système (cf. section 3.3). De plus, il est parfois souhaitable de répartir la connaissance sur les autres agents parmi les agents du système pour faciliter l'accès à ces informations et limiter les communications vers un broker.

#### La présentation diffuse

En distribuant ces informations dans la connaissance des agents, le problème de leur mise à jour suite à l'ajout d'un nouvel agent se complexifie. La solution la plus simple, utilisée dans les travaux nécessitant que chaque agent ait une représentation des autres mais n'en faisant pas une priorité de recherche, est d'imposer à un nouvel agent une phase de présentation pendant laquelle il envoie sa description à chaque agent du système. Dès la réception d'un message de ce type, un agent répond avec sa propre description.

Cette solution ne nécessite pas d'entité centrale mais est très lourde car elle implique de très nombreuses communications pouvant se révéler inutiles. Non seulement un agent doit envoyer autant de messages qu'il existe d'agents (et recevoir autant de réponses) à son arrivée mais il peut aussi avoir à recommencer cette phase de présentation s'il est évolutif à chaque fois que l'ensemble des services qu'il propose change. Il n'est pas possible d'imaginer que cet agent filtre les agents à qui il envoie cette mise à jour car les agents non-informés disposeraient d'une représentation fausse et n'auraient pas les moyens de détecter une nouvelle dépendance si ceux-ci sont également évolutifs. Une présentation diffuse pose également l'hypothèse qu'un agent a accès à l'adresse de tous les agents du système, ce qui n'est pas imaginable dans de grands systèmes ouverts sans un index central.

#### Le système Gnutella

Plutôt que de multiplier les communications pendant l'intégration d'un agent, il existe une autre approche distribuée visant à limiter la représentation des autres dans la connaissance d'un agent tant au niveau du nombre d'agents connus que sur le contenu de chaque représentation. Ce principe est appliqué par le système Gnutella [Gnutella, 2000] pour gérer le partage de fichiers, via Internet, par un ensemble d'utilisateurs. Il n'est ici pas question d'agents mais de logiciels associés au disque dur d'un utilisateur. Un

utilisateur choisit sur son disque les fichiers qu'il souhaite partager (rendre téléchargeables par d'autres utilisateurs) dont l'ensemble forme sa description.

La représentation qu'un logiciel se fait des autres logiciels est réduite au minimum, c'est-à-dire à leur adresse. De plus, chaque logiciel limite le nombre d'accointances à une valeur choisie arbitrairement. La nature même du système fait que les migrations sont très fréquentes parmi les entités du système et il faut faire attention à ne pas isoler une partie du réseau. La figure 3.5 illustre la composition de la représentation des autres.

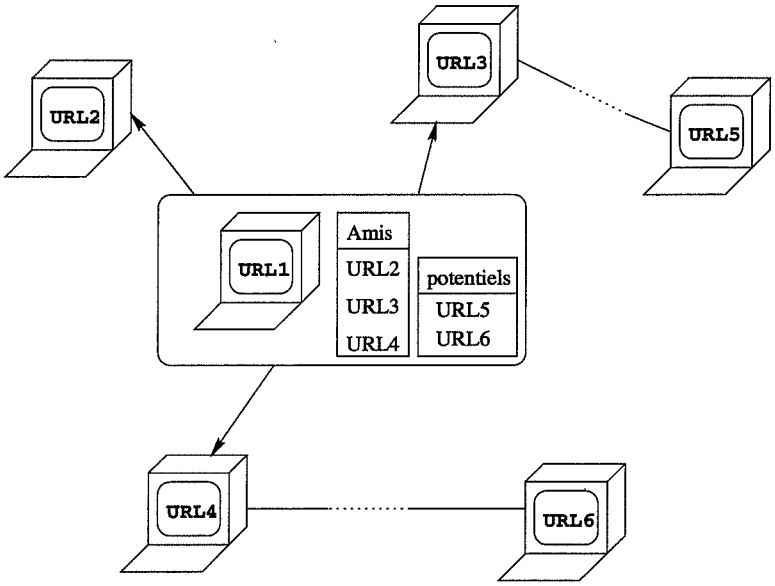


**Figure 3.5 :** Le système Gnutella

On se place du côté de l'ordinateur situé à l'URL1 qui maintient une liste de trois accointances. Il connaît dans cette liste les URL 2, 3 et 4 considérées comme ses amis. Pour maintenir son nombre d'amis à trois, il doit également connaître d'autres adresses dans une table d'"amis potentiels", dont il a obtenu l'adresse pendant une communication antérieure (comme nous le verrons, elles sont très nombreuses). Dès qu'un utilisateur ami se déconnecte du système, son adresse n'est plus valide, et le logiciel pioche aléatoirement dans sa liste d'amis potentiels pour passer l'adresse choisie dans la liste de ses amis et maintenir son nombre d'amis à trois. Dans l'exemple de la figure, si l'utilisateur situé à l'URL 2 disparaît il sera remplacé par celui de l'URL 5 ou 6.

L'inconvénient de cette approche est que la représentation des autres limitée à une simple adresse ne permet pas à un logiciel d'interagir intelligemment avec les autres. Quand un utilisateur cherche un type de fichier, il

envoie une requête à son logiciel qui la transmet à tous ses amis. Ceux-ci, s'ils ne peuvent pas y répondre favorablement, la propagent en l'envoyant à leurs propres amis et ainsi de suite... Pendant une recherche, ce système est très coûteux en communications mais c'est aussi ce qui permet à chacun d'obtenir régulièrement de nouvelles adresses dans sa liste d'hôtes potentiels.

L'ajout d'un nouvel utilisateur passe par une première connexion. Il suffit de donner au logiciel l'adresse d'un nœud du réseau comme ami, pour que celui-ci commence à recevoir des requêtes, via son ami, et retiennent les adresses des émetteurs comme amis jusqu'à atteindre son seuil d'adresses amies et ajouter les autres comme amis potentiels.

## 3.3 Synthèse

En partant de la définition des systèmes ouverts donnée par Carl Hewitt, nous avons établi qu'un SMA, pour être ouvert, doit mettre en place des mécanismes gérant son extensibilité. Nous avons constaté également que certains systèmes doivent gérer le départ d'agents et ce que cela implique au niveau de la coopération dans le système. En considérant qu'un système multi-agent peut être extensible au niveau des agents ou des services proposés, nous pouvons définir trois sous-catégories de systèmes multi-agents ouverts associant à chacun une tâche correspondant à sa propriété:

- Les *SMA incrémentaux* permettant l'ajout de nouveaux agents suivant un processus d'addition de nouveaux agents (figure 3.6).
- Les *SMA décrémentaux* autorisant un retrait d'agent par un mécanisme de soustraction d'agents (figure 3.7).
- Les *systèmes d'agents évolutifs* où certains agents peuvent évoluer (par apprentissage ou modification manuelle) nécessitant une technique de mise à jour de la représentation des autres (figure 3.8).

Les étapes qui nous intéressent dans chaque processus sont celles de mise à jour de la connaissance. Elles correspondent à l'*intégration d'un agent* pendant son ajout, ou la *notification* d'un départ ou d'une modification. Le trait caractéristique de chaque approche se situe dans ces étapes et plus particulièrement dans leur assignation à un agent ou à un ensemble d'agents.

On peut regrouper ces trois étapes dans un même service qui gère l'ouverture du système, et il faut qu'il soit assuré par des agents ou entités. En parcourant de nouveau les travaux existants on retrouve ce service assigné de manière:

- **centrale** dans un broker (ou équivalent) existant principalement pour assumer cette charge, car l'intégration revient à une notification de présence au broker et toutes ces notifications ne nécessitent qu'un message (voire sa réponse) vers le broker;

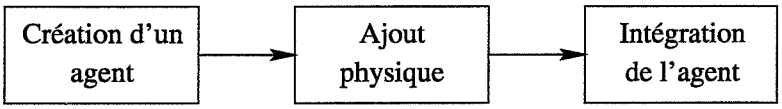


**Figure 3.6 :** ajout d'un agent dans un SMA incrémental

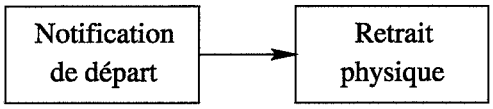


**Figure 3.7 :** retrait d'un agent dans un SMA décrémental

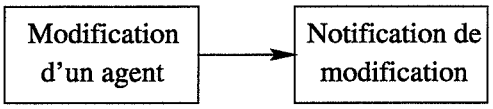


**Figure 3.8 :** modification d'un agent pour un agent évolutif

- **distribuée**, le service d'ouverture est réparti parmi les agents et chacun doit en assurer une partie pour que le SMA soit globalement ouvert.

La quantité de communications dans un système multi-agent est souvent considérée comme une mesure importante de coût. L'approche distribuée peut se révéler lente et coûteuse car elle nécessite de nombreuses communications à chaque ajout, retrait et modification d'agent (pour une présentation diffuse) ou reporte ce problème pendant les interactions entre agents en ne recueillant pas suffisamment d'informations sur les autres pendant l'intégration des agents (dans le système Gnutella par exemple).

L'approche utilisant un broker est alors séduisante mais elle présente l'inconvénient évident d'être centralisée. Les agents n'ont pas directement accès à la connaissance sur les autres et doivent communiquer avec le broker pour acquérir les informations nécessaires. Cela permet de gérer des accès sécurisés à ces informations mais peut engendrer de nombreuses communications. On peut imaginer qu'un agent retienne les informations reçues et ne les re-demande pas au broker par la suite mais cette mémoire n'est possible que dans une sous-catégorie de SMA où les agents évoluent peu ou n'évoluent pas car il y a le risque que l'agent ait des informations obsolètes.

Un autre problème de la centralisation est qu'elle fragilise le système. Le broker est indispensable à l'ouverture du système mais aussi à la coopération entre agents. Dans les systèmes réels, il arrive que des agents disparaissent soudainement suite à un dysfonctionnement, et la robustesse d'un système se mesure à sa faculté à tolérer les défaillances d'agents. Le broker est un

point d'entrée vers les autres agents du système et sa perte empêcherait toute communication, ce qui en fait un élément critique du système. Le système est alors fragilisé et sensible aux défaillances. Dans une architecture hiérarchique, le dysfonctionnement d'un broker peut se révéler moins grave (seuls les agents indexés par celui-ci sont "perdus") mais reste préjudiciable.

Dans les chapitres suivants, nous présentons nos travaux sur la conception de systèmes ouverts en appliquant un principe de distribution de la représentation des autres et du service d'ouverture. Afin de limiter les communications entre agents, nous définissons un modèle minimal de représentation des autres permettant la coopération (chapitre 4) puis nous proposons des mécanismes d'ajout, de retrait et d'évolution des agents dans un cadre de conception de systèmes multi-agents ouverts (chapitre 5). En intégrant ces deux aspects aux agents d'un SMA, le rôle de broker est assuré collectivement par la société, et nous désignons les agents y participant par l'appellation *agent accueillant.*

# Deuxième partie

# Vers un modèle d'agent accueillant

# Chapitre 4

# Connaissances d'un agent accueillant

Notre proposition pour l'ouverture d'un SMA se positionne parmi les approches distribuées présentées dans le chapitre précédent. Les approches distribuées posent une contrainte sur le comportement des agents d'un SMA, dans le sens où, s'ils doivent participer à son ouverture, ils doivent effectuer quelques actions requises pour assurer l'ouverture globale. Par exemple, lors d'une présentation diffuse (cf. section 3.2.3), quand un agent reçoit une présentation d'un autre agent, il **doit** lui répondre s'il pense que sa réponse est intéressante.

Dans notre modèle, nous définissons ces contraintes comme une *facette* propre à un agent. Cette facette regroupe des connaissances et mécanismes de raisonnement assurant le comportement de l'agent auquel elle appartient, quand il a la charge d'une tâche liée à l'ouverture du SMA. De plus, un agent pouvant bénéficier de plusieurs facettes (on peut imaginer qu'un agent a un facette liée à la gestion de ces interactions, une facette de planification, ...), nous désignons celle attachée à l'ouverture, la *facette accueillante*, et par extension, un agent disposant de cette facette est appelé *agent accueillant.*

La facette accueillante d'un agent doit prendre partiellement en charge les tâches habituellement allouées à un broker dans un système. C'est par une coopération entre plusieurs de ces facettes appartenant chacune à un agent du système que ces tâches seront assurées. La facette accueillante des agents doit permettre l'intégration des agents mais aussi assurer le maintien de cette intégration suite à une évolution ou un départ d'un des agents. La première étape de notre travail consiste à définir ce que l'on entend par "intégration" d'un agent. Une définition précise de l'intégration est donnnée dans la section 5.1 car elle s'appuie sur des notions introduites dans notre modèle de description d'agents. Néanmoins nous pouvons déjà considérer qu'un agent, pour être intégré à une société d'agents, doit connaître :

- l'existence et les adresses d'autres agents de la société;

- leurs compétences pour savoir en quoi ils peuvent lui être utiles;
- leurs besoins en matière de compétences pour savoir en quoi il peut leur être utile.

Les deux derniers points peuvent se résumer à notre sens en une connaissance des possibilités de coopération avec les accointances d'un agent. A partir des connaissances nécessaires pour considérer un agent comme intégré, nous pouvons en déduire les tâches et conditions que doit remplir un agent accueillant. Il doit pouvoir :

- exprimer ses propres compétences et besoins aux autres agents;
- comprendre les compétences et besoins d'un autre agent;

pour être lui-même *intégré* et :

- connaître d'autres agents de la société pour les présenter au nouvel agent;
- avoir un comportement coopératif et bienveillant face à une demande d'intégration;

pour être *accueillant.*

Notre objectif dans la définition d'un modèle d'agent accueillant est de pouvoir considérer un système multi-agent comme ouvert dès lors qu'il est composé uniquement d'agents présentant cette facette accueillante. L'intégration d'un agent doit en tenir compte et maintenir l'ouverture du système. Ainsi chaque nouvel agent doit, à l'issue de son intégration, remplir les conditions nécessaires pour être lui aussi considéré comme accueillant et, en plus des trois points énoncés précédemment, acquérir la connaissance nécessaire à sa facette accueillante.

Dans ce chapitre, nous commençons par présenter une architecture d'agent accueillant puis nous détaillons les modules de connaissance sur les accointances.

## 4.1 Architecture générale d'agent accueillant

Nous avons regroupé la représentation des compétences et besoins des agents dans un module de *connaissance d'accueil* (ainsi que quelques autres connaissances que nous présentons au fur et à mesure de ce chapitre) et les processus de raisonnement concernant l'intégration de nouveaux agents dans un module de *comportement accueillant.* L'architecture générale présentée figure 4.1 positionne nos modules issus de la facette accueillante dans une architecture d'agent très générale. La définition précise d'une architecture d'agent sort du cadre de cette thèse et nous laissons la liberté à chacun

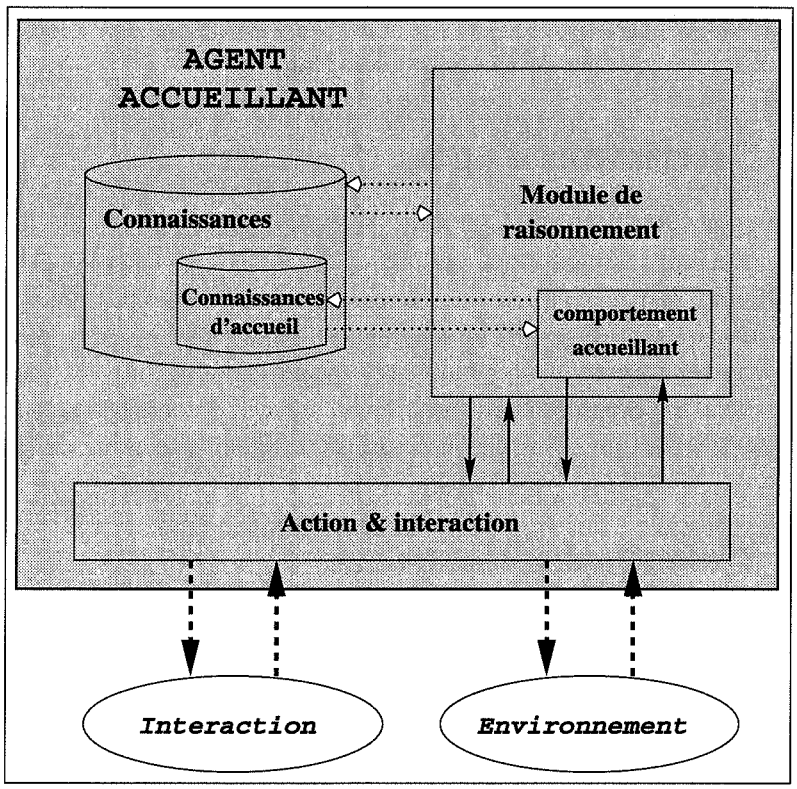


**Figure 4.1 :** Architecture générale d'un agent accueillant

de les détailler selon ses besoins (on peut trouver un état de l'art sur les architectures d'agent dans [Boissier, 2000]).

Les **connaissances d'accueil** d'un agent peuvent se diviser en deux parties. On y trouve d'abord la représentation des autres de l'agent accueillant. Celle-ci décrit plusieurs agents du système (y compris lui-même) au moyen d'une représentation regroupant les compétences et besoins d'un agent. La seconde partie consiste en un ensemble de connaissances sur les caractéristiques des agents en général. Par exemple, c'est ici qu'est exprimée la sémantique d'une action ou d'un plan auquels font références plusieurs représentations d'accointance. Nous appelons cette partie les *connaissances conceptuelles* d'un agent accueillant.

Le **comportement accueillant** définit un ensemble de tâches relatives à l'ouverture du SMA. Ce module implante le comportement accueillant de l'agent, c'est-à-dire les actions qu'il doit accomplir, les ressources qu'il doit utiliser, ... pour participer à l'ajout, le retrait ou l'évolution d'un agent. Ce module est détaillé dans le chapitre 5.

Dans ce chapitre, nous détaillons les connaissances d'accueil en commençant par les connaissances conceptuelles sur lesquels nous nous appuyons

pour définir la connaissance de soi et la représentation des autres, qui utilisent un même modèle de **description d'agent**.

## 4.2 Description d'agent

Une description d'agent est une représentation des compétences et besoins d'un agent. Nous définissons formellement ce que sont les compétences et les besoins respectivement dans les sections 4.2.2 et 4.2.3. Pour l'instant, considérons qu'une description d'agent est composée de deux parties : une description fonctionnelle décrivant ses compétences et buts et une description coopérative exprimant ses besoins en termes de compétences. Chacune de ces parties est une liste de descripteurs d'agent que nous définissons ci-dessous.

### 4.2.1 Définition des descripteurs

Pour représenter les compétences et besoins d'un agent nous devions définir des descripteurs exprimant les différentes fonctions qu'un agent peut effectuer ainsi que les combinaisons possibles de ces fonctions pour déduire les possibilités de coopération entre agents. Nous nous sommes inspirés de la description externe de [Sichman, 1995] (pour plus de détails voir la section 2.5.2) présentant cette orientation fonctionnelle et opérationnelle. Nous distinguons trois catégories de descripteurs: les buts, tâches et ressources.

La sémantique de ces descripteurs est définie en dehors des descriptions d'agent sous la forme de *classes de descripteurs*[1] . Un agent est alors décrit par un ensemble de classes de buts noté $\mathcal{G}$, un ensemble de classes de tâches noté $\mathcal{T}$ et un ensemble de classes de ressources noté $\mathcal{R}$. Ces ensembles sont organisés en une arborescence de telle sorte qu'une classe de descripteurs peut être une spécialisation d'une classe plus générale et peut être une généralisation de plusieurs classes plus spécifiques. La racine de chaque arborescence correspond à la notion générale de but, de tâche ou de ressource.

#### Buts

Un but est un état du monde qu'un agent veut atteindre ou satisfaire. C'est la motivation initiale qui pousse un agent à agir. L'ensemble $\mathcal{G}$ est composé de classes de buts, où une classe de buts $g_j^*$ est définie par :

- $id(g_j^*)$ son identifiant;
- $super(g_j^*)$ la classe de buts immédiatement plus générale;

[1] Dans nos notations formelles, nous utilisons une étoile ($*$) pour désigner une classe afin d'éviter la confusion avec l'instance utilisée dans une description d'agent. Ainsi $g_j^*$ est une classe de buts et $g_k$ une instance de classe de buts.

- $sub(g_j^*)$ les classes de buts immédiatement plus spécifiques.

**Tâches**

Une tâche est le moyen qu'utilise un agent pour influer sur son environnement et correspond à une ou plusieurs actions de cet agent. Chaque classe de tâches $t_j^*$ de l'ensemble $\mathcal{T}$ est définie par :

- $id(t_j^*)$ son identifiant;
- $super(t_j^*)$ la classe de tâches immédiatement plus générale;
- $sub(t_j^*)$ les classes de tâches immédiatement plus spécifiques;
- $uses(t_j^*)$ un ensemble de couples dont les deux éléments d'un couple $uses^k(t_j^*)$ sont :
  - $uses^{k,in}(t_j^*)$ un ensemble de classes de ressources utilisables par la tâche;
  - $uses^{k,out}(t_j^*)$ un ensemble de classes de ressources produites par la tâche en utilisant des ressources des classes spécifiées en première partie du couple correspondant.

  Chaque couple de cet ensemble représente une possibilité d'utilisation/production de ressources par la tâche concernée.

Si ces caractéristiques concernent toutes les classes de l'ensemble $\mathcal{T}$, cet ensemble est hétérogène et partitionné en trois sous-ensembles distincts : un sous-ensemble de classes d'actions noté $\mathcal{A}$, un sous-ensemble de classes de plans noté $\mathcal{P}$ et un sous-ensemble de classes de tâches génériques noté $\mathcal{GT}$ :

- Une classe de *tâches simples* ou classe d'*actions* représente un comportement actif indivisible d'un agent. C'est par le biais d'actions que les agents influent directement sur leur environnement.
- Une classe de *tâches complexes* ou classe de *plans* est une combinaison de plusieurs tâches génériques effectuées pour satisfaire un but donné. En cherchant à atteindre un but, un agent décide d'accomplir un plan l'y conduisant et d'exécuter toutes les tâches génériques spécifiées par ce plan. Cette combinaison de tâches génériques est un graphe orienté où les nœuds sont de type ET (pour paralléliser l'exécution des branches suivantes) ou OU (pour choisir une branche) et où les arcs représentent une tâche générique. Une classe de plans $p_j^*$ décrit deux caractéristiques supplémentaires :
  - $achieves(t_j^*)$ la classe de buts satisfaite par ce plan;

- $body(t_j^*)$ le graphe de classes de tâches génériques spécifiant les combinaisons de tâche à entreprendre et établissant certaines contraintes sur ces tâches (telles que l'utilisation de ressources produites en sortie d'une tâche précédente);

Nous avons défini les plans d'une manière très générale pour y faire principalement figurer des classes de tâches et une classe de buts qui sont utilisées lors du calcul des besoins d'un agent (cf. section 4.2.3). La structure d'un plan peut être spécialisée suivant les besoins d'une application en utilisant des formalismes existants de plans [Decker, 1996] [Wilkins and Myers, 1995].

- Une classe de *tâches génériques* représente une tâche en général et peut se spécialiser en actions et en plans.

L'arborescence de classes de tâches est un peu particulière par l'hétérogénéité de cet ensemble. Les particularités ou contraintes de cette arborescence sont:

1. Une classe de tâches génériques ne peut être généralisée que par une classe de tâches génériques mais peut se spécialiser en n'importe quelle classe de tâches;

2. Une classe de plans peut se généraliser en classe de plans ou en classe de tâches génériques mais ne peut se spécialiser qu'en classes de plans;

3. Une classe d'actions peut se généraliser en classe d'actions ou en classe de tâches génériques mais ne peut se spécialiser qu'en classes d'actions;

4. Un couple de ressources utilisables/productibles d'une tâche doit être une spécialisation d'un des couples de ses classes plus générales, au sens où toutes les classes de ressources du couple plus général doivent être présentes ou spécialisées dans la même partie du couple plus spécifique;

5. la racine de l'arborescence représentant une tâche en général (C'est une classe de tâches génériques notée *Tâche*) dispose d'un seul couple "ressources utilisables/productibles" tel que:
   $uses^{1,in}(Tâche) = \{Ressource+\}$ et
   $uses^{1,out}(Tâche) = \{Ressource+\}$ où *Ressource* est la racine de l'arborescence de ressources et où le + signifie qu'on peut utiliser aucune, une ou plusieurs instances de cette classe.

Nous avons défini pour chaque arborescence des opérateurs de comparaison entre deux classes testant si une classe est une généralisation ou une spécialisation d'une autre:

**Test de spécialisation.**

Soit deux classes $c$ et $d$, on note $c < d$ si $c$ est une spécialisation de $d$. C'est-à-dire :

$\exists x_1,\ldots,x_n \in \mathcal{G}$ (*respectivement* $\mathcal{T}$ *ou* $\mathcal{R}$)/
$x_1 = sub(d),\ldots,x_n = sub(x_{n-1}), c = sub(x_n)$

De même, l'opérateur $\leqslant$ teste si une classe est égale ou plus spécifique qu'une autre :

$c \leqslant d \equiv ((c < d) \vee c = d))$

**Test de généralisation.**

Soit deux classes $c$ et $d$, on note $c > d$ si $c$ est une généralisation de $d$. C'est-à-dire :

$\exists x_1,\ldots,x_n \in \mathcal{G}$ (*respectivement* $\mathcal{T}$ *ou* $\mathcal{R}$)/
$x_1 = super(d),\ldots,x_n = super(x_{n-1}), c = super(x_n)$

De même, l'opérateur $\geqslant$ teste si une classe est égale ou plus générale qu'une autre :

$c \geqslant d \equiv ((c > d) \vee c = d))$

La figure 4.2 donne un exemple d'une arborescence de classes de tâches dans un contexte de location de films dans un vidéo-club.

Les classes de tâches qui y sont représentées décrivent différentes fonctions associées au processus de location d'une vidéo. Ainsi, une classe de plans *Recherche_et_réserve_film* permet à un agent d'assister un utilisateur dans une location de film vidéo en structurant les autres classes de tâches représentées dans l'arborescence. Cette classe de plans demande l'exécution d'une tâche de classe *Saisie_requete* puis d'une autre de classe *Recherche_film* et finit par une tâche de classe *Réserve_film*. Quelques précisions sont à apporter sur ces tâches : (i) le plan précise des contraintes sur le passage des ressources entre tâches, ainsi la requête utilisée par la tâche *Recherche_film* doit être celle produite par la tâche *Saisie_requete* et la ressource de classe *Description_de_film* utilisée par la tâche de classe *Réserve_film* doit faire partie de l'ensemble de ressources produites par *Recherche_film*; (ii) chaque classe d'actions ou de plans est une spécialisation d'une classe de tâches génériques présentant les mêmes caractéristiques. Cela est nécessaire pour définir précisément ces classes dans le corps d'un plan.

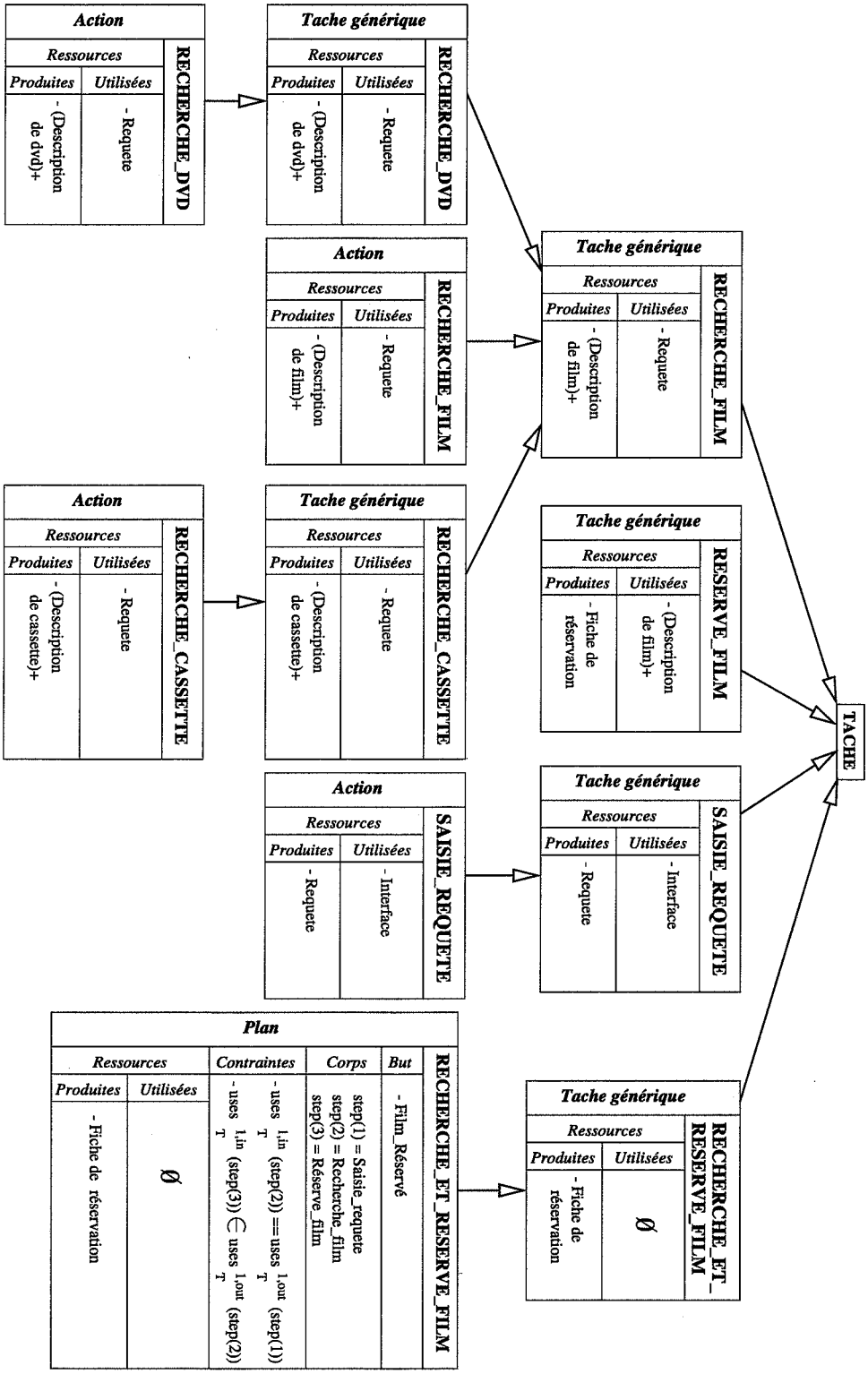


**Figure 4.2 :** Exemple d'une arborescence de classes de tâches

### Ressources

Une ressource est un objet manipulable par les agents. Une ressource est utilisée pour l'exécution de certaines tâches. Il faut distinguer deux types de classes de ressources :

- Les classes de ressources permanentes qui sont généralement des objets physiques ou informatiques de l'environnement.
- Les classes de ressources temporaires ou productibles qui n'ont qu'une existence éphémère. Elles sont produites par l'exécution d'une action et peuvent être consommées par d'autres actions.

Chaque classe de ressources $r_j^*$ de l'ensemble $\mathcal{R}$ est définie par :

- $id(r_j^*)$ son identifiant;
- $super(r_j^*)$ la classe de ressources immédiatement plus générale;
- $sub(r_j^*)$ les classes de ressources immédiatement plus spécifiques.

Ces descripteurs généraux définissent la sémantique des classes qui sont utilisées dans une description d'agent, dont nous détaillons maintenant le contenu.

## 4.2.2 Description fonctionnelle

La première partie d'une description d'agent représente les *compétences* et les *buts* de l'agent correspondant. Sous l'appellation "compétences", nous regroupons les plans, actions et ressources d'un agent. Chaque descripteur de cette *description fonctionnelle* fait référence à une des classes de descripteurs définies ci-dessus et peut être considéré comme une instance de la classe désignée.

Une description fonctionnelle $FD_k^l$ maintenue par un agent $l$ et représentant les compétences et buts d'un agent $k$ se note :

**Description fonctionnelle.**

$FD_k^l = (G_k^l, P_k^l, A_k^l, R_k^l)$

où $G_k^l$ est un ensemble de buts, $P_k^l$ est un ensemble de plans, $A_k^l$ est un ensemble d'actions, $R_k^l$ est un ensemble de ressources.

Un but $g \in G_k^l$ est décrit par :

- la classe de buts dont il est instance $class(g) \in \mathcal{G}$;
- l'importance qui lui est accordée par l'agent $k$ $weight(g)$.

Un plan $p \in P_k^l$ est décrit par :

- la classe de plans dont il est instance $class(p) \in \mathcal{P}$;
- l'efficacité avec laquelle il atteint ses buts $efficiency(p)$.

Une action $a \in A_k^l$ est décrite par :

- la classe d'actions dont elle est instance $class(a) \in \mathcal{A}$;
- l'efficacité avec laquelle l'agent $k$ accomplit cette action $efficiency(a)$.

Une ressource $r \in R_k^l$ est décrite par :

- la classe de ressources dont elle est instance $class(r) \in \mathcal{R}$;
- la qualité de la ressource à laquelle l'agent $k$ a accès $quality(r)$.

Les ressources présentes dans la description fonctionnelle d'un agent peuvent être permanentes ou productibles. Ainsi, un agent qui peut effectuer une action $a$ produisant une ressource $r$, est représenté par une description fonctionnelle contenant à la fois $a$ et $r$. Par contre, les ressources produites par les plans ne sont pas indiquées car elles sont réellement produites par des tâches dans ce plan, et il se peut qu'un agent connaissant un plan ne puisse pas accomplir toutes les tâches le composant.

Les mesures d'efficacité assignées aux compétences d'un agent et les poids assignés à ses buts sont utilisés pour choisir ses partenaires pour une coopération. Si un agent connaît plusieurs agents pouvant effectuer une action dont il a besoin, il choisira celui qui lui semble le plus efficace. Ces valeurs peuvent être communiquées avec le reste de la description ou elles peuvent être apprises à la suite de précédentes coopérations. Nous ne détaillons pas leur utilisation ou leur acquisition car elles interviennent pendant le raisonnement social d'un agent pour établir une coopération et non pendant l'intégration d'un agent et la reconnaissance du potentiel de coopération. Il nous semble néanmoins intéressant de les faire apparaître dans notre formalisme pour enrichir la description fonctionnelle et donner une esquisse de raisonnement social.

La description fonctionnelle d'un agent est utilisée conjointement à une description coopérative d'un autre agent pour reconnaître leur potentiel de coopération (dans quelle mesure le premier agent peut aider le second). Il faut donc garder à l'esprit son caractère descriptif et non actif. Cette remarque est importante quand on considère les buts d'un agent : un agent ne cherche pas forcément à atteindre ses buts au moment où est acquis une description fonctionnelle le représentant. La présence de buts dans la description d'agents signifie qu'il *peut* les activer et tenter de les satisfaire mais sans préciser si ce but est actif à l'instant présent. Cela devient plus

évident si l'on considère les compétences. Par exemple, un agent, disposant de plusieurs actions, n'est pas en train d'accomplir toutes ses actions. La description fonctionnelle représente le faît qu'il peut les effectuer.

Le tableau 4.1 illustre deux descriptions fonctionnelles dans l'exemple de location de vidéos.

**Table 4.1 :** Exemple de descriptions fonctionnelles

| **Agent Assistant** | |
|---|---|
| *Description fonctionnelle* | |
| Descripteur | Classe |
| But : $g_1$ | *Film_réservé* |
| Plan : $p_1$ | *Recherche_et_réserve_film* |
| Action : $a_1$ | *Saisie_requete* |
| Ressource : $r_1$ | *Requete* |
| Ressource : $r_2$ | *Interface* |
| **Vidéo-Club** | |
| *Description fonctionnelle* | |
| Descripteur | Classe |
| Plan : $p_2$ | *Réserve_film* |
| Action : $a_2$ | *Recherche_film* |
| Ressource : $r_3$ | *Fiche_de_réservation* |
| Ressource : $r_4$ | *Description_de_film* |

On se place du côté d'un agent assistant un utilisateur à réserver la location de films vidéo. Dans sa connaissance, l'agent assistant a un modèle de lui-même exprimant un but de la classe *Film_réservé* que l'utilisateur peut activer. Quand ce but est actif, l'agent cherche à le satisfaire en choisissant un plan l'atteignant. Il connaît un plan de la classe *Recherche_et_réserve_film* et peut effectuer, parmi les tâches spécifiées dans ce plan, une action de classe *Saisie_requete* lui permettant de recueillir une requête sur le type de films souhaité par son utilisateur. Cette action nécessite l'accès à une ressource de la classe *Interface* pour communiquer avec l'utilisateur et produit une ressource de la classe *Requete* qui sont toutes deux indiquées dans sa description fonctionnelle.

Pour atteindre son but, l'agent assistant devra interagir avec un autre agent représentant le vidéo-club. Pour l'instant nous nous contentons de représenter l'agent vidéo-club dans la connaissance de l'agent assistant par une description fonctionnelle contenant une action de la classe *Recherche_film* produisant des ressources de la classe *Description_de_film* et un plan de la classe *Réserve_film*. Il n'est pas nécessaire pour l'agent assistant de connaître le détail de ce plan. Il lui suffit de savoir que l'agent vidéo-

club le connaît et est le plus à même de réaliser les tâches spécifiées par ce plan ou de connaître d'autres agents pouvant les effectuer. Sa description fonctionnelle précise aussi qu'il peut produire une ressource de la classe *Fiche_de_réservation*.

Le potentiel de coopération entre ces deux agents commence à apparaître au travers de ces deux descriptions fonctionnelles. Nous l'explicitons dans une description coopérative.

### 4.2.3 Description coopérative

L'acquisition de la représentation qu'un agent a de lui-même est particulière. C'est la première description d'agent qu'il connaît (l'acquisition de descriptions d'agent sur les autres est détaillée chapitre 5) car elle doit être partiellement définie par le concepteur à la création de l'agent. La connaissance initale nécessaire à un agent pour déduire sa description complète est sa description fonctionnelle. L'agent peut alors raisonner sur ses propres compétences et buts pour savoir de quelles compétences il a *besoin*.

La notion de *besoin* est à la base de la description coopérative constituant la deuxième composante d'une description d'agent. En effet, certains descripteurs ne sont pas isolés dans leur définition et s'appuient sur l'utilisation d'autres descripteurs. Ainsi un plan permet d'atteindre un but par un ensemble de tâches, et une action utilise et produit des ressources. Ces relations font qu'une compétence (plan, action ou ressource) ou un but, que nous appellons $a$, d'un agent peut nécessiter l'emploi d'une autre compétence, que nous appellons $b$. Nous dirons alors qu'un agent dans cette situation a *besoin* de la compétence $b$ et que les descripteurs $a$ et $b$ sont *complémentaires*.

La description coopérative d'un agent regroupe tous ses besoins qui sont déduits des descripteurs de sa description fonctionnelle. Suivant les compétences concernées nous distinguons trois types de besoin : le besoin d'un plan, le besoin d'une tâche et le besoin d'une ressource. Une description coopérative $CD_k^l$ d'un agent $k$ connue d'un agent $l$ se note :

**Description coopérative.**

$CD_k^l = (PN_k^l, TN_k^l, RN_k^l)$

où $PN_k^l$ est l'ensemble de ses besoins de plan, $TN_k^l$ est l'ensemble de ses besoins de tâche et $RN_k^l$ est l'ensemble de ses besoins de ressource.

#### Besoin de plan

Un plan est le moyen fourni à un agent pour atteindre une classe de buts donnée. Ainsi, quand un agent a un but, il a besoin d'un plan lui permettant de le satisfaire. Le besoin en plan désigne une classe de buts pour signifier que l'agent représenté est intéressé par tout plan permettant de satisfaire

cette classe de buts. Un plan $p$ et un besoin en plan $pn$ sont *complémentaires* si la classe de buts satisfaite par $p$ est une spécialisation ou la même que celle désignée par $pn$. Il y a redondance d'informations entre les besoins en plan et les buts d'un agent mais nous tenons malgré tout à représenter ces deux ensembles pour leur sémantique différente (les buts sont fonctionnels alors que les besoins en plan sont une possibilité de coopération) et dans un souci de cohérence globale dans notre description d'agent. Un besoin en plan $pn \in PN_k^l$ est décrit par une classe de buts $class(pn) \in \mathcal{G}$.

### Besoin en Tâche

Le besoin en tâche désigne une classe de tâches génériques mais sera satisfait pendant une coopération par un plan ou une action. Les classes de tâches génériques ainsi désignées sont celles utilisées dans les plans connus de l'agent représenté. On considère donc qu'un agent connaissant un plan $p$ a besoin de toutes les tâches génériques présentes dans le graphe $body(p^*)$ associé à la classe de tâches $p^*$ de $p$. Une tâche $t$ et un besoin en tâche $tn$ sont *complémentaires* si la classe de tâches de $t$ est une spécialisation ou la même que celle désignée par $tn$. Un besoin en tâche $tn \in TN_k^l$ est décrit par une classe de tâches génériques $classGT(tn) \in \mathcal{GT}$.

### Besoin en ressource

L'utilisation directe d'une ressource est faite par une action. Un besoin en ressource, portant sur une classe de ressources, se déduit des ensembles de ressources utilisables par les actions de l'agent. Un agent pouvant effectuer une action $a$ a besoin de toutes les ressources présentes dans l'ensemble des ressources utilisables de la classe d'actions de $a$. Une ressource $r$ et un besoin en ressource $rn$ sont *complémentaires* si la classe de ressources de $r$ est une spécialisation ou la même que celle désignée par $rn$. Un besoin en ressource $rn \in RN_k^l$ est décrit par une classe de ressources $class(rn) \in \mathcal{R}$.

Nous avons défini deux prédicats dans lequels interviennent les besoins. Le premier exprime la complémentarité entre un besoin et une compétence alors que le second est vrai si deux besoins sont reliés par un lien de généralisation (un des besoins est plus général que l'autre).

**Complémentarité.**

La complémentarité entre une compétence $c$ et un besoin $n$ se note :

$$compl(c, n) \equiv class(c) \leqslant class(n)$$

**Prédicat related.**

Ce prédicat exprime que deux besoins $n$ et $n'$ sont reliés par un lien de généralisation. Il se note :

$$related(n, n') \equiv ((class(n) < class(n')) \vee (class(n) \geqslant class(n')))$$

La remarque concernant l'inactivité des descripteurs de la description fonctionnelle est également valable pour la description coopérative. Un besoin d'un agent ne signifie pas qu'il a un besoin immédiat de la compétence complémentaire. Ce besoin est actif quand la compétence ou le but ayant conduit à sa déduction sera activé.

On peut remarquer, dans notre définition des besoins, qu'un agent peut avoir besoin d'une compétence présente dans sa propre description fonctionnelle. Pour distinguer l'importance d'un besoin, nous utilisons le terme de *besoin fort* pour désigner un besoin d'un agent n'ayant pas la compétence complémentaire. Il nous est cependant nécessaire de définir, comme nous l'avons fait, les besoins d'un agent pour que celui-ci puisse connaître d'autres agents ayant la compétence concernée et étant peut-être plus efficaces dans l'utilisation de cette compétence.

La tableau 4.2 donne la description coopérative de l'agent assistant dans notre exemple de location de vidéo.

**Table 4.2 :** Exemple de description coopérative

| **Agent Assistant** | |
|---|---|
| *Description coopérative* | |
| Descripteur | Classe |
| Besoin en plan : $pn_1$ | *Film_réservé* |
| Besoin en tâche : $tn_1$ | *Saisie_requete* |
| Besoin en tâche : $tn_2$ | *Recherche_film* |
| Besoin en tâche : $tn_3$ | *Réserve_film* |
| Besoin en ressource : $rn_1$ | *Interface* |

Les besoins en plan regroupent les buts de l'agent assistant (*Film_réservé*). Il a ensuite déduit ses besoins en tâches à partir de son plan de classe *Recherche_et_réserve_film* qui consistent en l'ensemble des tâches spécifiées dans le corps de ce plan (*Saisie_requete*, *Recherche_film* et *Réserve_film*). Enfin, son besoin en ressource désignant la classe de ressources *Interface* est issue de son action de la classe *Saisie_requete*. L'agent assistant ne déduit pas la description coopérative de l'agent vidéo-club car l'agent vidéo-club est le mieux placé pour la déduire (l'agent assistant ne connaît qu'une partie de sa description fonctionnelle) et elle lui sera communiquée par la suite (voir

chapitre 5).

### 4.2.4 Propriétés de la description d'agent

La dualité de notre description d'agent ainsi que notre définition des descripteurs apportent certaines propriétés intéressantes à la représentation des autres pour des agents accueillants. La première de ces propriétés est la pertinence des informations. Il est souhaitable qu'un agent ne tente pas d'acquérir toutes les informations possibles sur les autres agents. En effet, des informations inutiles nécessiteraient des processus d'acquisition inutiles et pourraient ralentir le raisonnement de l'agent en traitant des cas sans intérêt. E. H. Durfee [Durfee, 1995] appelle cette limitation l'*ignorance bienheureuse* en arguant qu'il n'est pas nécessaire de tenter de s'approcher d'une représentation complète des autres et qu'il est même souhaitable de limiter celle-ci pour ne pas perdre du temps à raisonner sur des informations inutiles. Avant de définir un modèle de représentation des autres, il faut savoir quels sont les aspects intéressants à représenter. Dans notre exemple de locations de vidéo, il est important, pour l'agent assistant de savoir, si l'agent vidéo-club peut effectuer une réservation de cassettes mais il n'a aucun intérêt à connaître le nom de tous les employés du vidéo-club.

A. Haddadi [Haddadi and Sundermeyer, 1993] a défini deux critères de caractérisation de la représentation des autres : la *pertinence* et la *familiarité*. La pertinence s'attache à évaluer les possibilités d'interférence positives et négatives des autres, du point de vue de l'agent se les représentant. Une information est jugée pertinente si elle permet de déduire une situation de coopération ou de conflit entre l'agent représenté et l'agent doté de cette représentation. La familiarité est une notion plus vague caractérisant un degré qualitatif de connaissance sur les autres. Une faible familiarité consiste à connaître uniquement l'adresse d'un agent alors qu'une forte familiarité peut caractériser une représentation contenant des compétences, buts, intentions, états, ...

Dans une problématique de conception de systèmes, le niveau de familiarité des agents et la pertinence des informations sur les autres sont intimement liés. Le concepteur doit déterminer les champs de connaissance sur les autres nécessaires à un agent, pour définir leur niveau de familiarité. Le point central de notre approche est la détection locale du potentiel de coopération. La description d'agent, telle que nous l'avons définie, fournit à chaque agent une structure où est représenté ce potentiel. C'est dans ce sens qu'il faut considérer la description coopérative des agents. Elle n'apporte pas de nouvelles informations mais permet d'expliciter la demande d'un agent en matière de coopération. Comme nous l'avons souligné précédemment, la définition des concepts de but, plan et action nécessite l'emploi de certaines autres compétences entrainant un besoin sur cette autre compétence. La description coopérative est donc directement calculée à partir de la description

fonctionnelle mais elle représente une facette essentielle d'un agent car elle permet de détecter des complémentarités entre descripteurs.

Nous disposons maintenant, grâce aux descriptions d'agent, de plusieurs représentations, chacune étant associée à *un* autre et, grâce à la notion de complémentarité, d'informations sur les relations entre *deux* agents. Dans la prochaine section, nous introduisons des connaissances sur des *ensembles* d'agents, en nous inspirant de la construction de stéréotypes utilisée pour des profils utilisateurs (cf. section 2.2).

## 4.3 Stéréotypes d'agents

Pour justifier l'existence de stéréotypes dans la connaissance des agents, nous devons d'ores et déjà décrire quelques aspects du processus d'intégration des agents qui est détaillé dans le chapitre suivant. Nous suivons une approche totalement distribuée ne reposant pas sur une entité gérant l'ouverture du système. Le point d'entrée d'un agent souhaitant intégrer le SMA est un agent accueillant quelconque appartenant à ce SMA.

L'intégration d'un agent se fait progressivement par plusieurs présentations successives, chacune d'entre elles consistant en un protocole d'interaction impliquant deux agents pour effectuer un échange de leur propre description d'agent. Pour éviter toute confusion, nous appellons "hôte" l'agent appartenant au SMA qui remplit son rôle d'"accueil" pendant la présentation et "agent arrivant" celui qui est en cours d'intégration. Cet agent arrivant, pour s'intégrer au SMA, doit acquérir des descriptions d'agent représentant les agents avec qui il est susceptible de coopérer. Nous ne souhaitons pas qu'un agent arrivant contacte et se présente à tous les agents du système et il faut qu'il soit rapidemment dirigé vers des agents lui ressemblant pour "copier" leur intégration dans le SMA.

Cette redirection est une des tâches accueillantes qu'effectue l'agent hôte par une *recommandation* de ses accointances. Nous avons constaté, dans la section 2.2, que certaines recherches en représentation des utilisateurs utilisent également une technique de recommandation pour rapprocher des profils utilisateurs présentant des points communs et exploitent les informations de représentations existantes pour enrichir un nouveau profil. Nous adaptons la notion de stéréotypes à des agents pour recommander à un agent arrivant d'autres agents auxquels se présenter pour une intégration rapide. Le mécanisme de recommandation est décrit dans le chapitre 5.

Dans cette section, nous commençons par définir la notion de stéréotype d'agents. Nous nous intéressons ensuite à leur construction, que nous appelons le *stéréotypage*, par apprentissage. Enfin, nous concluons sur un point de vue plus général pour situer les stéréotypes dans la connaissance d'un agent accueillant et leur position par rapport aux descriptions d'agents.

### 4.3.1 Définitions

Dans la section 4.2, nous avons distingué descripteurs et classes de descripteurs. On peut considérer que la même relation existe entre la représentation des autres et les stéréotypes. *Un stéréotype est une classe décrite par une description d'agent qui est un sous-ensemble ou une généralisation de chaque description d'agent des accointances ou stéréotypes appartenant à ce stéréotype.* Deux idées sont importantes dans cette définition :

- L'ensemble des stéréotypes connus par un agent est organisé en hiérarchie. Un stéréotype peut être lié à un stéréotype plus général et peut caractériser plusieurs stéréotypes plus spécifiques.
- Un stéréotype contient une description d'agent exprimant une partie de la sémantique de ce stéréotype (l'autre partie est représentée par sa position dans la hiérarchie). Ainsi, un stéréotype définit un agent abstrait ayant des buts, plans, actions, ressources et des besoins en plan, tâche et ressource qui sont des descripteurs caractérisant tout stéréotype ou représentation d'un autre spécialisant ce stéréotype. Chaque descripteur d'un stéréotype doit être présent ou spécialisé dans les descriptions d'agents plus spécifiques.

Nous pouvons maintenant adopter un point de vue plus général sur la connaissance sociale d'un agent accueillant. Il connaît un ensemble d'agents du système (ses accointances) et un ensemble de stéréotypes. Pour distinguer une représentation dédiée à un agent du système et la structure regroupant des descripteurs et caractérisant une accointance ou un stéréotype, nous utilisons le terme *représentation d'accointance* dans le premier cas et *description d'agent* dans le second.

Ainsi, la connaissance sociale d'un agent accueillant $l$ se note $SK^l = (AR^l, ST^l)$ où $AR^l$ est un ensemble de représentations d'accointance et $ST^l$ un ensemble de stéréotypes. Chaque représentation d'accointance $AR^l_k \in AR$ désignant un agent d'identifiant $k$ est décrite par :

- l'adresse de l'agent représenté $addr^l_k$;
- la description d'agent le caractérisant, notée $description(AR^l_k)$;
- l'ensemble des stéréotypes auxquels il appartient, noté $stéréotypes(AR^l_k) \subset ST^l$.

Chaque stéréotype $ST^l_k \in ST^l$ est décrit par :

- une description d'agent $description(ST^l_k)$;
- l'ensemble des stéréotypes auxquels il appartient, noté $super(ST^l_k) \subset ST^l$;

- l'ensemble des stéréotypes qu'il décrit, noté $sub(ST_k^l) \subset SK$.

Un agent accueillant construit dynamiquement des stéréotypes à partir de l'ensemble de ses représentations d'accointance. Notre SMA étant ouvert, cette construction est incrémentale, c'est-à-dire qu'à chaque présentation d'un agent arrivant, de nouveaux stéréotypes apparaissent et s'insèrent dans la hiérarchie sans la remettre en cause. La figure 4.3 illustre les conséquences d'une présentation sur la connaissance sociale d'un agent accueillant.

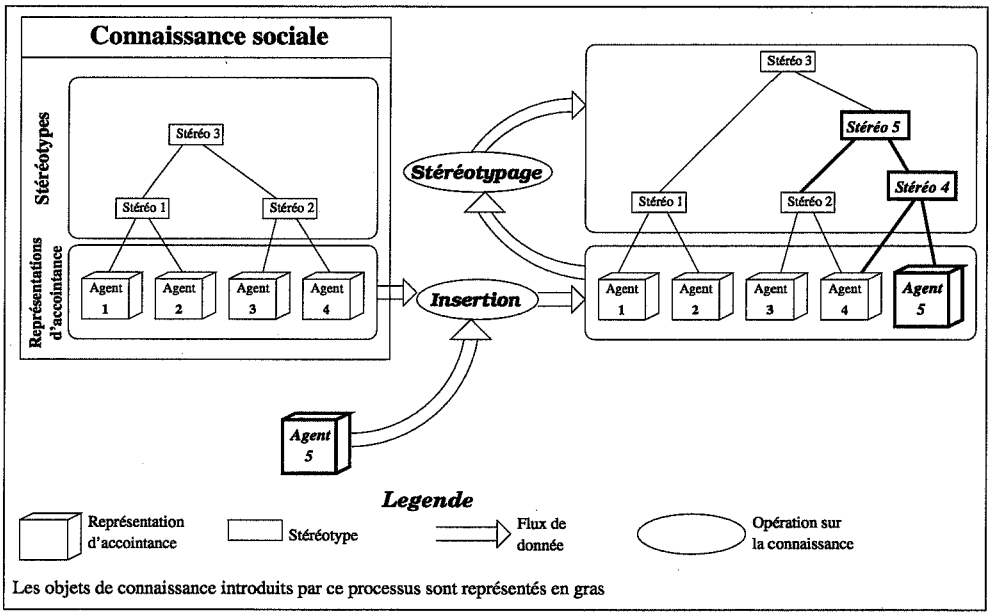


**Figure 4.3 :** Processus d'insertion de connaissances sociales

La première étape est un simple ajout. La nouvelle représentation d'accointance s'insère à la suite de la liste de représentations d'accointance déjà connues. Cette insertion déclenche la deuxième étape qui consiste à mettre à jour les stéréotypes pour prendre en compte les nouvelles informations apportées par cette représentation d'accointance. Le stéréotypage produit des nouveaux stéréotypes et enrichit la hiérarchie existante de stéréotypes. Nous détaillons maintenant ce processus de stéréotypage.

### 4.3.2 Stéréotypage

Le stéréotypage consiste en la définition de classes recouvrant chacune un sous-ensemble de représentations d'accointance. Il existe de nombreux travaux en apprentissage automatique, et plus particulièrement en classification automatique, traitant ce genre de problèmes.

### Classification automatique et stéréotypage

Les différents algorithmes de classification existants se distinguent par la structure et la qualité des données prises en entrée et le résultat souhaité en sortie de l'algorithme. Ainsi, certains algorithmes créent des arbres de décisions (ID3 [Quinlan, 1986]), d'autres des hiérarchies probabilistes de concepts (COBWEB [Fisher, 1987]), ...P. Langley [Langley, 1996] fournit un état de l'art détaillé des différentes approches de classification.

L'ensemble de stéréotypes connus d'un agent accueillant est le fruit d'un algorithme incrémental de classification non-supervisée produisant une hiérarchie de concepts. L'*incrémentalité* de l'algorithme nous est indispensable car il n'est pas question de reconstruire toute la hiérarchie de stéréotypes à chaque introduction de représentation d'accointance. Cette hiérarchie est uniquement mise à jour pour prendre en compte cette description. La mise à jour doit également se faire de manière automatique par l'agent accueillant sans intervention de l'utilisateur, d'où l'utilisation d'une classification *non-supervisée*. Enfin, les stéréotypes doivent être organisés *hiérarchiquement* pour représenter différents niveaux de généralisation d'agents.

Cette problématique de formation de hiérarchie de concepts est partagée par le système UNIMEM [Lebowitz, 1986]. La formation de telles hiérarchies est traité par une Mémoire Fondée sur la Généralisation (GBM).

### Mémoire Fondée sur la Généralisation

Le principe de la Mémoire Fondée sur la Généralisation est de généraliser une situation décrite par un petit nombre d'exemples en un ensemble de *concepts*, puis de généraliser à nouveau ces concepts en d'autres concepts. Les exemples de départ sont considérés comme des *instances*. Chaque instance est décrite par un ensemble de propriétés (des paires propriété/valeur) qui est utilisé pour décrire les concepts la généralisant.

La formation de hiérarchie est incrémentale, permettant ainsi d'introduire de nouvelles instances et de construire de nouveaux concepts. Lors de l'introduction d'une nouvelle instance, on recherche d'abord les concepts généralisant cette nouvelle instance. On compare ensuite cette instance aux concepts et instances plus spécifiques que ceux trouvés précédemment, ce qui peut amener la construction de nouveaux concepts généralisant les deux éléments comparés. Le détail de l'algorithme, adapté au stéréotypage, est fourni dans l'annexe 8.3.

### Exemple

Nous donnons par la figure 4.4 un exemple de stéréotype, connu de l'agent assistant, recouvrant sa propre représentation et celle d'un autre agent appelé *gestion de profil*. Pour cet exemple, nous définissons ce nouvel agent avec pour compétences un plan de la classe de tâches *Saisie_requete*

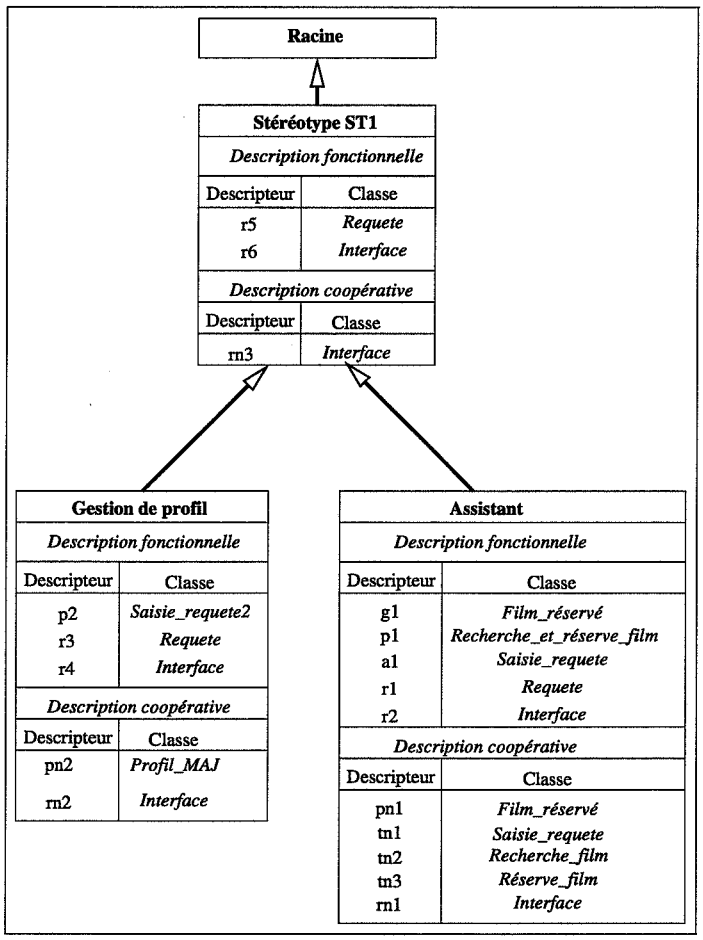


**Figure 4.4 :** Stéréotypes de l'agent connus de l'agent assistant

en considérant ce plan comme une alternative à l'action correspondante de l'agent assistant, dont le corps consiste en une saisie effective d'une requête et une action apprenant un profil de l'utilisateur d'après les requêtes qu'il émet. L'agent assistant n'a pas besoin de connaître le détail de ce plan, ni d'autres compétences que celles représentées car elles ne lui sont pas pertinentes. Précisons néanmoins que cet agent peut produire une requête à l'aide de son profil utilisateur (remplaçant l'action $Saisie\_requete$). On peut remarquer dans le stéréotype $ST1$ que l'action $a1$ de l'agent assistant et le plan $p2$ de l'agent de gestion de profil sont généralisés en une tâche générique $t1$.

En l'état actuel du système, le nouveau service de l'agent de gestion de profil ne peut pas être utilisé car il n'existe aucune compétence dans le système utilisant ce profil utilisateur. Il faut introduire un nouvel agent assistant prenant en compte ce profil. Une nouvelle compétence est introduite sous la forme d'un plan spécialisant la classe $Recherche\_et\_réserve\_film$. Le corps

de ce plan spécifie trois tâches de classes : *Recherche_film*, *Saisie_confirmation* et *Réserve_film*. Pour réserver un film, l'agent *assistant 2* nécessite une requête (l'origine de cette requête est laissée libre) pour lancer une recherche de film, demander une confirmation à l'utilisateur et réserver le film. La figure 4.5 représente la nouvelle connaissance de l'agent assistant avec l'inclusion d'une représentation de l'agent assistant 2 et la création d'un nouveau stéréotype.

Le stéréotype $ST2$ est l'intersection des descriptions d'agents assistant et assistant 2, et le stéréotype $ST1$ devient une généralisation du stéréotype $ST2$ et de l'agent *Gestion de profil*. Il n'existe pas de stéréotypes englobant l'agent de gestion de profil à un seul autre car il n'existe pas de généralisation de descriptions plus spécifique que $ST1$.

### 4.3.3 Propriétés des stéréotypes

Les stéréotypes connus d'un agent accueillant lui sont d'un intérêt variable. En effet, s'il généralise dans un stéréotype peu de descripteurs communs à peu d'agents, le stéréotype n'est pas forcément représentatif. Pour estimer la pertinence d'un stéréotype, nous définissons deux indicateurs : la couverture et la spécificité d'un stéréotype.

#### Couverture

La couverture d'un stéréotype est la proportion de représentations d'accointances appartenant à ce stéréotype. Soit un stéréotype $ST_k^l \in ST^l$ et $AR(ST_k^l)$ un ensemble de représentation d'accointances telles que pour tout $AR_i \in AR(ST_k^l)$ :

$$\exists ST_1, \ldots, ST_n \in \mathcal{ST} / \\ ST_1 = sub(ST_k^l), \ldots, ST_n = sub(ST_{n-1}), AR_i = sub(ST_n)$$

La couverture de $ST_k^l$ est le cardinal de l'ensemble $AR(ST_k^l)$ divisé par le cardinal de l'ensemble des représentations d'accointance de l'agent : $couverture(ST_k^l) = \frac{|AR(ST_k^l)|}{|AR|}$.

#### Spécificité

La spécificité d'un stéréotype correspond à son potentiel de discrimination. La formation de hiérarchie par une Mémoire Fondée sur la Généralisation attribue une valeur à chaque descripteur d'un concept (d'un stéréotype dans notre cas). Cette valeur (initialement 0) est incrémentée dès qu'une nouvelle instance (représentation d'accointance dans notre cas) contient ce descripteur. Par contre, si une instance ne peut pas être classée dans un stéréotype à cause de cette valeur (l'instance est classée dans un stéréotype directement plus général) cette valeur est décrémentée. Plus cette valeur est

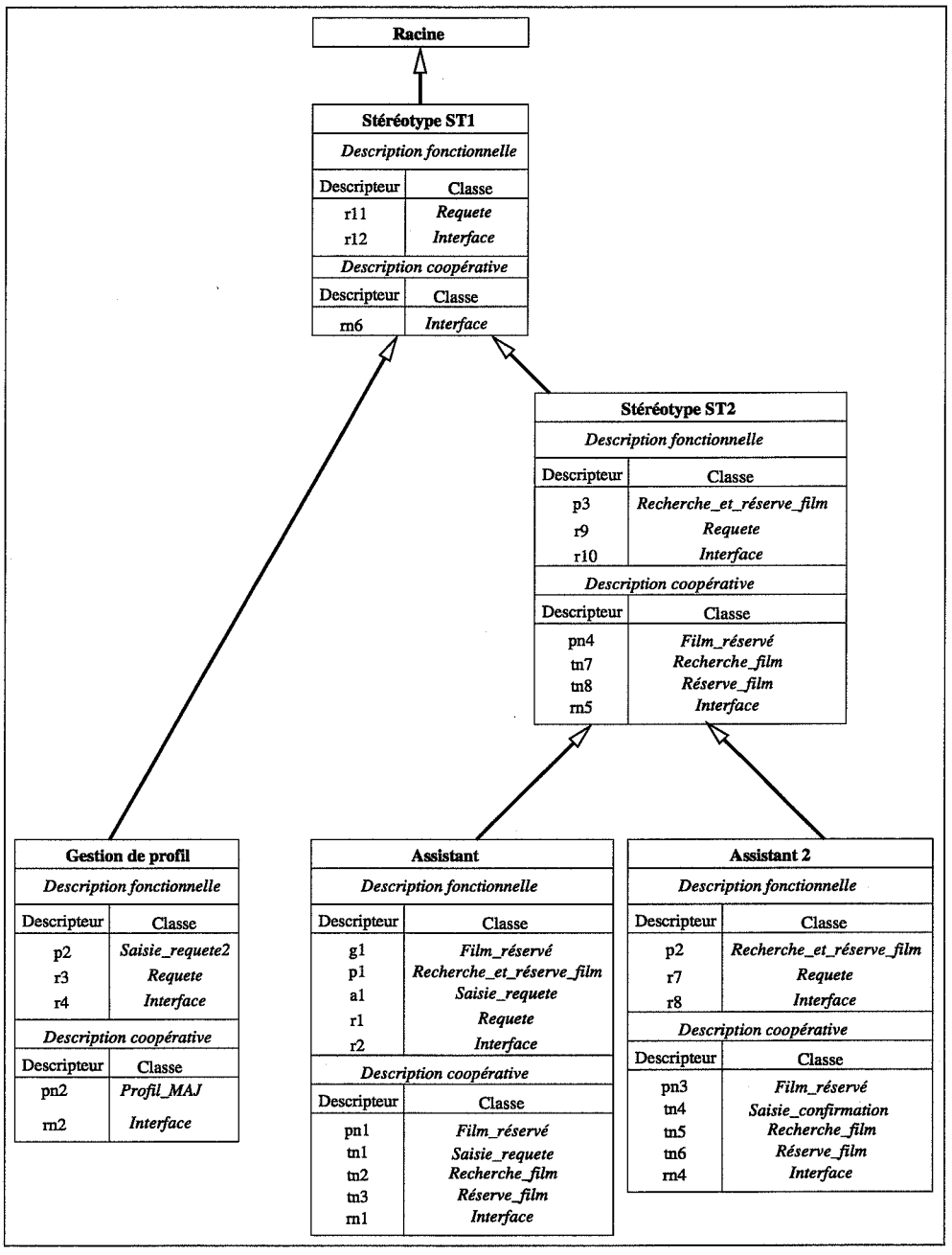


**Figure 4.5 :** Stéréotypage du deuxième assistant

faible et plus le descripteur de ce stéréotype est discriminant. Le potentiel de discrimination du stéréotype correspond à la moyenne des valeurs de ses descripteurs et se note : $spécificité(ST_k^l)$.

### Pertinence

Muni de ces deux indicateurs, on peut évaluer la pertinence d'un stéréotype. cette pertinence est utilisée dans la section 5.4.2. Nous estimons qu'un stéréotype est pertinent s'il couvre beaucoup d'agents et s'il est très discriminant. Comme un stéréotype couvre beaucoup d'agents quand sa valeur de couverture est élevée et qu'il est discriminant quand sa valeur de spécificité est faible, la pertinence est calculée par la différence de ces deux valeurs :

$$pertinence(ST_k^l) = couverture(ST_k^l) - spécificité(ST_k^l)$$

## 4.4 Synthèse

Nous avons défini dans ce chapitre la composition de la connaissance sociale d'un agent accueillant. La figure 4.6 récapitule les différents éléments introduits. Nous tenons à insister sur le fait que cette connaissance sociale est partielle. En effet, elle est à la base d'un agent accueillant et la propriété "accueillante" d'un agent n'est qu'une facette qui peut co-exister avec d'autres facettes (comme certaines propriétés de perception ou des facettes organisationnelles lui définissant des rôles, ...) de cet agent. Ces autres facettes nécessitent d'autres types de connaissances venant compléter nos connaissances sociales.

En dehors de la connaissance sociale, nous avons défini des ensembles de classes de descripteurs que nous englobons dans une *connaissance conceptuelle* de l'agent. Les différentes compétences et les différents besoins utilisés pour décrire un agent y sont représentés de manière générale permettant ainsi à un agent accueillant de raisonner sur plusieurs descriptions d'agent et d'en déduire des relations ou dépendances entre descripteurs et donc entre agents. Les stéréotypes sont également des classes mais qui décrivent des objets de connaissance plus complexes : les descriptions d'agent.

La plupart des champs de connaissance que nous avons définis dans ce chapitre l'ont été de manière statique. Seul un algorithme de construction de stéréotype a été décrit permettant de mettre à jour l'ensemble de stéréotypes connus d'un agent accueillant. Néanmoins, la connaissance conceptuelle et les représentations d'accointance sont eux aussi amenées à évoluer, surtout dans un SMA ouvert. L'ajout ou le retrait d'un agent doit être répercuté dans les représentations des autres des agents du système, de même qu'une modification des compétences et besoins d'un agent doit être pris en compte par des mises à jour des représentations d'accointance correspondantes. De plus, un SMA ouvert doit permettre l'ajout de nouvelles fonctionnalités via

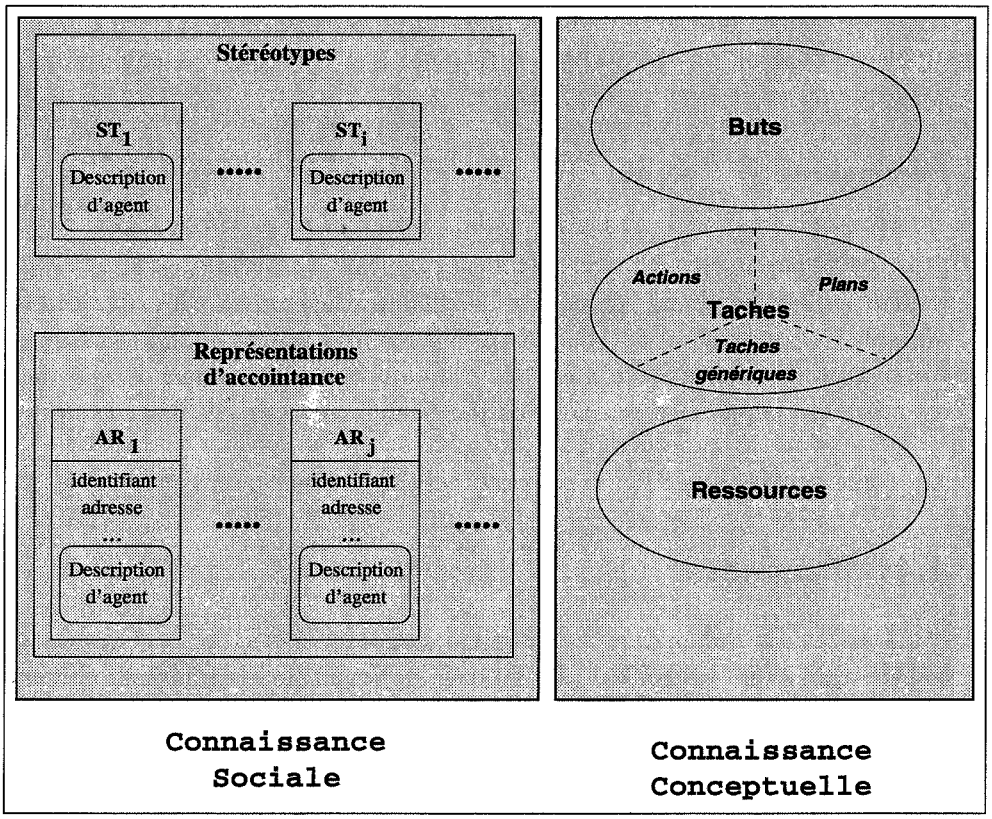


**Figure 4.6 :** Composition de la connaissance d'accueil

des compétences qui n'ont pas été initialement décrites dans la connaissance conceptuelle des agents. Il faut également définir des mécanismes de mise à jour de cette connaissance. Nous traitons ces problèmes dans le prochain chapitre en détaillant les processus d'intégration, de retrait et d'évolution des agents et la propagation des modifications dans la connaissance des agents.

# Chapitre 5

# Comportements accueillants

Un système multi-agent ouvert doit permettre l'ajout de nouveaux agents ainsi que le retrait et l'évolution d'agents du système. Dans le chapitre 3, nous avons constaté que ces trois tâches étaient gérées conjointement et assignées soit à une entité dédiée à l'ouverture du système, soit à l'agent concerné par l'ajout, le retrait ou l'évolution.

Notre approche consistant à partager la responsabilité des tâches d'ouverture entre des agents accueillants, nous a naturellement conduit à définir une activité coopérative de plusieurs facettes accueillantes pour remplir ces tâches. Les agents accueillants doivent bénéficier de compétences liées à leur module accueillant. Pour chaque tâche relative à l'ouverture du SMA, nous avons défini des classes de descripteurs nécessaires à la réalisation d'une de ces tâches. Nous supposons donc qu'un agent accueillant a des compétences instanciant ces classes pour assurer ses fonctions accueillantes. Au début de chaque section consacrée à une tâche liée à l'ouverture, nous présentons le plan qui y est associé et décrivons ses différentes étapes.

Nous commençons, dans ce chapitre, par définir la notion d'intégration d'un agent dans le SMA. Ensuite, nous détaillons le processus d'ajout d'agents composé d'une phase de description, d'une phase de présentation et d'une phase de recommandation. La gestion du retrait et de l'évolution d'agents est expliquée en fin de chapitre.

## 5.1 Intégration d'un agent dans un SMA

L'intégration d'un agent dans un SMA va plus loin que son simple ajout physique. Un agent doit être capable d'établir des coopérations avec d'autres agents du système ou de répondre à des propositions de coopération. Avec notre modèle d'agent accueillant, un agent peut détecter ses possibilités de coopération en confrontant sa propre description fonctionnelle aux descriptions coopératives de ses accointances et inversement. En considérant qu'un agent est intégré s'il peut coopérer avec certains agents du système, l'intégra-

tion d'un agent peut être vue comme une situation caractérisant les relations entre sa représentation de lui-même et sa représentation des autres.

Un des problèmes qui se posent dès lors que l'on met en relation plusieurs représentations issues d'agents différents est celui de l'ontologie de chaque agent. En effet, il se peut que les agents utilisent des arborescences de classes différentes pour se décrire et que celles-ci ne soient pas appropriées pour décrire un autre agent. La gestion d'ontologies différentes est un problème vaste dépassant même le domaine des Systèmes Multi-Agents. Aussi, pour centrer notre travail sur l'ouverture, nous posons l'hypothèse que les agents utilisent une ontologie commune pour se décrire et représenter d'autres agents. Dans le cadre de la conception de systèmes multi-agents, cette hypothèse est peu contraignante car on peut imaginer qu'un *concepteur* opérant la création et l'ajout physique d'un agent dans le SMA soit le garant du maintien d'une ontologie commune. Il serait néanmoins intéressant d'étendre à l'avenir nos travaux pour lever cette hypothèse et permettre la présence d'agents ayant des ontologies différentes au sein d'un même SMA (par exemple en dotant les agents de capacités d'apprentissage sur les ontologies des autres ou par des techniques de une construction collective d'une ontologie commune à partir de plusieurs ontologies).

Nous distinguons deux types d'intégration : l'intégration totale et l'intégration partielle (qui sont définis ci-dessous). Notre travail s'est focalisé sur l'intégration totale mais nous évoquons également les particularités d'une intégration partielle. Aussi, lorsque nous utilisons les termes "intégration" ou "agent intégré" sans préciser le type de cette intégration, nous faisons référence à une intégration totale.

### 5.1.1 Intégration totale

L'intégration d'un agent se définit formellement d'un point de vue global. C'est-à-dire que nous référençons la connaissance de plusieurs agents plutôt que de nous placer dans la connaissance d'un seul agent, ce qui en fait une considération sur l'ensemble du SMA. Pour être intégré totalement, la représentation des autres d'un agent accueillant doit respecter deux contraintes :

- Il doit connaître tous les agents ayant besoin d'une de ses compétences ou dont il a besoin d'une compétence et, pour chacun de ces agents, il doit connaître tous leurs descripteurs présentant cette complémentarité.

- Pour le bon déroulement de l'intégration des autres agents, un agent doit connaître tous les agents ayant un besoin désignant une classe située sur une même branche de l'arborescence de la classe désignée par un de ses besoins. C'est-à-dire qu'un agent connaît tous les agents ayant un besoin égal, plus général ou plus spécifique qu'un de ses besoins. Comme pour la première contrainte, il connaît, à propos de

ces agents, tous leurs besoins concernés par cette contrainte. Cette connaissance est justifiée dans les sections 5.4.1 et 5.6.3.

Soit $\Omega = \{\omega_1, \ldots, \omega_i, \ldots\}$ l'ensemble des agents compris dans un SMA, tels que $\omega_i$ est un agent dont l'identifiant est $i$. Un agent $\omega_i$ est coopérativement intégré si sa description coopérative respecte les contraintes ci-dessus et fonctionnellement intégré si sa description fonctionnelle respecte ces contraintes. Formellement cela se traduit par :

**Intégration Coopérative.**

Un agent $\omega_i$ est dit *coopérativement intégré* si et seulement si :

$\forall \omega_j \in \Omega, \forall n \in CD_i^i, \forall c \in FD_j^j$
$compl(c, n) \supset c \in FD_j^i$

et $\forall \omega_j \in \Omega, \forall m \in CD_i^i, \forall d \in CD_j^j$
$related(d, m) \supset d \in CD_j^i$

**Intégration Fonctionnelle.**

$\omega_i$ est dit *fonctionnellement intégré* si et seulement si :

$\forall \omega_j \in \Omega, \forall c \in FD_i^i, \forall n \in CD_j^j$
$compl(c, n) \supset n \in CD_j^i$

**Pertinence.**

La représentation qu'a un agent $\omega_i$ d'un agent $\omega_j$ est dite *pertinente* si et seulement si :

$\forall n \in CD_i^i, \forall c \in FD_j^j, compl(c, n) \equiv c \in FD_j^i$ et

$\forall c \in FD_i^i, \forall n \in CD_i^i, \forall n' \in CD_j^j,$
$(compl(c, n') \vee related(n, n')) \equiv n \in CD_j^i$

L'agent $\omega_i$ est intégré à un SMA s'il est fonctionnellement et coopérativement intégré. En posant certaines hypothèses, il est possible de transcrire cette définition au niveau local à un agent pour lui permettre de rechercher son intégration. Nous décrivons ces hypothèses dans la section 5.4.1.

### 5.1.2 Intégration partielle

Dans certains contextes, il est très difficile, voire impossible, d'intégrer totalement les agents du SMA. Pour cela nous avons défini l'intégration partielle dont les contraintes sur la représentation des autres sont :

- Pour chacune de ses compétences, un agent doit connaître au moins un agent en ayant besoin et, pour chacun de ses besoins, il doit connaître au moins un agent ayant une compétence complémentaire, si un tel

agent existe. Il doit également connaître ce descripteur complémentaire chez cet agent.

- Pour chacun de ses besoins, un agent doit connaître au moins un agent ayant un besoin plus général que celui-ci, si un tel agent existe. Il connaît également ce besoin chez cet agent.

Un agent $\omega_i$ est coopérativement partiellement intégré si sa description coopérative respecte les contraintes ci-dessus et fonctionnellement partiellement intégré si sa description fonctionnelle respecte ces contraintes. Formellement cela se traduit par :

**Intégration Coopérative Partielle.**

Un agent $\omega_i$ est dit *coopérativement partiellement intégré* si et seulement si :

$\forall \omega_j \in \Omega, \forall n \in CD_i^i, \forall c \in FD_j^j$
$compl(c,n) \supset \exists FD_k^i \in SK^i / \exists c' \in FD_k^i / compl(c',n)$
et $\forall \omega_j \in \Omega, \forall m \in CD_i^i, \forall d \in CD_j^j$
$class(d) > class(m) \supset \exists CD_j^i \in SK^i / \exists m' \in CD_j^i / class(d) > class(m')$

**Intégration Fonctionnelle Partielle.**

$\omega_i$ est dit *fonctionnellement partiellement intégré* si et seulement si :

$\forall \omega_j \in \Omega, \forall c \in FD_i^i, \forall n \in CD_j^j$
$compl(c,n) \supset \exists CD_k^i \in SK^i / \exists n' \in CD_k^i / compl(c,n')$

Une intégration partielle nécessite qu'un agent connaisse, pour chacun de ses besoins, au moins un agent pouvant y répondre si un tel agent existe, et, pour chacune de ses compétences, au moins un agent en ayant besoin si un tel agent existe. Dans la suite de ce chapitre, nous considérons des intégrations totales. Nous reviendrons sur l'intégration partielle dans le chapitre 6.

## 5.2 Ajout et présentation d'un agent

La figure 3.6 présentée à la fin du chapitre 3 distingue trois étapes dans le processus d'ajout d'un agent :

1. **La création de l'agent** : outre l'implémentation des fonctions proposées par l'agent, il faut représenter ses compétences dans une description fonctionnelle (phase descriptive). Dans le cas de migrations d'agents existants (issus d'un autre SMA par exemple), seule la construction d'une description est nécessaire en utilisant les ontologies de buts, tâches et ressources en vigueur dans le SMA qu'il doit intégrer.

2. **L'ajout physique** : l'agent doit être reconnu comme appartenant au SMA. Cette appartenance dépend de la plate-forme servant de support au SMA (elle peut requérir une inscription de l'agent, par exemple).

3. **L'intégration** : pendant cette étape, l'agent arrivant doit mettre à jour sa représentation des autres pour y inclure les agents du SMA avec lesquels il peut coopérer. De la même manière la représentation des autres de certains agents du SMA doit incorporer une représentation de l'agent arrivant.

Ces étapes sont générales et s'appliquent à tout SMA ouvert. Pour des agents accueillants, la création consiste en une saisie de leur description fonctionnelle et en un calcul de leur description coopérative. Nous n'avons pas représenté la partie d'implémentation de l'agent ni son ajout physique car ce sont deux aspects qui relèvent d'un problème de programmation, d'une application particulière ou de la plate-forme utilisée ce qui sort de l'objectif de ce chapitre.

### 5.2.1 Phase descriptive

Pour définir la description fonctionnelle d'un agent, le concepteur représente ses fonctions sous la forme d'actions et les objets qu'il peut utiliser sous la forme de ressources. Les plans sont des descripteurs de plus haut niveau faisant le lien entre des buts et l'invocation de différentes actions.

La description fonctionnelle doit être exprimée en fonction des ontologies de buts, tâches et ressources utilisées par les autres agents du système. Même si un agent peut n'avoir qu'une connaissance partielle de ces ontologies, la description de nouveaux buts, plans, ... peut nécessiter l'enrichissement de certaines arborescences (l'ajout de nouvelles branches) mais doit respecter les classes existantes. Nous expliquons dans la section 5.6.3 dans quelles mesures ces arborescences peuvent être modifiées.

A partir de sa description fonctionnelle, l'agent complète sa représentation de lui-même en déduisant ses besoins pour se créer une description coopérative. Le processus d'ajout d'un agent ne nécessite que deux interventions du concepteur : la saisie de sa description fonctionnelle et celle de l'adresse d'un agent quelconque du SMA. Cette adresse, constituant le point d'entrée de l'agent dans le SMA, est indispensable pour qu'il puisse initier une première présentation (on peut considérer la saisie de cette adresse comme l'ajout physique de l'agent au SMA).

### 5.2.2 Phase d'intégration

L'intégration d'un agent accueillant suit un plan dont le corps est représenté par la figure 5.1. Un agent à intégrer au SMA active un but de la classe *Intégré* puis son plan de la classe *Intégration*.

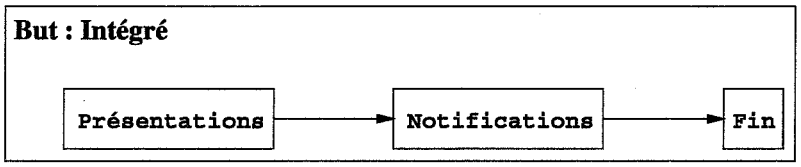


**Figure 5.1** : Plan d'intégration d'un agent

Nous avons décomposé l'intégration d'un agent en deux phases :

- Une phase de **présentations** pendant laquelle le nouvel agent déroule successivement plusieurs protocoles de présentations avec pour objectif de se construire une représentation de ces autres et de leur fournir une représentation de lui-même.

- Quand l'agent n'aura plus besoin de présentations, il effectuera des **notifications** pour informer d'autres agents de son existence et de certaines de ses compétences ou de ses besoins. Il choisira les agents à qui envoyer une notification en fonction du résultat des présentations (les agents auxquels il se présente lui recommandent de contacter d'autres agents).

## 5.3 Le procotole de présentation

Pour être intégré au système, l'agent arrivant doit fournir des informations le décrivant à certains agents du système et acquérir des informations les décrivant. Cet échange est représenté par un plan de la classe *Présentations* représenté en figure 5.2.

Cette succession de tâches fournit aux deux agents participant à un protocole de présentation une représentation de l'autre. Nous décrivons chacune de ses actions ci-dessous (pour rappel, nous appelons "agent arrivant" l'agent à intégrer et "agent hôte" son interlocuteur présent dans le SMA) :

- **Etape 1 : Envoi de descripteurs.** L'agent arrivant envoie un message à l'agent hôte contenant des descripteurs de sa description coopérative. Lors de la première présentation, il envoie tous les descripteurs de sa description coopérative et il n'en sélectionnera qu'une partie pour les présentations suivantes.

- **Etape 2 : Filtrage des compétences.** L'agent hôte filtre sa description fonctionnelle avec les descripteurs reçus dans le premier message. Il ne retient que ses compétences complémentaires à un besoin de l'agent arrivant.

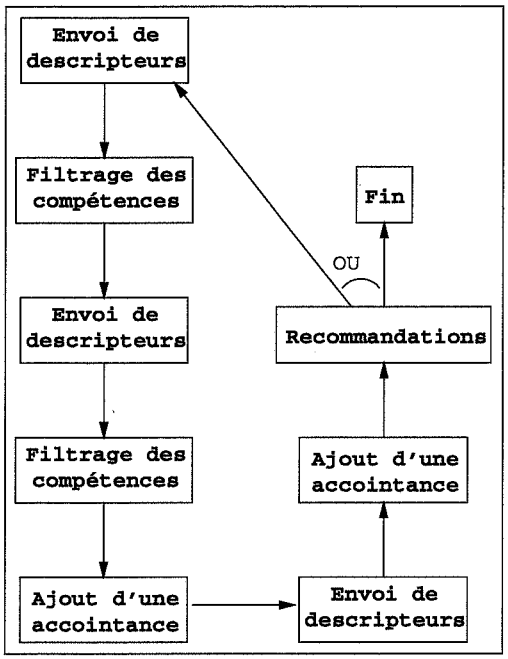


**Figure 5.2 :** Plan gérant les présentations d'un agent

- **Etape 3 : Envoi de descripteurs.** L'agent hôte envoie à l'agent arrivant ses compétences retenues lors du filtrage précédent ainsi que l'ensemble des descripteurs de sa description coopérative.
- **Etape 4 : Filtrage des compétences.** L'agent arrivant filtre lui aussi sa description fonctionnelle avec les besoins de l'agent hôte. Il ne retient que ses compétences complémentaires à un besoin de l'agent hôte.
- **Etape 5 : Ajout d'une accointance.** L'agent arrivant se crée une représentation de l'agent hôte. Un plan spécialise cette tâche (représenté figure 5.3), composé d'un ajout d'une représentation d'accointance et de la création de nouveaux stéréotypes.

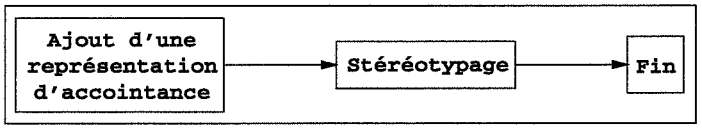


**Figure 5.3 :** Plan d'ajout d'une accointance

- **Etape 6 : Envoi de descripteurs.** L'agent arrivant envoie à l'agent hôte ses compétences retenues lors du filtrage précédent.

- **Etape 7 : Ajout d'une accointance.** L'agent hôte se créé une représentation de l'agent arrivant et construit les stéréotypes que cet ajout implique.
- **Etape 8 : Recommandations.** A l'issue d'une présentation, l'agent hôte recommande à l'agent arrivant de contacter d'autres agents d'après ce qu'il peut déduire de sa représentation des autres. L'agent hôte a ensuite le choix d'entamer de nouvelles présentations ou de finir cette phase pour passer aux notifications. Cette étape est détaillée dans la section 5.4.

Lors de la création d'une nouvelle représentation d'accointance, il n'est pas nécessaire que chaque agent y inclut la totalité de la description coopérative, ni la totalité des compétences de l'autre. Seuls les descripteurs pour lesquels une complémentarité a été trouvée sont retenus dans la représentation d'accointance. Les autres descripteurs ne sont pas d'une utilité directe pour la coopération entre ces deux agents. Pour permettre d'intégrer d'autres agents, il faut néanmoins que les agents retiennent les besoins ayant une relation d'égalité, de généralisation ou de spécialisation avec un de leurs besoins.

Cette acquisition de représentation d'accointance présente la particularité d'être subjective. En effet, en prenant comme filtre la description coopérative de chacun, les agents disposent d'une représentation de l'autre constituée uniquement de ses propriétés pertinentes pour l'agent se les représentant. Dès qu'une nouvelle représentation d'accointance est construite, celle-ci est prise en entrée du processus de stéréotypage pour rattacher le nouvel agent à des stéréotypes connus ou pour en créer de nouveaux.

## 5.4 Recommandation d'accointances

Le dernier devoir d'un agent accueillant après une présentation est de transmettre à son interlocuteur les adresses d'autres agents pour qu'il puisse poursuivre son intégration. Le plan associé à cette phase de recommandation est représenté par la figure 5.4.

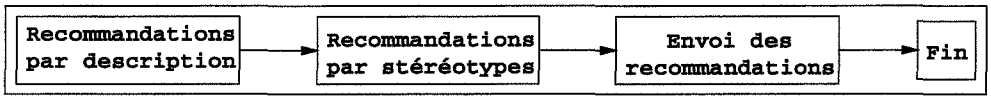


**Figure 5.4 :** Plan d'élaboration des recommandations

L'objectif des recommandations est de répéter des présentations, et donc des acquisitions de représentations d'accointance pour l'agent arrivant et pour les agents contactés, jusqu'à ce que l'agent arrivant se considère intégré. Dans un premier temps, nous expliquons suivant quels critères l'agent

arrivant se considère comme intégré ou non. Puis nous nous plaçons du côté des agents hôtes pour définir le mécanisme de recommandation. Enfin, la prise en compte et le suivi des recommandations par l'agent arrivant sont détaillés.

### 5.4.1 Critères d'évaluation de l'état d'intégration

Pour être intégré, un agent doit connaître tous les agents ayant des descripteurs complémentaires aux siens (et, pour un besoin, les agents ayant des besoins situés sur la même branche d'une arborescence). La notion d'intégration se réfère ainsi aux descripteurs d'un agent et à un état de connaissance relatif à ces descripteurs. On peut alors étendre l'intégration aux descripteurs en disant qu'un descripteur est *intégré* si ce descripteur n'empêche pas l'agent d'être totalement intégré. On peut maintenant ajouter que, si tous les descripteurs d'un agent sont intégrés, alors cet agent est intégré.

L'intégration rapportée aux niveaux des descripteurs d'un agent, nous permet de transposer la notion d'intégration telle que nous l'avons définie, au niveau local d'un agent. Celui-ci aura ainsi les moyens d'évaluer son état d'intégration, en connaissant l'état d'intégration de ses descripteurs. Avant d'entamer sa première présentation, l'agent arrivant étiquette temporairement (cette étiquette n'est plus nécessaire après l'intégration) chacun de ses descripteurs à la valeur booléenne *faux*. Cette valeur signifie que le descripteur n'est pas encore intégré. Nous montrons ci-dessous comment l'étiquette d'un descripteur peut être passée à *vrai* grâce aux recommandations.

Si on note $compl_\Omega(d)$ l'ensemble des agents ayant un descripteur complémentaire à $d$ (ou un besoin situé sur une même branche d'une arborescence), l'agent arrivant doit connaître $compl_\Omega(d)$ pour considérer $d$ comme intégré. Une recommandation contenant l'ensemble de ces agents peut lui être fournie dans les deux cas suivants :

#### Intégration d'une compétence

$d$ est une compétence et l'agent hôte a une compétence $c$ telle que :

$$class(c) \leqslant class(d)$$

L'ensemble $compl_\Omega(d)$ est défini tel que :

$$compl_\Omega(d) = \{\omega_k \in \Omega / \exists n \in CD_k^k / compl(d, n)\}$$

L'agent hôte étant intégré, il connaît l'ensemble $compl_\Omega(c)$. Or, la définition de la complémentarité entre une compétence $x$ et un besoin $y$ étant :

$$compl(x, y) \equiv class(x) \leqslant class(y),$$

on peut dire que pour tout besoin $n$ :

$$class(d) \leqslant class(n) \supset class(c) \leqslant class(n) \text{ ou}$$

$$compl(d, n) \supset compl(c, n)$$

On peut enfin déduire :

$$compl_\Omega(d) \subset compl_\Omega(c)$$

L'agent hôte peut alors recommander à l'agent arrivant tous les agents de l'ensemble $compl_\Omega(d)$ car il les connaît. L'agent arrivant considérera une de ses compétences comme intégrée dès qu'il aura reçu une recommandation d'un agent ayant une compétence plus spécifique que celle-ci. Une compétence peut également être intégrée par une recommandation d'un agent ayant un besoin sur une classe égale ou plus générale que la compétence. La démonstration est similaire à celle énoncée ci-dessus.

### Intégration d'un besoin

$d$ est un besoin et l'agent hôte a un besoin $n$ tel que :

$$class(n) \geqslant class(d)$$

L'ensemble $compl_\Omega(d)$ est défini tel que :

$$compl_\Omega(d) = \{\omega_k \in \Omega / (\exists c \in FD_k^k / compl(c, d)) \vee (\exists n' \in CD_k^k / related(d, n'))\}$$

D'après la définition de la complémentarité, on peut dire que pour toute compétence $c$ :

$$class(d) \geqslant class(c) \supset class(n) \geqslant class(c) \text{ ou}$$

$$compl(c, d) \supset compl(c, n)$$

Le prédicat *related* reliant deux besoins $x$ et $y$ exprimant que :

$$related(x, y) \equiv ((class(x) < class(y)) \vee (class(x) \geqslant class(y)))$$

et la structure des ensembles de classes étant une arborescence on peut dire que pour tout besoin $n'$ :

$$class(n') < class(d) \supset class(n') < class(n) \text{ et}$$

$$class(n') \geqslant class(d) \supset class(n') \geqslant class(n) \text{ c'est-à-dire que :}$$

$$related(d, n') \supset related(n, n')$$

On peut donc déduire :

$$compl_\Omega(d) \subset compl_\Omega(n)$$

Comme pour ses compétences, l'agent arrivant considérera un de ses besoins comme intégré dès qu'il recevra une recommandation d'un agent ayant un besoin plus général que celui-ci.

Le fait qu'un agent connaisse les agents ayant des besoins plus généraux que les siens se justifie ici. Comme une compétence satisfait les besoins désignant sa classe ainsi que ceux désignant une classe plus générale, il doit connaître tous les agents ayant de tels besoins. Pour considérer sa compétence intégrée, il doit trouver un agent ayant connaissance de tous ces agents ou contacter tous les agents du SMA (nous souhaitons éviter ce cas au maximum). Si notre contrainte sur la connaissance de besoins plus généraux n'existait pas, les seuls agents pouvant lui fournir cette connaissance seraient ceux ayant exactement la même compétence ou une plus spécifique. L'introduction de nouvelles compétences plus spécifiques que celles déjà présentes dans le SMA deviendrait alors très coûteuse car il faudrait à chaque fois que l'agent arrivant contacte tous les agents du SMA.

#### Remarques sur les plans

Les plans représentent un cas particulier de descripteurs au niveau de l'intégration. En effet, un plan peut satisfaire deux types de besoins : les besoins en plan et les besoins en tâches. Aussi, deux étiquettes doivent être attachées à chaque plan. La première est validée quand l'agent connaît tous les agents ayant un besoin en plan d'une classe plus générale ou égale à celle de son plan, et la seconde est validée qand l'agent connaît tous les agents ayant un besoin en tâche d'une classe plus générale ou égale à celle de son plan. Le plan ne sera considéré intégré que quand ses deux étiquettes seront vraies.

Maintenant que nous avons expliqué l'objectif des recommandations, nous détaillons le contenu des étapes du plan d'élaboration des recommandations.

### 5.4.2 Types de recommandation

Il existe deux types de recommandations effectuées par un agent hôte. Il peut recommander certaines de ses accointances par comparaison de :

- descriptions : l'agent hôte peut étiqueter certains descripteurs à *vrai* ou connaît les agents qui peuvent le faire.
- stéréotypes : il n'a pas d'informations pertinentes sur ce descripteur. Il va ordonner la liste de ses accointances selon le degré de similarité avec l'agent arrivant, en se basant sur les stéréotypes qu'il connaît, et lui conseiller de se présenter aux agents qui lui ressemblent le plus.

### Recommandation par descriptions

Cette recommandation est légèrement différente selon qu'elle s'applique à une compétence ou à un besoin. Pour chaque type de descripteur, nous distinguons plusieurs catégories de recommandations par description. Une recommandation est un message dont le contenu est noté $R(j, i, d)$ et représente un ensemble d'agents que l'agent hôte $\omega_j$ recommande à l'agent arrivant $\omega_i$ à propos d'un descripteur $d$. Pour émettre une recommandation, la connaissance de l'agent $\omega_j$ hôte doit satisfaire certaines conditions propres à la recommandation. Si plusieurs conditions sont satisfaites, l'agent enverra de préférence une recommandation directe. S'il ne peut pas en envoyer il enverra une recommandation indirecte et, s'il ne satisfait pas non plus aux conditions requises, enverra une recommandation partielle (ces types de recommandation sont définis ci-dessous).

**Recommandations directes pour une compétence** Ces recommandations fournissent une liste d'agent suffisantes pour que la compétence $d$ soit considérée comme intégrée par l'agent arrivant. Il existe deux types de recommandation directe pour une compétence, notés RDC1 et RDC2 :

**RDC1.**

Condition :
$\exists n \in CD_j^j / (class(n) \geqslant class(d)) \vee (class(n) < class(d))$
Agents recommandés :
$R(j, i, d) = \{\omega_k \in \Omega / \exists n \in CD_k^j / compl(d, n)\}$

**RDC2.**

Condition :
$\exists c \in FD_j^j / class(c) \leqslant class(d)$
Agents recommandés :
$R(j, i, d) = \{\omega_k \in \Omega / \exists n \in CD_k^j / compl(d, n)\}$

**Recommandations indirectes pour une compétence** Ces recommandations fournissent l'adresse d'un agent qui est capable de fournir une recommandation directe sur une compétence $d$. Les deux types de recommandation indirecte pour une compétence sont notés RIC1 et RIC2 :

**RIC1.**

Condition :
$\exists CD_k^j \in SK^j, \exists n \in CD_k^j / class(n) \geqslant class(d)$
Agents recommandés :
$R(j,i,d) = \{\omega_k \in \Omega / \} / \exists n \in CD_k^j / class(n) \geqslant class(d)$ et
$|R(j,i,d)| = 1$

**RIC2.**

Condition :
$\exists FD_k^j \in SK^j, \exists c \in FD_k^j / class(c) \leqslant class(d)$
Agents recommandés :
$R(j,i,d) = \{\omega_k \in \Omega / \} / \exists c \in FD_k^j / class(c) \leqslant class(d)$ et
$|R(j,i,d)| = 1$

**Recommandations directes pour un besoin** Ces recommandations fournissent une liste d'agent suffisantes pour que le besoin $d$ soit considéré comme intégré par l'agent arrivant. Il n'existe qu'un type de recommandation directe pour un besoin, noté RDB1 :

**RDB1.**

Condition :
$\exists n \in CD_j^j / class(n) \geqslant class(d)$
Agents recommandés :
$R(j,i,d) = \{\omega_k \in \Omega / (\exists c \in FD_k^j / compl(c,d)) \vee (\exists n \in CD_k^j / related(n,d))\}$

**Recommandations indirectes pour un besoin** Ces recommandations fournissent l'adresse d'un agent qui est capable de fournir une recommandation directe sur un besoin $d$. La recommandation indirecte pour un besoin est notée RIB1 :

**RIB1.**

Condition :
$\exists CD_k^j \in SK^j, \exists n \in CD_k^j / class(n) \geqslant class(d)$
Agents recommandés :
$R(j,i,d) = \{\omega_k \in \Omega /\} / \exists n \in CD_k^j / class(n) \geqslant class(d)$ et
$|R(j,i,d)| = 1$

**Recommandations partielles pour un besoin** Cette recommandation intervient dans le cas particulier où il n'existe pas d'agent pouvant effectuer des recommandations directes pour un besoin $d$ (aucun agent n'a de besoin plus général ou égal à $d$). L'agent arrivant devra contacter tous les agents du SMA pour considérer son besoin comme intégré. Le premier type de recommandation partielle, noté RPB1, fournit une partie des agents nécessaires à l'intégration du besoin. Le second type, noté RPB2, communique l'adresse d'un agent pouvant émettre une recommandation RPB1.

**RPB1.**

Condition :
$\exists n \in CD_j^j / class(n) < class(d)$ et
$\forall \omega_k \in \Omega, not \exists n' \in CD_k^j / class(n) < class(n')$
Agents recommandés :
$R(j,i,d) = \{\omega_k \in \Omega / (\exists c \in FD_k^j / compl(c,d)) \vee$
$(\exists n \in CD_k^j / related(n,d))\}$

**RPB2.**

Condition :
$\forall \omega_k \in \Omega, \exists n \in CD_k^j / class(n) < class(d)$
Agents recommandés :
$R(j,i,d) = \{\omega_k \in \Omega /\} / \exists n \in CD_k^j / class(n) < class(d)$ et
$\forall \omega_l \in \Omega, not \exists n' \in CD_l^j / class(n) < class(n')$ et
$|R(j,i,d)| = 1$

Un besoin peut uniquement être étiqueté à *vrai* par un agent ayant le même besoin ou un besoin plus général (recommandation directe RDB1). Les recommandations indirectes recommandent un agent pouvant faire une recommandation directe, et celles incertaines, des agents pouvant faire des

recommandations incertaines ou indirectes.

Pour simplifier la notation, nous avons écrit qu'une recommandation était un ensemble d'agents. C'est en réalité un ensemble d'adresses d'agent auxquelles on attache le descripteur justifiant la recommandation de cet agent. Enfin, seuls les descripteurs étiquetés à faux avant la recommandation sont considérés.

#### Recommandation par stéréotypes

L'intégration d'un agent ne peut pas être assurée par la seule recommandation par descriptions. En effet, il est très probable que le premier agent auquel l'agent arrivant se présente, ne connaisse pas des agents complémentaires à chacun de ses descripteurs. De plus chaque agent n'ayant qu'une représentation subjective de ses accointances, il peut ne pas reconnaître des complémentarités entre deux autres agents.

En complément de ses éventuelles recommandations par descriptions, l'agent hôte va envoyer la liste des adresses de ses accointances ordonnées suivant les similarités entre leur description d'agent et celle de l'agent arrivant. En effet, les agents les mieux placés pour accueillir le nouvel arrivant sont ceux ayant un grand nombre de compétences ou de besoins en commun avec lui. Les stéréotypes construits par l'agent sont alors utilisés pour évaluer la similarité entre agents.

La similarité entre deux agents correspond à la plus haute pertinence (définie dans la section 4.3.3) parmi les stéréotypes qu'ils partagent. Ainsi, dès qu'un agent hôte a mis à jour sa hiérarchie de stéréotypes pour prendre en compte l'agent arrivant, il parcourt tous les stéréotypes dans lesquels il entre et, pour chaque stéréotype, si les agents y appartenant n'ont pas encore de valeur de similarité ou s'ils en ont une plus basse que la pertinence du stéréotype, il leur assigne la pertinence du stéréotype.

### 5.4.3 Exploitation des recommandations

Dans le traitement des recommandations, l'agent arrivant accorde plus d'importance aux recommandations par descriptions. Pour le choix de son interlocuteur pour la prochaine présentation il considère par ordre de priorité décroissante :

1. Le nombre de recommandations de type indirectes dans lesquelles apparaissent l'interlocuteur;

2. Le nombre de recommandations de type incertaines dans lesquelles apparaissent l'interlocuteur;

3. La plus haute similarité entre lui et son interlocuteur.

Si ces critères ne suffisent pas à ressortir un agent, le choix est effectué aléatoirement.

Chaque nouvelle présentation se terminant par une recommandation, l'agent arrivant doit fusionner les nouvelles informations à ses précédentes listes d'agent. Les recommandations par descriptions consistent en une union des ensembles et les nouvelles mesures de similarité sont ajoutées à celles précédemment reçues.

Comme nous l'avons précisé précédemment, les descripteurs étiquetés à la valeur *vrai* ne sont pas assujettis à de nouvelles recommandations par description, et, un changement de valeur entraîne l'abandon de la liste attachée (sauf celles issues de recommandations directes car ces agents doivent être informés du descripteur de l'agent arrivant) et peut remettre en cause l'ordre de présentation obtenu au cycle précédent.

Un agent arrivant n'entame plus de nouvelles présentations quand tous ses descripteurs ont été étiquetés à *vrai*. En effet, il connaît tous les agents ayant des descripteurs complémentaires aux siens et il n'a plus à acquérir des représentations d'autres agents (elles lui seraient inutiles). Pour maintenir la cohérence globale du système et finir son intégration, il doit néanmoins les contacter pour les informer de leur complémentarité afin qu'ils puissent initier une coopération. C'est la seconde partie du plan d'intégration, la phase de notification.

## 5.5 Notifications

L'agent arrivant notifie sa présence et certaines de ses caractéristiques en suivant le plan présenté en figure 5.5

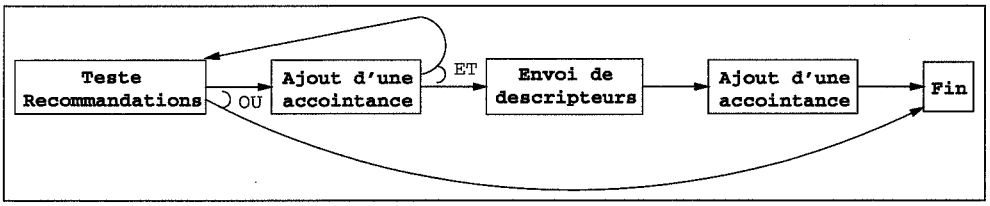


**Figure 5.5 :** Plan pour les notifications

Lorsque l'agent arrivant en est à cette phase, tous ses descripteurs sont intégrés. Sa représentation des autres est donc complète mais elle n'est pas réciproque. C'est-à-dire qu'il connaît sa complémentarité avec certains agents sans que ces agents la connaissent (il la connaît grâce à une recommandation par description). La phase de notification lui demande alors de les informer de ces complémentarités.

L'agent hôte teste dans un premier temps s'il a reçu des recommandations directes sur des agents avec lesquels il n'a pas effectué de présentation.

Si c'est le cas il se créé une représentation d'accointance de cet agent et lui notifie leurs complémentarités. Celui-ci ne répond pas au message mais se créé également une représentation d'accointance caractérisant l'agent arrivant.

## 5.6 Mises à jour de la représentation des autres

L'ajout d'un agent à un SMA est une cause de mise à jour des connaissances sociales de certains agents. Les agents présentant des complémentarités avec l'agent arrivant ajoutent une représentation de cet agent à leurs connaissances sociales. Il existe d'autres modifications possibles des connaissances d'un agent, chacune étant associée à un événement spécifique : le retrait d'un agent, l'évolution d'un agent ou la modification des arborescences de classes de descripteurs.

### 5.6.1 Retrait d'un agent

Les migrations d'agent dans un SMA ouvert peuvent s'effectuer dans les deux sens. Ainsi, on peut ajouter des agents à la société existante mais aussi en retirer. Il faut distinguer la suppression volontaire d'un agent, que nous appelons son retrait, et celle involontaire due à un dysfonctionnement de l'agent (traitée dans la section 6.2.1).

Notre modèle de représentation des autres facilite grandement le retrait d'un agent. Cette représentation est symétrique, c'est-à-dire que si un agent $\omega_i$ a une représentation d'un agent $\omega_j$, l'agent $\omega_j$ a aussi une représentation de $\omega_i$.

Grâce à cette symétrie de représentations, un agent, avant de quitter le SMA, peut prévenir les agents ayant des descripteurs complémentaires aux siens, car, s'il est intégré au système, il les connaît tous. Il envoie alors un message à ses accointances pour leur signaler son départ (la figure 5.6 présente le plan de retrait d'un agent). Ceux-ci répercutent ce départ par deux opérations sur leurs connaissances :

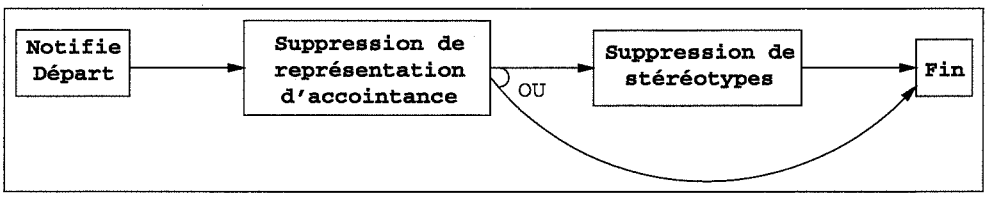


**Figure 5.6 :** Plan pour un retrait d'agent

- La suppression de la représentation d'accointance correspondant à cet agent ;

- la suppression des stéréotypes auxquels il appartient et regroupant un seul autre agent. Les stéréotypes ou représentations d'accointance plus spécifiques que celui supprimé sont reliés à ses stéréotypes plus généraux (pour conserver la cohérence de la hiérarchie).

La suppression des représentations d'un agent quittant le SMA n'est pas obligatoire et se justifie principalement dans le cas d'un retrait définitif. En effet, on peut imaginer qu'un agent soit temporairement indisponible puis réintègre le SMA. Dans ce cas, l'agent doit signaler son absence momentanée pour qu'il n'y ait pas ambiguïté avec un dysfonctionnement. Ce message n'entraîne pas de suppression de représentation ou de stéréotype.

### 5.6.2 Evolution d'un agent

Un agent évolutif est un agent dont les compétences et buts peuvent changer. Ses besoins étant déduits à partir de ses compétences et buts, l'évolution d'un agent entraîne alors la modification de sa description coopérative. Toutes ces modifications ont deux conséquences :

- Les représentations de cet agent maintenues par ses accointances deviennent obsolètes;
- L'agent n'est plus intégré au SMA car sa représentation des autres n'est plus pertinente et elle est incomplète car il ne connaît pas les agents complémentaires à ses nouveaux descripteurs.

Avant d'adopter ses nouveaux descripteurs, l'agent garde deux descriptions d'agent le représentant: son ancienne description et celle qu'il aura après son évolution. De nouvelles étiquettes booléennes sont assignées à chaque descripteur de la nouvelle description d'agent. Les descripteurs inchangés prennent la valeur *vrai* et les nouveaux descripteurs prennent la valeur *faux*. La figure 5.7 montre les étapes du plan d'évolution d'un agent.

A l'instar du processus d'ajout d'un agent, l'agent ayant évolué se considérera intégré à nouveau dès que tous ses descripteurs auront une étiquette à *vrai*. Le traitement d'une compétence $c$ non satisfaite (étiquetée à *faux*) dépend de sa situation par rapport à l'ancienne description de l'agent que nous notons $AD^{old}$ (les situations sont présentées par ordre de priorité décroissante) :

- $\exists d \in AD^{old} / class(c) > class(d)$. $c$ est une généralisation d'une compétence de l'ancienne description. L'ensemble des agents ayant besoin de $c$ est un sous-ensemble des agents ayant besoin de $d$. Il suffit alors de prévenir ce sous-ensemble de la présence de $c$ (la tâche *Notifie Changement* du plan). $c$ est étiquetée à *vrai*.

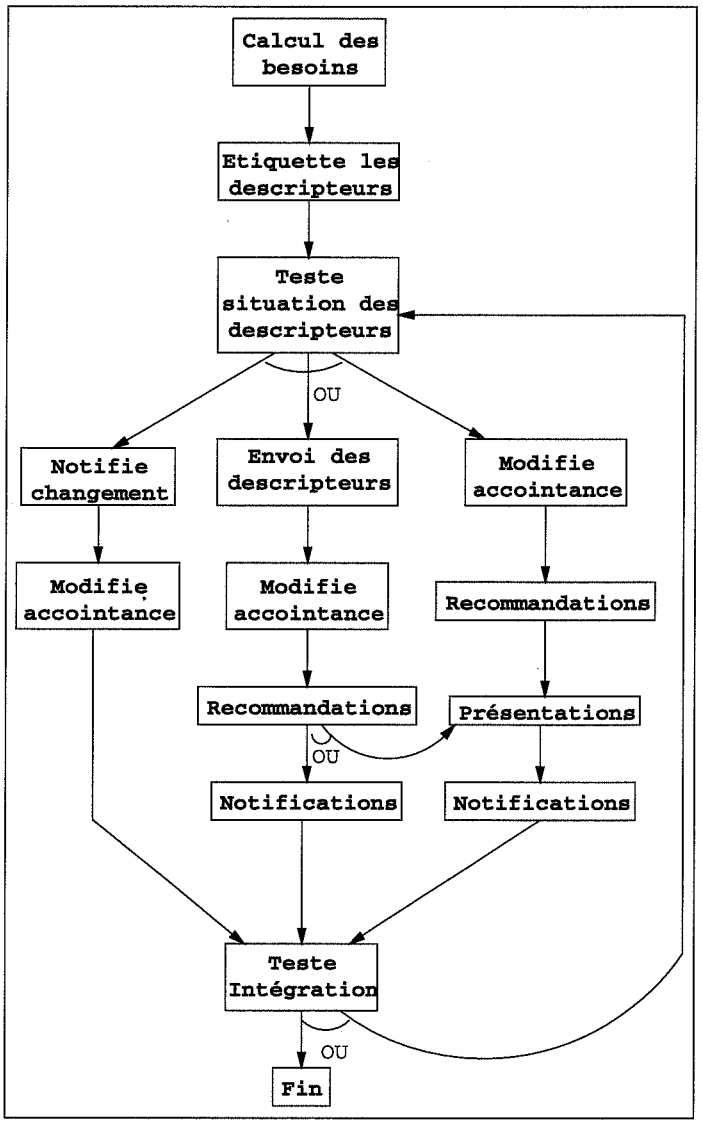


**Figure 5.7 :** Plan de gestion de l'évolution d'un agent

- $\exists d \in AD^{old}/class(c) < class(d)$. $c$ est une spécialisation d'une compétence de l'ancienne description. Cette fois l'agent ne connaît pas les agents ayant besoin de $c$ mais ceux qui ont besoin de $d$ lui fourniront des recommandations directes de type RDC1. Il demande alors à l'un d'entre eux une recommandation sur $c$, ce qui étiquette $c$ à *vrai*. S'il ne connaît aucun agent ayant besoin de $d$, la compétence $c$ reste à faux et est conservée dans une liste "à présenter".

- Dans tout autre cas, $c$ est gardée dans la liste "à présenter".

Les situations d'un besoin $n$ par rapport à $AD^{old}$ sont (par ordre de priorité décroissante) :

- $\exists m \in AD^{old}/class(n) < class(m)$. $n$ est une spécialisation d'un besoin de l'ancienne description. L'ensemble des agents ayant des compétences complémentaires à $n$, ou ayant un besoin plus spécifique ou plus général, est un sous-ensemble des agents ayant des compétences complémentaires à $d$. Il suffit alors de prévenir ce sous-ensemble de la présence de $n$. $n$ est étiquetée à *vrai*.

- $\exists m \in AD^{old}/class(n) > class(m)$. $n$ est une généralisation d'un besoin de l'ancienne description. L'agent ne connaît pas les agent ayant des compétences complémentaires à $n$, ou ceux qui ont un besoin plus spécifique, mais ceux qui ont un besoin plus général ou égal à $n$ lui fourniront des recommandations directes. Il demande alors une recommandation sur $n$ aux agent ayant un besoin $n' \geqslant n$, ce qui étiquete $n$ à *vrai*. S'il ne connaît aucun agent ayant ce profil, le besoin $n$ reste à faux et est conservée dans une liste "à présenter".

- Dans tout autre cas, $n$ est gardé dans la liste "à présenter".

Les descripteurs restés à la valeur *faux* sont regroupés dans une liste "à présenter". L'agent va d'abord mettre à jour ses stéréotypes (en enlevant son ancienne description et en ajoutant sa nouvelle) et se recommander (par similarité) à lui-même les accointances à qui présenter ses nouveaux descripteurs. L'agent reprendra alors une série de présentations pour valider ses nouveaux descripteurs.

Quand tous ses descripteurs sont passés à *vrai*, l'agent n'a plus qu'à prévenir certaines de ses accointances de la suppression de certains de ses descripteurs ne se trouvant plus dans sa nouvelle description d'agent. Ces accointances modifient alors leurs représentations de l'agent ayant évolué et modifient leurs stéréotypes en enlevant l'ancienne description et en ajoutant la nouvelle.

### 5.6.3 Modification des arborescences de classes

L'ajout de nouveaux services au SMA se fait soit par l'ajout d'un agent les encapsulant soit par la modification d'un agent et l'évolution de sa description d'agent pour les prendre en compte. Une propriété intéressante des SMA ouverts est de pouvoir y ajouter des services n'étant pas spécifiés ou définis à la création du SMA. Rapporté à notre modèle, ce cas correspond à l'ajout de nouvelles classes de buts ou compétences inconnues des autres agents du système.

Chaque agent ne connaît qu'une partie des arborescences de classes, mais toutes ces parties doivent former des arborescences globales cohérentes. Si deux agents n'ont pas la même définition d'une classe ou ne lui attribuent pas la même position dans l'arborescence, ils ne déduiront pas les mêmes complémentarités avec les descripteurs des autres classes alors qu'ils conviendront qu'il s'agit des mêmes classes.

La modification des arborescences est un sujet sensible que nous limitons par deux contraintes :

- Il ne peut pas y avoir de suppression de classe ou de modification du contenu (identifiant, buts satisfaits, ressources utilisées, ...) d'une classe;
- Si deux classes sont reliées par un chemin, il faut conserver un chemin (qui peut néanmoins parcourir d'autres classes) reliant ces deux classes et de même orientation (la classe qui était la plus générale le reste).

En interdisant la suppression de classe ou leurs modifications internes, il n'existe plus que trois possibilités de modifications d'arborescences : l'ajout et l'insertion de nouvelles classes et le déplacement de classes dans l'arborescence. La figure 5.8 illustre les différentes modifications permises. Dans une arborescence, l'ajout de classes correspond à l'ajout de feuilles, l'insertion de classes correspond à l'ajout d'un nœud qui n'est pas une feuille, et le déplacement de classes se traduit par l'ajout de nœuds plus généraux et plus spécifiques qui étaient déjà présent dans l'arborescence.

L'**ajout de classes** ne nécessite pas de traitement particulier. Lors d'une présentation, un agent explique à son interlocuteur comment rattacher une classe à son arborescence (en précisant le chemin depuis la racine) si elle lui est inconnue. Les arborescences des agents ayant une complémentarité avec les nouvelles classes seront mises à jour prendant ces présentations.

Pour l'**insertion de classes**, tout est également réglé pendant les présentations. Pour les classes appartenant à $EC_1$ le cas est semblable à l'ajout de classes. Les agents ayant des descripteurs appartenant à une classe de $EC_2$ seront également prévenus pendant une présentation car les complémentarités sont détectées.

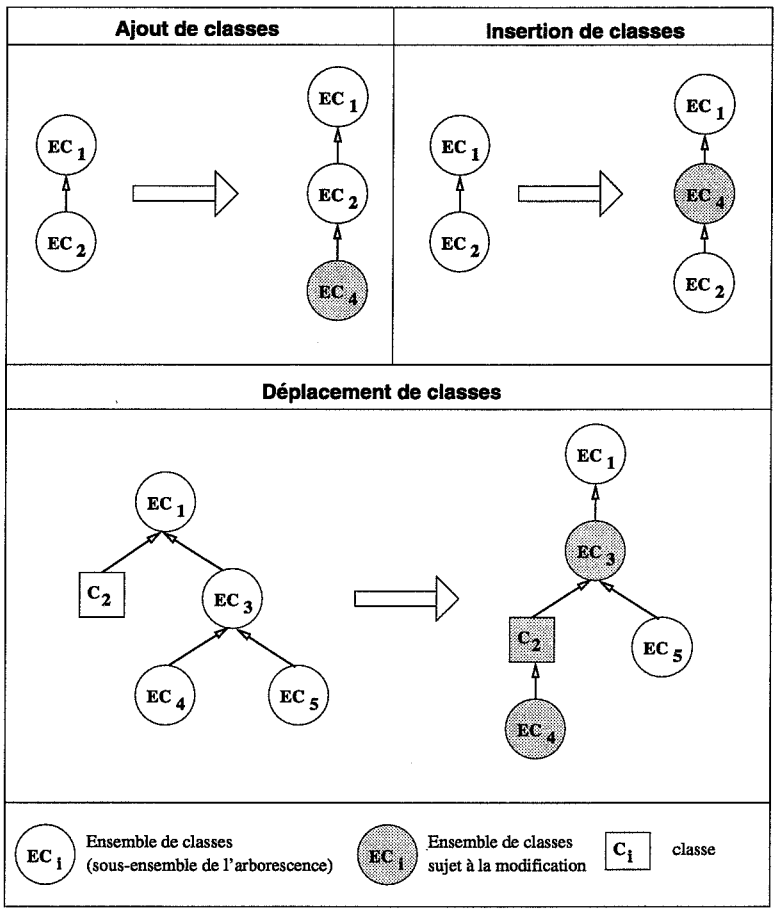


**Figure 5.8 :** Modifications d'arborescence

Pour traiter l'ajout et l'insertion de classes, un agent accueillant peut utiliser un plan à la place d'une action chaque fois que la tâche *Envoi de descripteurs* apparaît dans un plan. La figure 5.9 représente ce plan.

Le **déplacement de classes** dans une arborescence est plus problématique. Les agents qui doivent être mis au courant de ce changement sont ceux ayant besoin d'une classe de $EC_3$, ayant des besoins ou compétences de la classe $C_2$ ou ayant des besoins ou compétences d'une classe de $EC_4$. L'agent ayant provoqué ce changement est chargé de trouver tous ces agents pour leur signaler le changement. Il se comportera comme s'il avait un besoin de la classe la plus spécifique appartenant à $EC_1$ suite à une évolution. Bien entendu, pendant les messages échangés avec d'autres agents, il précise que ce besoin est fictif pour qu'il ne soit pas représenté dans les descriptions d'agents le concernant.

L'agent peut alors signaler la modification à tous les agents concernés.

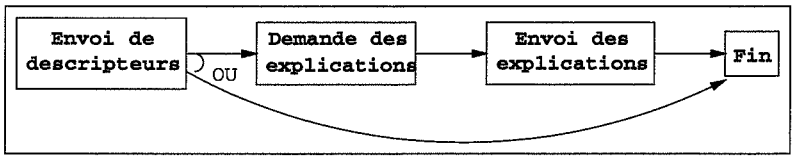


**Figure 5.9 :** Plan d'envoi de descripteurs avec leurs explications

Cependant, ceux-ci ne sont plus intégrés au SMA car les agents ayant une compétence de la classe $C_2$ peuvent désormais satisfaire des besoins d'une classe de $EC_3$ sans que les agents le sachent, de même que les agents ayant un besoin de la classe $C_2$ ne connaissent pas les agents ayant une compétence d'une classe de $EC_4$ ni les agents ayant des besoins d'une classe de $EC_3$ ou $EC_4$. L'agent initiateur du déplacement peut les aider car il connaît tous ces agents et leurs complémentarités et leur fournira les informations manquantes pour leur éviter d'entamer de nouvelles présentations. Après ces derniers messages, l'agent peut "oublier" les informations acquises pour la mise à jour des connaissances et qui ne lui sont pas pertinentes pour ses propres coopérations.

## 5.7 Conclusion sur l'intégration d'agents

Dans ce chapitre, nous avons présenté l'intégration des agents comme la notion centrale et essentielle à l'ouverture d'un SMA. Notre définition de l'intégration est fondée sur la pertinence de la connaissance d'un agent sur les autres agents dans le sens où celle-ci lui permet de déduire ses possibilités de coopération avec ses accointances. Tous les événements caractéristiques d'un SMA ouvert (ajout, retrait ou évolution d'agent) consistent en un problème de création et de maintien de l'intégration des agents.

Des recommandations sont utilisées pour diriger rapidemment un agent vers ceux qui lui sont complémentaires. Cependant, il reste possible qu'un agent pour atteindre son intégration doive contacter tous les agents du SMA. Cela reste néanmoins un cas particulier provoqué par une très forte indépendance des agents du SMA (dans le sens où les agents ont très peu de possibilités de coopération).

Dans la dernière partie de cette thèse, nous prenons du recul par rapport à la connaissance et au raisonnement des agents pour considérer le SMA ouvert de manière globale. L'intégration d'agents est définie au niveau du système ce qui nous permet de comparer les différentes approches de l'ouverture d'un SMA. Nous apportons également quelques précisions sur la robustesse ou la sécurité dans nos SMA ouverts, avant d'expliquer comment un SMA ouvert peut être utilisé comme une couche d'encapsulation d'un système d'information distribué pour l'"ouvrir" en autorisant l'ajout,

le retrait et l'évolution des entités le composant.

# Troisième partie

# Conception d'un Système Multi-Agent Ouvert

# Chapitre 6

# D'un agent à une société accueillante

Nous avons exposé dans les chapitres 4 et 5 un mécanisme de distribution de la tâche d'ouverture d'un SMA, parmi les agents le composant, sous la forme de facettes accueillantes assurant une partie de l'intégration d'agents. Les champs de connaissance et le processus de raisonnement d'une facette accueillante y ont été présentés. Dans ce chapitre, nous étudions les conséquences de notre définition de la facette accueillante au niveau du SMA. Nous y adoptons un point de vue global en considérant que l'objectif, derrière la définition d'un agent accueillant, est de pouvoir concevoir une *société accueillante*.

Nous reprenons la notion d'intégration des agents pour l'étudier de manière globale et établir les hypothèses avec lesquelles on peut garantir et montrer qu'un agent peut intégrer totalement ou partiellement le SMA. Nous détaillons par la suite comment la robustesse du SMA envers les pannes d'agent peut être assurée et comment l'ouverture peut être limitée pour gagner en sécurité. Cette analyse générale nous permet alors de comparer notre approche aux autres approches existantes. Enfin, nous généralisons notre modèle pour montrer de quelle manière on peut s'en servir pour définir l'ouverture de manière centralisée ou pour gérer la fusion de SMA (intégrer un SMA, plutôt qu'un agent, à un autre SMA).

## 6.1 Caractéristiques de notre approche

Le raisonnement accueillant de chaque agent a pour objectif de maintenir ou de créer l'intégration d'agents (que ce soit sa propre intégration ou celle de ses accointances). Nous avons distingué l'intégration totale, telle que chaque agent intégré connaît tous les agents ayant des descripteurs complémentaires aux siens (et connaît tous ces descripteurs), et l'intégration partielle où un agent doit connaître au moins un agent ayant un descripteur complémentaire

pour chacun de ses propres descripteurs.

### 6.1.1 Intégration totale

L'intégration d'un agent (et son maintien) s'effectue par le biais de présentations successives pendant lesquelles les deux agents impliqués s'échangent des informations sur leur propre description pour transmettre à l'autre un ensemble de connaissances nécessaires à son intégration (cf. chapitre 5). Le bon déroulement des mécanismes de création et de maintien de l'intégration dépend de plusieurs hypothèses. Les deux premières hypothèses sont :

**Hypothèse 1.** *Les agents sont coopératifs (ils acceptent toute entame de protocole de présentation) donc sincères (un agent envoie des connaissances qu'il croit lui-même et n'occulte pas d'informations qui lui semblent pertinentes) pour les opérations en rapport avec l'intégration d'agents.*

**Hypothèse 2.** *Tout agent appartenant au SMA est intégré et accueillant.*

Elles nous assurent du bon fonctionnement d'une présentation. Si les agents du SMA sont intégrés et accueillants, ils disposent des connaissances nécessaires pour remplir leur rôle dans le protocole de présentation. En les qualifiant de coopératifs et sincères, on suppose que les agents acceptent d'entamer toute présentation et qu'ils y communiquent les informations appropriées. L'état final d'intégration d'un agent dépend cependant de la succession de plusieurs présentations. Chaque agent du SMA doit pouvoir être contactable pour une présentation et ce, quelque soit l'agent utilisé comme point d'entrée. En considérant la représentation des autres (au sens général) comme un graphe (que nous appelons *graphe d'accointance*) dans lequel un nœud représente un agent et un arc signifie que les deux agents reliés ont chacun une représentation de l'autre, il faut que, quelque soit deux nœuds du graphe d'accointance, il existe un chemin entre ceux-ci. En d'autres termes :

**Hypothèse 3.** *Le graphe d'accointance est connexe.*

Enfin il est possible que plusieurs agents soient en même temps en cours d'intégration. Si deux agents A et B sont en cours d'intégration et présentent des descripteurs complémentaires, ils seront amenés à se présenter à certains agents en commun et le second agent à se présenter (disons B) se verra recommander l'agent A. Par contre cela n'est possible que si ces agents ayant pris part à une présentation avec chaque agent, a eu le temps de se construire une représentation du premier agent qui s'est présenté. Ce qui revient à dire que :

**Hypothèse 4.** *Les protocoles de présentation sont des sections critiques (c'est-à-dire qu'un agent ne peut participer en même temps à deux protocoles de présentation) et un agent non-intégré au SMA ne peut pas fournir de recommandation directe.*

Cette hypothèse garantit également le bon déroulement de modifications simultanées des arborescences de classes. Celles-ci n'ont qu'à respecter les règles énoncées comme si elles avaient lieu séquentiellement, c'est-à-dire respecter certaines contraintes sur l'arborescence modifiée, et elles doivent également être compatibles (deux modifications ne doivent pas se contredire).

Dans un SMA respectant ces hypothèses, il ne peut pas y avoir d'agent qui ne soit pas intégré au système (hormis ceux en cours d'intégration). Examinons les deux situations dans lesquels un agent ne serait pas intégré:

1. **Un agent $\omega_i$ ne connaît pas un autre agent $\omega_j$ ayant des descripteurs complémentaires aux siens.** S'il ne le connaît pas c'est qu'il n'a pas effectué de présentation avec celui-ci ou que $\omega_j$ n'a pas fait l'objet d'une recommandation directe par une description (cf. section 5.4.2). S'il existe un autre agent $\omega_k$, connu de $\omega_i$, et ayant des complémentarités avec $\omega_i$ qui sont communes avec celles entre $\omega_i$ et $\omega_j$, l'agent $\omega_k$ doit connaître $\omega_j$ (car il est intégré, hypothèse 2) et il doit avoir recommandé $\omega_j$ à $\omega_i$ (hypothèse 1). S'il n'existe pas d'agent correspondant à $\omega_k$, l'agent $\omega_i$ n'a pas étiquetté ses descripteurs concernés tant qu'il ne s'est pas présenté à $\omega_j$. Il enchaîne alors de nouvelles présentations jusqu'à contacter $\omega_j$. L'adresse de l'agent $\omega_j$ lui sera transmise à un moment car il existe au moins un agent le connaissant (hypothèse 3).

2. **Un agent $\omega_i$ connaît un autre agent $\omega_j$ mais pas tous ses descripteurs complémentaires aux siens.** Ce cas n'est pas possible car si l'agent $\omega_i$ a contacté l'agent $\omega_j$, c'est soit pour une présentation, soit à la suite d'une recommandation par description. Si c'est lors d'une présentation, toutes les complémentarités entre les deux agents ont été communiquées (hypothèse 1), si c'est pour une notification, celle-ci ne concerne pas les descripteurs manquants, or l'agent $\omega_i$ continuera des présentations pour ses propres descripteurs non validés et se présentera à $\omega_j$.

L'ensemble de ces hypothèses peut représenter une contrainte trop restrictive pour certains systèmes multi-agents ouverts. Il est alors possible de "réduire" l'intégration des agents pour supprimer une de ces hypothèses ou pour limiter le coût de la phase d'intégration.

### 6.1.2 Intégration dans un graphe d'accointance non-connexe

On peut ainsi lever l'hypothèse selon laquelle le graphe d'accointance est connexe si l'on peut avoir une vue globale de ce graphe. La connexité est requise pour permettre à un agent de contacter n'importe quel autre agent si on le lui recommande. Pour préserver la connexité, un agent peut avoir à

se représenter d'autres agents uniquement par leur adresse, c'est-à-dire sans qu'il existe de complémentarités entre eux. Cette situation peut se produire en cas de retrait ou de défaillance d'un agent du SMA. Nous proposons une méthode pour restaurer la connexité dans la section 6.2.1.

Il est cependant possible que les agents s'organisent sous la forme de groupes de complémentarité dans lesquels les agents coopèrent exclusivement avec les agents appartenant au même groupe. Un SMA de ce type proposerait plusieurs services distincts où chaque service est accompli par un ensemble de tâches, lesquelles ne sont pas utilisées dans la réalisation d'un autre service. De plus, les descripteurs d'un agent ne concerneraient qu'un seul service, c'est-à-dire qu'un agent ne peut participer qu'à la réalisation d'un seul service donné.

Dans ce cas particulier, le graphe d'accointance du SMA dispose de plusieurs composantes connexes, de telle sorte que chaque composante représente un ensemble de services et regroupe les agents le réalisant. On peut alors considérer chacune de ces composantes comme un SMA à part entière dans lequel les agents peuvent être intégrés totalement. Il ne peut pas y avoir d'intégration totale au niveau du SMA global car son graphe d'accointance n'est pas connexe mais elle reste possible dans chaque composante connexe.

La perte de la connexité globale nécessite cependant une vue globale du graphe d'accointance du SMA. En effet, pour intégrer un nouvel agent au SMA, il faut que le premier agent qu'il contacte (son point d'entrée) appartienne à la composante connexe regroupant les agents complémentaires au nouvel arrivant. Le concepteur doit alors connaître les différentes composantes connexes ainsi que leurs spécialités pour savoir dans laquelle il doit ajouter un nouvel agent. Cette contrainte s'applique également à l'évolution des agents. L'évolution ne peut se faire que vers des classes de descripteurs n'appartenant pas à d'autres composantes connexes car, si cela se produisait, l'agent ne pourrait pas contacter et informer les agents d'autres composantes connexes de leurs nouvelles complémentarités.

### 6.1.3 Intégration partielle

L'intégration totale peut se révéler coûteuse si on l'applique à des SMA composés de nombreux agents et sujets à de nombreuses migrations ou évolutions d'agent. Dans la section 5.1.2, nous avons défini l'intégration partielle d'un agent comme une intégration plus souple et moins contraignante que l'intégration totale. Pour rappel, un agent est partiellement intégré s'il connaît, pour chacun de ses descripteurs, au moins un agent présentant un descripteur complémentaire, et, pour chacun de ses besoins, au moins un agent ayant un besoin plus général et au moins un agent ayant un besoin plus spécifique. Le choix d'une intégration partielle plutôt qu'une intégration totale a des implications importantes sur le SMA et son ouverture.

Les avantages d'une telle approche se situent au niveau du coût en com-

munication lors de l'intégration des agents. Un agent ne cherchant pas, pour chacun de ses descripteurs, tous les agents ayant un descripteur complémentaire, il n'enverra pas de messages autres que pour entamer une présentation. Les recommandations directes pour une compétence n'ont plus de sens pour une intégration partielle car il suffit de connaître l'agent pouvant émettre cette recommandation pour que le descripteur soit considéré comme intégré. Les communications sont donc réduites pour une intégration partielle, à l'exception des recommandations indirectes pour un besoin qui ne débouchent pas forcément sur des recommandations directes pour un besoin comme pendant une intégration totale. En effet, les agents ne connaissent pas tous les agents ayant des besoins plus généraux que les leurs et ils ne peuvent pas recommander celui ayant le besoin le plus général sur la même branche de l'arborescence concernée. Pour atteindre cet agent (qui peut valider le descripteur traité), il est probable qu'une recommandation indirecte pour un besoin recommande un agent fournissant une autre recommandation indirecte pour un besoin et ainsi de suite jusqu'à recommander l'agent souhaité.

Il est cependant assez logique que les recommandations soient moins efficaces car elles s'appuient sur moins de connaissances que celles disponibles pour un agent totalement intégré. Le gain en coût de communications n'est évidemment pas gratuit et s'accompagne de certaines restrictions portant sur les agents et le SMA. Ainsi, un agent partiellement intégré ne peut pas modifier librement les arborescences de classes de descripteurs de la même manière qu'un agent totalement intégré. En effet, il lui est toujours possible d'ajouter ou d'insérer de nouvelles classes mais il ne peut plus les déplacer dans l'arborescence. Une telle opération a des conséquences sur les autres agents ayant des descripteurs en rapport avec cette classe et ils doivent être informés du changement. Or, ceux-ci sont connus d'un agent totalement intégré mais pas d'un agent partiellement intégré.

Une autre conséquence de l'intégration partielle apparaît pendant la recherche de partenaires pour une coopération. En connaissant moins d'agents complémentaires, le choix se fait sur la base de peu d'agents (voire d'un seul). Cette caractéristique nous donne des indications sur les contextes d'application adaptés à l'utilisation d'une intégration partielle. En ne donnant pas à un agent les moyens de choisir un partenaire parmi tous les agents présentant des mêmes classes de compétences, ce choix est en grande partie aléatoire et, si l'on ne souhaite pas que cela influe sur les performances de la coopération, il est souhaitable que les agents bénéficiant des mêmes classes de compétences ne diffèrent pas notablement dans leur efficacité ou dans leur spécialité.

Enfin, la possibilité d'intégrer totalement ou partiellement des agents dépend de l'état d'intégration des autres agents du SMA. L'intégration partielle est permise dans une société composée d'agents intégrés partiellement ou totalement alors que l'intégration totale nécessite que tous les agents

soient intégrés totalement.

Une de nos hypothèses pour assurer l'intégration totale des agents est que le graphe d'accointance est connexe. Or, le retrait d'un agent, ou même la panne d'un agent, peut mettre en péril cette connexité. Nous proposons dans la section suivante une spécialisation de la facette accueillante permettant de restaurer la connexité et donc d'apporter plus de robustesse ainsi que des pistes de spécialisation de cette facette pour introduire des notions de sécurité dans notre SMA.

## 6.2 Spécialisations de la facette accueillante

La facette accueillante permet de concevoir des SMA distribués et ouverts si l'on respecte les hypothèses énoncées au début de ce chapitre. La confrontation de notre modèle théorique à un système réel peut néanmoins engendrer des conflits avec ces hypothèses. Deux problèmes importants concernent le maintien de la connexité suite à une défaillance ou au retrait d'un agent, et la sécurité dans l'accès à certaines ressources ou compétences sensibles. Nous proposons des spécialisations de la facette accueillante pour répondre à l'un ou l'autre de ces problèmes.

### 6.2.1 Robustesse du système

La connexité du graphe d'accointance est indispensable pour permettre à un agent de se faire recommander n'importe quel autre agent du SMA quelque soit l'agent lui servant de point d'entrée. Si un agent quitte le SMA (par un retrait volontaire ou un dysfonctionnement), le graphe d'accointance peut se scinder en plusieurs composantes connexes ce qui aurait pour conséquence l'impossibilité d'intégrer totalement de nouveaux agents. La figure 6.1 représente un graphe d'accointance vulnérable à la panne d'un agent.

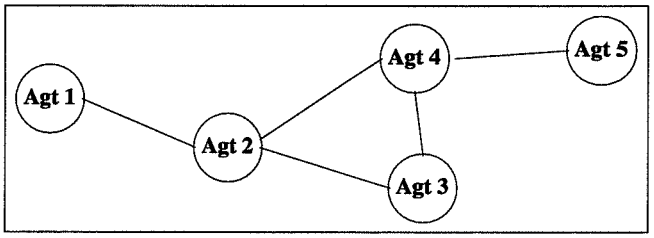


**Figure 6.1 :** Graphe d'accointance vulnérable à une panne

Les agents 2 et 4 sont indispensables à la connexité de l'ensemble. Si, par exemple, l'agent 2 disparaît, l'agent 1 se trouve isolé des agents 3, 4 et 5. Ainsi tout nouvel agent se présentant à l'agent 3, 4 ou 5 ne pourra pas

être mis en contact avec l'agent 1 et inversement. La connexité du graphe doit alors être restaurée en créant des liens d'accointance "artificiels" entre les agents. Nous qualifions ces liens d'artificiels car leur raison d'être tient simplement à la connexité du graphe d'accointance alors que les liens originels signifient que deux agents ont des complémentarités et une représentation de l'autre motivée par leur potentiel de coopération. Nous appelons ces nouveaux arcs du graphe les *liens de robustesse*, et nous nommons les arcs originels les *liens de complémentarité.*

Ces liens de robustesse doivent exister avant la défaillance d'un agent car ils ne peuvent être établis automatiquement à partir d'un graphe d'accointance non connexe. Pour prévoir la disparition d'une de ses accointances, chaque agent doit connaître les accointances de ses accointances par des liens de robustesse (cette règle n'est bien entendu pas récursive car cela impliquerait qu'un agent connaît tous les agents du SMA). Sur le graphe d'accointance, il doit donc exister un lien de robustesse entre chaque agent séparé par deux liens de complémentarités. La figure 6.2 reprend notre exemple en y faisant apparaître les liens de robustesse.

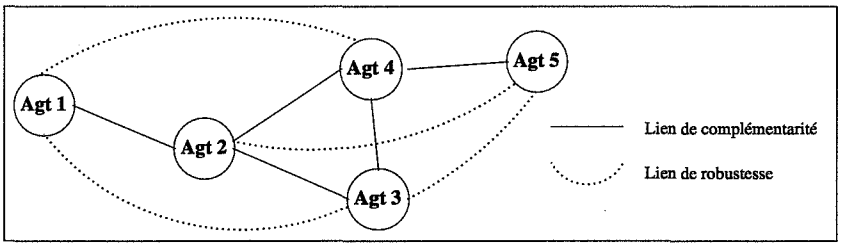


**Figure 6.2 :** Graphe d'accointance avec des liens de robustesse

Un lien de robustesse représente le fait qu'un agent connaisse uniquement l'adresse de l'agent auquel il est relié. Cette connaissance n'est utilisée que pour restaurer la connexité et n'est pas prise en compte pendant une recommandation. La création des liens de robustesse a lieu à l'issue de l'intégration d'un agent. Dès que celui-ci se considère comme intégré, il créé des liens de robustesse avec les agents qui restent dans sa liste d'agents recommandés (en les en informant pour que le lien soit bilatéral). De plus, il demande aux agents qui lui ont été recommandés par descriptions et avec lesquels il n'a pas effectué de présentation leur liste d'accointance pour créer des liens de robustesse avec ceux-ci. Enfin, il ne lui reste plus qu'à mettre en contact ses propres accointances (sauf s'il sait qu'ils se connaissent) pour qu'ils créent des liens de robustesse pour parer à sa propre défaillance éventuelle.

La détection de la disparition d'un agent diffère selon qu'elle soit issue d'un retrait volontaire ou d'une défaillance. Avant de quitter le SMA, un agent le signale à ses accointances pour qu'ils agissent de manière à conserver la connexité du graphe d'accointance. Si l'agent subit une panne, celle-ci ne

peut être détectée que si un message vers cet agent reste sans réponse. On peut alors imaginer que les agents "testent" à intervalles réguliers la validité de leurs accointances. Quelqu'en soit sa cause, la disparition de cet agent aura pour conséquence la transformation des liens de robustesse reliant deux de ses précédentes accointances en liens de complémentarités. La figure 6.3 illustre l'état du graphe d'accointance après une défaillance de l'agent 2.

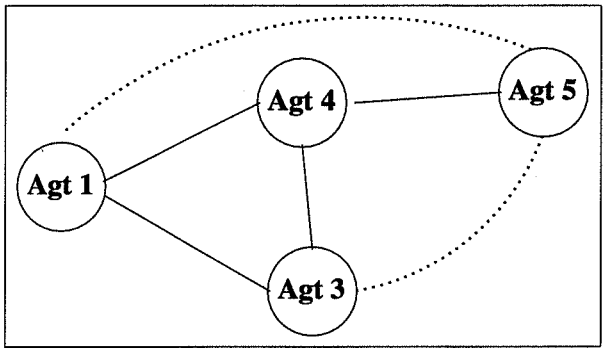


**Figure 6.3 :** Graphe d'accointance après une panne

La transformation des liens de robustesse n'est pas suffisante. Il faut ensuite créer de nouveaux liens de robustesse issu des nouveaux liens de complémentarités. Lors du passage d'un lien de robustesse à un lien de complémentarité, les deux agents concernés s'échangent leurs accointances pour créer ces nouveaux liens de robustesse (dans l'exemple de la figure 6.3 un nouveau lien est créé entre l'agent 1 et l'agent 5). Cette méthode permet de restaurer la connexité du graphe suite à la panne d'un agent mais ne gère pas forcément la panne simultanée de plusieurs agents.

### 6.2.2 Sécurité

L'ouverture d'un système et la sécurité d'accès à ses diverses fonctions sont deux propriétés difficilement conciliables. En effet, l'objectif de l'ouverture est d'apporter automatiquement des informations sur les compétences et besoins des agents du système pour les utiliser lors d'une coopération alors que la sécurité s'occupe au contraire de cacher (ou d'interdire l'accès à) certaines fonctions pour qu'elles ne soient pas exploitables par n'importe quel autre agent.

Une première possibilité de conciliation est de distinguer les compétences et besoins réels d'un agent de ceux qu'il présente dans sa description d'agent. Celle-ci ne regroupe que les descripteurs qu'il souhaite partager au sein d'un SMA et ne fait pas apparaître certaines compétences ou buts qu'il souhaite garder pour lui-même. On peut ainsi imaginer qu'un agent soit intégré à plusieurs SMA avec des descriptions d'agent différentes.

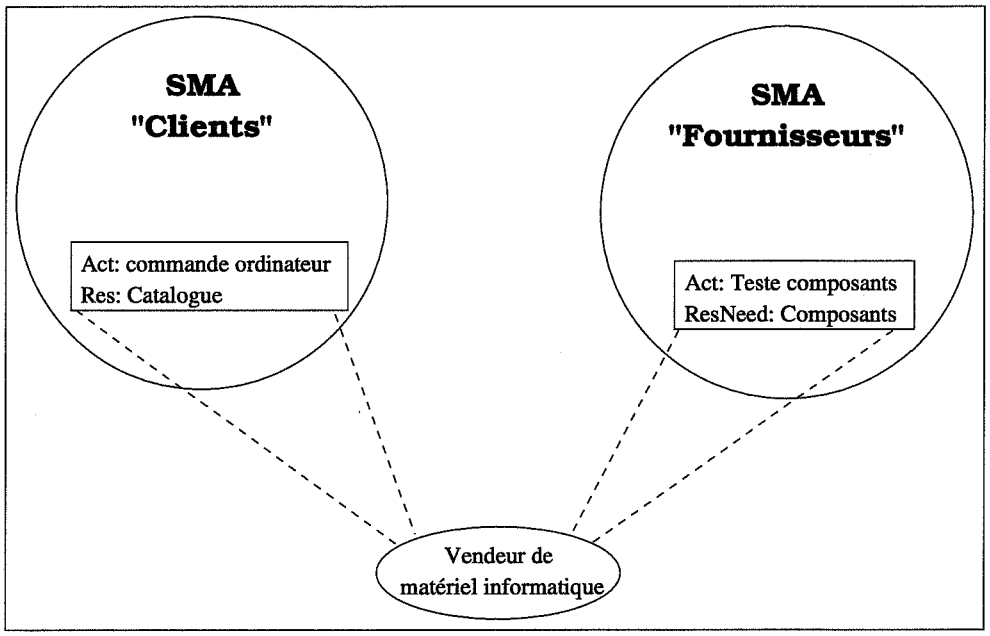


**Figure 6.4 :** Exemple de descriptions d'agent multiples

Un exemple typique, donné par la figure 6.4 est celui d'un agent représentant une entreprise disposant de fournisseurs et de clients. Dans notre exemple, un vendeur de matériel informatique est intégré par un agent le décrivant dans un SMA auquel appartiennent d'autres vendeurs de même type ainsi que des clients, et dans un autre SMA regroupant ses fournisseurs en composants informatiques. Cet agent s'est intégré dans le SMA "clients" avec des descripteurs mis à disposition des autres agents tels que son catalogue ou l'action de commander un ordinateur. Dans le SMA "fournisseur", il fait apparaître d'autres descripteurs tels qu'un besoin en composants ou qu'une action testant la performance de composants. Il peut également avoir d'autres compétences (comme assembler un ordinateur à partir de composants) qu'il garde pour lui-même.

Cette sélection au moment de la création d'une description d'agents permet de contrôler la mise à disposition de compétences pour un SMA sans discrimination entre les différents agents d'un même SMA. Si l'on souhaite une telle discrimination, il faut alors se placer au niveau des classes de descripteurs plutôt qu'à celui de la description d'agent. Nous proposons ici quelques pistes pour gérer la sécurité d'accès à des descripteurs :

- **Le cryptage d'information pendant une présentation :** Un agent crypte certains de ses descripteurs avant de les présenter. Pour le comprendre, l'agent récepteur du message doit posséder la clé de cryptage pour le décoder et reconnaître la classe de descripteurs présentée. Seuls

les agents ayant cette clé (par exemple les agents issus d'une même entreprise ou groupement d'entreprise) peuvent introduire le descripteur dans leur représentation de l'autre. Ceux qui ne l'ont pas n'ont pas d'autre choix que d'ignorer cette partie du message (mais ils peuvent construire une description d'agents à partir d'autres descripteurs non sécurisés et présentés en clair).

- **Des étiquettes de sécurisation sur des classes de descripteurs :** Certaines classes de descripteurs sont étiquettées comme sécurisées, ce qui signifie qu'elle ne peuvent pas être utilisées comme descripteur de n'importe quel agent. Les agents ayant un besoin sur ce descripteur ne demanderont qu'à des agents dont ils sont "sûrs", de satisfaire leur besoin. Des agents peuvent être qualifiés de sûrs soit parce qu'ils présentent des caractéristiques spéciales (comme de s'exécuter sur la même machine ou le même réseau local que l'agent ayant le besoin), soit parce qu'une énumération d'identifiants ou d'adresses d'agent, précisée avec la classe de descripteurs, indique les agents autorisés à utiliser une compétence de cette classe. Par exemple, on peut considérer qu'une action consistant à vérifier un mot de passe provient d'une classe sécurisée. Toutes les classes spécialisant une classe sécurisée héritent de ces contraintes de sécurité et ne peuvent que les restreindre.

Ces précautions se situent pendant les présentations et leur motivation est de ne pas informer l'autre agent d'une compétence ou d'un besoin. Bien sûr, si un agent ne communique pas d'informations sur un de ses descripteurs, il ne donnera pas de recommandations à propos des descripteurs concernés.

Il faut toutefois rappeler qu'informer n'est pas coopérer. Le fait qu'un agent communique une de ces compétences à un autre agent ne veut pas dire qu'il la propose librement à cet autre. Il peut lui refuser par la suite toute coopération, s'il ne juge pas cet agent sûr. Une autre possibilité est de ne pas coopérer gratuitement mais qu'une coopération ne soit initiée qu'à partir du moment où tous les agents y trouvent leur compte comme c'est le cas avec la théorie des dépendances ([Sichman, 1995]).

Ces dernières sections ont apportées des précisions à notre approche distribuée nous assurant de sa bonne gestion de l'ouverture et de la robustesse du SMA. Dans la section suivante, nous reprenons les différentes approches présentés pour les comparer et souligner les avantages et inconvénients de chacune.

## 6.3 Comparaison des différentes approches de l'ouverture

La distinction entre intégration totale et partielle décline notre proposition en deux approches distribuées de l'ouverture d'un SMA. La distribution de la tâche d'ouverture apporte une plus grande robustesse au système ne dépendant pas du bon fonctionnement d'une entité centrale. Nous pouvons désormais comparer nos propositions aux systèmes existants décrits dans le chapitre 3 en fonction de plusieurs critères énoncés ci-dessous :

- **Le coût en communications :** Nous l'exprimons en nombre d'échanges de messages (un message et sa réponse). Il existe trois situations de communication dont le coût dépend de l'approche adoptée : (i) *l'intégration* d'un agent nécessite un certain nombre de messages pour que l'agent fasse partie intégrante du SMA; (ii) *la coopération* qui est une situation où un agent recherche un partenaire (un agent ayant un descripteur complémentaire à l'un des siens) pour coopérer; (iii) et *l'évolution*, c'est-à-dire le nombre de messages nécessaires pour que l'évolution d'un agent soit prise en compte par les autres agents.

- **Le modèle de représentation des autres :** nous étudions pour chaque approche si ce modèle est *complet* (si un agent connaît tous les agents avec qui il peut coopérer) et s'il est *pertinent* (si l'agent ne connaît des autres que des informations qui peuvent l'intéresser pour une coopération). Ces caractéristiques sont importantes car elles permettent de distinguer une représentation des autres judicieuse d'une qui comporterait des informations inutiles. Un coût peut également être attaché à ces deux sous-critères : une représentation des autres qui n'est pas complète peut nécessiter la recherche de nouveaux partenaires de coopération et entraîner de nouvelles communications pour cette recherche alors qu'une représentation des autres non pertinente implique un coût de stockage pour des informations inutiles ainsi qu'un coût en raisonnement pour faire le tri et ressortir les informations pertinentes.

- **La tolérance aux pannes :** Un système, pour être qualifié de robuste, doit supporter le dysfonctionnement d'un de ses éléments. L'arrêt spontané d'un agent ne doit pas avoir pour conséquence une défaillance globale du SMA (telle que la perte de son ouverture ou l'impossibilité d'établir des coopération entre agents) et doit se limiter à l'indisponibilité des compétences de l'agent défaillant.

Le tableau 6.1 décrit les caractéristiques des approches évoquées précédemment suivant ces critères de comparaison.

**Table 6.1 :** Comparaison des différentes approches de l'ouverture

| | | Coût en communications | | | Représent. des autres | | Tolérance aux pannes |
|---|---|---|---|---|---|---|---|
| | | **Intégration** | **Coopération** | **Evolution** | **Complète** | **Pertinente** | |
| Distribuée | **Présentation diffuse** | $\|\Omega\|$[a] | 1 | $\|\Omega\|$ | Oui | Oui | Oui |
| | **type Gnutella** | 1 | $\leqslant \|\Omega\|$ | 0 | $\emptyset$ | $\emptyset$ | Oui |
| | **Intégration totale** | $\leqslant \|\Omega\|$ | 1 | $\leqslant \|\Omega\|$ | Oui | Oui | Oui[b] |
| | **Intégration partielle** | $\leqslant \|\Omega\|$ | 1 | $\leqslant \|\Omega\|$ | Non | Oui | Oui[b] |
| Centralisée | **Broker** | 1 | 2 | 1 | Broker: Oui | Non | Non |
| | | | | | Agent: $\emptyset$ | $\emptyset$ | Oui |
| | **Broker et mémorisation par les agents** | $\leqslant \|\Omega\|$ | 1 ou 2 | $\leqslant \|\Omega\|$ | Broker: Oui | Non | Non |
| | | | | | Agent: Non | Oui | Oui |
| | **Brokers hiérarchiques** | $\leqslant \|B\|$[c] | $\leqslant \|B\| + 1$ | $\leqslant \|B\|$ | Broker: Oui | Non[d] | Non |
| | | | | | Agent: $\emptyset$ | $\emptyset$ | Oui |

[a] $|\Omega|$ est le nombre d'agents du SMA
[b] Moyennant quelques ajouts à la connaissance
[c] $|B|$ est le nombre de brokers
[d] Même si chaque broker regroupe une spécialité de compétences, l'ensemble de ces compétences n'est pas forcément pertinente à un agent

On retrouve distinctement le clivage approche centralisée / approche distribuée dans ce tableau. En effet, il apparaît nettement que les systèmes utilisant un broker sont moins coûteux en communication quelque soit la situation considérée. Dès lors que la représentation des autres est centralisée dans un broker (ou une hiérarchie de brokers), un seul message est nécessaire pour ajouter ou modifier une description d'agent, ce qui n'est pas le cas lorsque l'information est disséminée dans les différents agents du SMA. Dans le cas où les agents mémorisent les réponses antérieures qu'a pu leur donner le broker (ils gardent une représentation de certains agents), les échanges de messages peuvent atteindre le nombre d'agents présents dans le SMA pour mettre à jour la mémoire de chaque agent. Ce grand nombre de messages reste cependant très rare car il faudrait que chaque agent ait préalablement manifesté un intérêt pour une même compétence et qu'un agent ayant cette compétence intègre le système ou évolue.

En contrepartie des faibles coûts de communication, les approches centralisées sont vulnérables à la défaillance du broker (ou d'un des brokers) qui est indispensable à l'ouverture du SMA et à la coopération entre agents. De plus, il dispose de l'ensemble de la représentation des autres, ce qui nécessite une grande capacité de stockage d'information et un certain coût de recherche pour ressortir une information demandée. Les brokers hiérarchiques organisent cette connaissance pour faciliter son accès mais cela implique des échanges de messages supplémentaires.

Les deux premières approches distribuées compensent l'absence de broker soit en augmentant considérablement les communications pour une intégration ou une évolution d'agent, soit en supprimant toute représentation des autres entraînant ainsi de forts coûts de communication pour initier une coopération. Notre proposition se situe entre ces deux extrêmes. Un point fort de notre approche est que les agents n'ont pas de connaissances *a priori* inutiles sur les autres (une information peut n'être jamais exploitée mais, si elle est connue, c'est qu'elle est considérée comme potentiellement utile pendant l'intégration d'un agent). Leur représentation des autres est pertinente et complète (dans le cas d'une intégration totale) et n'implique pas de raisonnement inutile ou de tri préalable des connaissances. L'acquisition de cette représentation nécessite cependant de nombreux échanges de messages pour l'intégration ou l'évolution d'un agent, mais ce coût est plus faible que pour la présentation diffuse car les agents sont dirigés vers ceux qui leur sont pertinents notamment par l'utilisation de stéréotypes d'agents. L'ensemble de ces stéréotypes est une connaissance supplémentaire à gérer pour chaque agent mais qui n'entre pas en compte dans le coût de raisonnement d'un agent pendant une coopération car ceux-ci ne sont utilisés que pour l'ouverture du système.

Enfin, l'avantage majeur des approches distribuées est leur robustesse. Etant donné que tous les agents prennent part à l'ouverture du SMA sans que l'un soit plus important que l'autre et sans que chacun y soit indispensable,

le dysfonctionnement d'un agent n'entraîne pas plus de conséquences que la perte de ses compétences alors qu'une panne d'un broker a de graves conséquences.

Les différences entre les approches centralisées et distribuées sont des conséquences de l'assignation des tâches d'ouverture à un ou plusieurs agents mais font appel à des fonctions similaires. Nous nous sommes intéressés dans la section suivante à la généralisation de notre modèle d'agent accueillant pour pouvoir implanter l'ouverture de manière centralisée ou gérer des problèmes proches comme la fusion de deux SMA.

## 6.4 Vers une approche générale de l'ouverture

Notre définition de la facette accueillante permet à un agent de gérer une partie de l'ouverture du SMA. En décrivant notre approche distribuée, nous avons considéré que les agents étaient homogènes dans la gestion de l'ouverture car il n'y a pas d'agent qui ait plus d'importance ou de responsabilité dans l'exécution d'une tâche liée à l'ouverture.

Il est possible de "déséquilibrer" ces responsabilités en définissant des facettes accueillantes différentes pour des agents d'un même SMA. Nous proposons dans cette section un modèle de facette accueillante pour des brokers, puis nous nous intéressons à la définition d'agents pour la fusion de deux SMA.

### 6.4.1 Gestion par brokers

Une contrainte qui doit être apportée à notre approche pour gérer l'ouverture par un broker est que celui-ci doit être le point d'entrée de tous les agents à intégrer au SMA alors que ce point d'entrée reste libre dans un SMA dont l'ouverture est distribuée. Nous avons vu précédemment plusieurs utilisations possibles d'un (ou de plusieurs) broker(s) et nous présentons la description d'agent du broker dans chacun de ces cas.

#### Un broker pour l'intégration

Le broker sert uniquement de point d'entrée pour les agents à intégrer. Chaque nouvel agent lui envoie alors sa description et le broker lui renvoie l'ensemble des agents qui lui sont complémentaires. Les agents n'utilisent plus par la suite le broker sauf pour lui signaler leur départ ou leur évolution. La facette accueillante de ce type de broker ne nécessite pas d'aménagement particulier excepté que sa propre description d'agent du broker ne comprend que trois besoins : un besoin sur la classe racine de l'arborescence de buts, un sur la racine de l'arborescence de tâches et un sur la racine de l'arborescence de ressources.

Le broker suit alors le même comportement que n'importe quel agent accueillant : chaque agent lui présentera l'intégralité de sa description d'agents (car toutes les compétences sont complémentaires à l'un des besoins du broker et tous les besoins sont plus spécifiques que l'un de ces besoins) et, à partir de ces connaissances complètes sur tous les agents, il pourra recommander directement tous les agents complémentaires au nouvel arrivant.

### Un broker pour la coopération

Dans ce cas, le broker est un intermédiaire pour toute coopération. Les nouveaux agents se présentent au broker mais ne reçoivent pas de réponses. Le protocole de présentation se résume donc à l'envoi de la description d'agent de l'agent arrivant. C'est au moment où un agent cherche un partenaire pour une coopération qu'il demande au broker quels sont les agents répondant aux caractéristiques qu'il recherche. Celui-ci lui renvoie alors la liste des agents pertinents.

Pour que les agents lui envoient leur description d'agent complète, ce broker présente les trois mêmes besoins que dans le cas précédent. Les autres agents du SMA doivent connaître un plan supplémentaire par rapport à la facette accueillante pour une approche distribuée. En effet, les agents doivent avoir un moyen de demander au broker de trouver un agent ayant certains descripteurs et de lui transmettre une proposition de coopération. Un exemple de plan est donné par la figure 6.5.

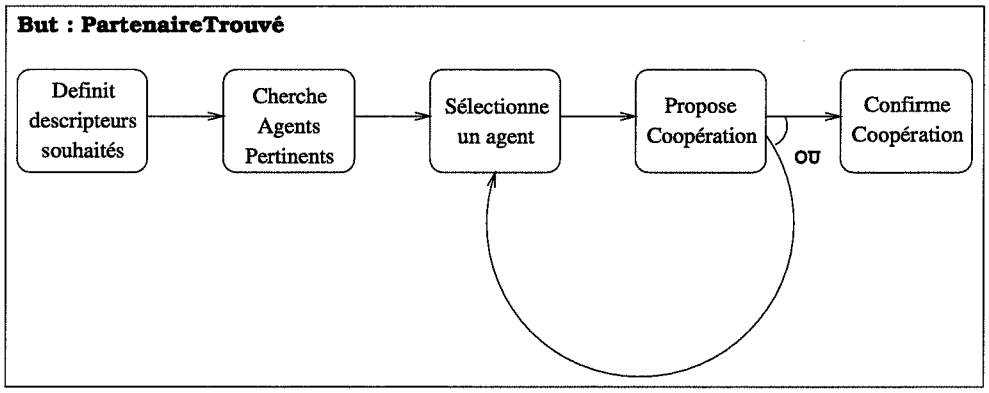


**Figure 6.5 :** Un plan de recherche de partenaires

Ce plan peut prendre différentes formes (selon la stratégie de formation de coopération choisie) mais doit faire figurer une tâche de classe *Cherche Agents Pertinents*. Ainsi quand un agent cherche un partenaire de coopération, il active ce plan qui le conduira à un moment donné à demander au broker d'effectuer son action de la classe *Cherche Agents Pertinents* en lui fournissant les descripteurs qu'il recherche. Le broker lui renvoie alors la

liste des agents susceptibles de coopérer avec lui.

### Une hiérarchie de brokers

L'utilisation de plusieurs brokers est envisageable dans l'un ou l'autre des types de brokers évoqués ci-dessus. Un broker "racine" est défini comme s'il était seul (avec la même description que ci-dessus). Les autres brokers sont dédiés à une spécialité. Cette spécialité se traduit :

- Pour des brokers dédiés à l'intégration : Les brokers "spécialistes" ont des besoins correspondant à des classes de descripteurs représentant leurs spécialités. Par exemple, si un broker doit connaître toutes les ressources des agents, il aura un besoin sur la racine de l'arborescence de ressource, si un autre doit connaître tous les agents ayant un descripteur d'une classe de tâches $t*$ ou plus spécifique, il aura un besoin sur $t*$, ...

- Pour des brokers dédiés à la coopération : leurs besoins sont définis comme ci-dessus mais ils ont en plus une spécialisation de la classe d'actions *Cherche Agents Pertinents* dédiée au traitement de leur spécialité de descripteurs.

Une hiérarchie de brokers nécessite cependant une légère modification dans le comportement accueillant de chaque broker. Pour éviter que les descriptions d'agent ne soient dupliquées dans plusieurs brokers (le broker racine par exemple centraliserait l'ensemble des descriptions d'agents, rendant les autres brokers inutiles), un broker ne garde des descripteurs sur un agent que si ceux-ci ne sont pas présents dans un broker qui lui est plus spécifique. Cet aménagement permet de répartir la représentation des autres dans les différents brokers.

D'une manière générale, la facette accueillante des agents peut être allégée. En effet, dès lors qu'un broker s'occupe entièrement de l'intégration des nouveaux agents, les autres agents sont libérés de cette charge et n'ont plus à effectuer de recommandations. Il n'est alors pas utile de construire et de maintenir des stéréotypes des autres agents.

Il est intéressant de constater que la facette accueillante est également adaptée à la définition de broker car cela permet d'imaginer des SMA passant de l'une à l'autre approche. Par exemple, l'ouverture d'un SMA pourrait être gérée de manière distribuée pour disposer d'un système robuste, et si, pour une courte période, on doit intégrer de nombreux agents, on peut passer à une approche centralisée, en intégrant d'abord un broker, pour limiter les coûts de ces intégrations massives, puis revenir à une gestion distribuée en retirant le broker du SMA.

### 6.4.2 Fusion de deux SMA

Une centralisation temporaire de l'ouverture est également souhaitable si l'on souhaite fusionner deux SMA. On peut intégrer successivement chaque agent d'un SMA dans l'autre, mais on peut aussi opter pour une solution de type broker consistant à définir temporairement un agent représentant chaque SMA. La fusion des SMA se déroule alors en quatre étapes :

1. La définition d'un agent pour chaque SMA représentant les besoins et compétences des agents de son SMA;
2. Une présentation entre ces deux agents;
3. La mise en relation d'agents de SMA différents suivant leur complémentarité;
4. La suppression des agents temporaires représentant un SMA.

Un agent représentant un SMA est créé puis intégré au SMA qu'il représente comme un broker simple, c'est-à-dire qu'il se décrit par des besoins sur les racines des arborescences de buts, tâches et ressources. Dès qu'il est intégré il dispose alors d'une description de chaque agent de son SMA. La deuxième étape consiste en une présentation avec un autre agent représentant un autre SMA. Lors de cette présentation, les deux agents temporaires ne présentent pas leurs trois besoins mais l'union des descriptions d'agent qu'ils connaissent. De cette manière, chacun présente l'ensemble des compétences et des besoins présents dans son SMA et chacun se construit une représentation de l'autre SMA.

La mise en relation des agents de chaque SMA s'effectue après les recommandations. Les recommandations se limitent à des recommandations directes. Les autres recommandations n'ont pas d'intérêt car elles doivent initier de nouvelles présentations qui n'ont pas de justification dans ce cas précis. A l'issue de leurs présentations, chaque agent transmet les recommandations reçues aux agents concernés (qui ont un descripteur ayant entrainé la recommandation) dans son propre SMA. Si les deux agents représentant un SMA transmettent ces informations, il n'est même pas besoin que les agents de chaque SMA communiquent car ils ont déjà toutes les informations nécessaires. Les agents créés pour cette fusion peuvent alors être détruits.

La contrainte de compatibilité des arborescences de classes est essentielle pour la fusion de SMA. Lors de la création d'un agent, il est assez simple de s'y plier mais quant il s'agit d'un autre SMA, celui-ci s'est construit indépendamment de l'autre et il est très probable que les arborescences de classes divergent sur certains points. Dans ce cas, il faut que les arborescences de chaque SMA soient rendues compatibles soit en en modifiant une, soit en en créant une nouvelle satisfaisante pour chaque SMA. La création d'une arborescence commune peut se faire en ajoutant dans chaque SMA

des nouvelles classes devant appartenir à cette arborescence et en faisant évoluer les descripteurs des agents vers ces nouvelles classes.

## 6.5 Conclusion

Le contexte dans lequel évolue un SMA est primordial dans le choix d'une approche pour son ouverture. Si les communications sont coûteuses ou peu fiables, il vaut mieux se reposer sur un broker centralisant la représentation des autres. Par contre, si les agents sont sujets à de nombreuses pannes, l'approche distribuée est préférable pour éviter qu'une défaillance locale ait de graves répercussions globales.

Nous avons traité dans ce chapitre plusieurs problèmes généraux qui se posent dans un SMA et plus particulièrement s'il est ouvert. En complément de notre définition de la facette accueillante, nous avons proposé des spécialisations permettant de centraliser notre approche, de fusionner des SMA ou de supporter la panne d'agents. Nous avons également donné quelques pistes pour introduire des notions de sécurité dans l'accès aux descripteurs des autres agents. Mais plus la sécurité est grande, plus l'ouverture du SMA est réduite, si bien que ces deux propriétés semblent presque contradictoire. En effet, le système le plus sûr n'est-il pas un système fermé ?

Les systèmes multi-agents sont souvent considérés comme des outils pour un contexte d'application, exploitant leurs propriétés pour en construire une simulation ou y apporter une aide à la prise de décision, à la négociation, ... L'ouverture est également une propriété qui peut être transmise à un système informatique encapsulé par un SMA ouvert. Nous avons proposé, dans [Vercouter et al., 2000], un modèle d'ouverture d'un système d'information distribué en utilisant un SMA ouvert. Nous détaillons cette application dans le prochain chapitre.

# Chapitre 7

# Implantation et exemple d'application

Nous avons développé et appliqué notre modèle d'agent accueillant à l'encapsulation de systèmes informatiques distribués. Cette encapsulation se retrouve au niveau des agents, chacun d'entre eux ayant le contrôle d'une partie du système distribué. Ce système peut alors bénéficier des fonctionnalités propres à un SMA.

Si les agents du SMA sont accueillants, on peut "ouvrir" le système encapsulé. Les agents représentent une couche descriptive de ou des entités ou fonctions dont ils ont la charge et ont une représentation d'autres agents (donc d'autres entités ou fonctions). Le comportement accueillant des agents permet alors l'ajout, le retrait ou l'évolution de nouvelles entités ou fonctions et gère la mise à jour des différentes représentations pour garder la cohérence de relations entre différentes entités ou fonctions.

Dans ce chapitre, Nous commençons par présenter l'implémentation de notre proposition et son utilisation au sein de la plate-forme multi-agent développée par l'équipe SMA de l'Ecole des Mines de SaintEtienne. Ensuite, nous expliquons comment les modules accueillants de cette plate-forme permettent la conception de SMA ouvert et leur application à un système distribué en illustrant notre approche sur l'exemple des systèmes d'information distribués.

## 7.1 Implémentation

Le modèle d'agent accueillant a été implémenté comme un modèle social partiel d'agent sur la plate-forme MAST (*Multi Agent System Toolkit*) [Boissier et al., 1998] en cours de développement dans l'équipe SMA de l'Ecole des Mines de Saint-Etienne. Nous présentons d'abord cette plate-forme avant de nous intéresser plus particulièrement aux modèles partiels d'agent disponibles et plus particulièrement à la facette accueillante.

### 7.1.1 Présentation de la plate-forme MAST

La plate-forme MAST est un environnement de développement et de programmation multi-agent, écrit en Java, et dont l'objectif est de fournir un environnement de conception d'applications multi-agents. L'architecture générale de MAST est présenté en figure 7.1.

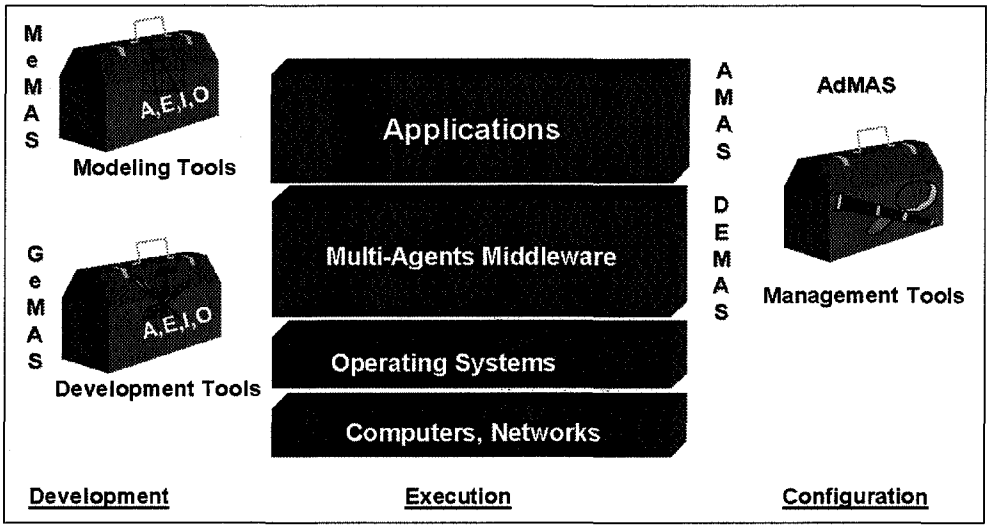


**Figure 7.1 :** La plate-forme MAST

La plate-forme est composée des cinq modules suivants :

- **DeMas** : C'est la couche basse gérant l'attribution d'identifants, les communications entre agents. Chaque agent doit se connecter à cette couche pour recevoir un identifiant et pouvoir utiliser les services qu'elle fournit.
- **AdMas** : Un ensemble d'outils d'administration et de supervision est disponible dans cette couche. Les fonctions de la couche DeMas peuvent être observées et controlées par ces outils.
- **GeMas** : Cette boite à outils comprend des modèles d'organisation, d'interaction et d'agents utilisés pour implémenter un agent de l'application.
- **MeMas** : Plusieurs interfaces de développement y sont regroupées pour utiliser les modèles définis dans GeMas.
- **AMas** : Les applications multi-agents sont définies à ce niveau.

La facette accueillante est définie dans la couche GeMas comme un composant des modèles d'agent proposés.

### 7.1.2 Les modèles GeMas

Dans MAST, les agents sont composés de facettes Agent, Environnement, Interaction et Organisation (approche Voyelles [Demazeau, 1997]). Chaque facette est représentée par un modèle d'Agent, d'Interaction ou d'Organisation fournit par la boite à outils GeMas. La figure 7.2 présente la composition des agents dans MAST.

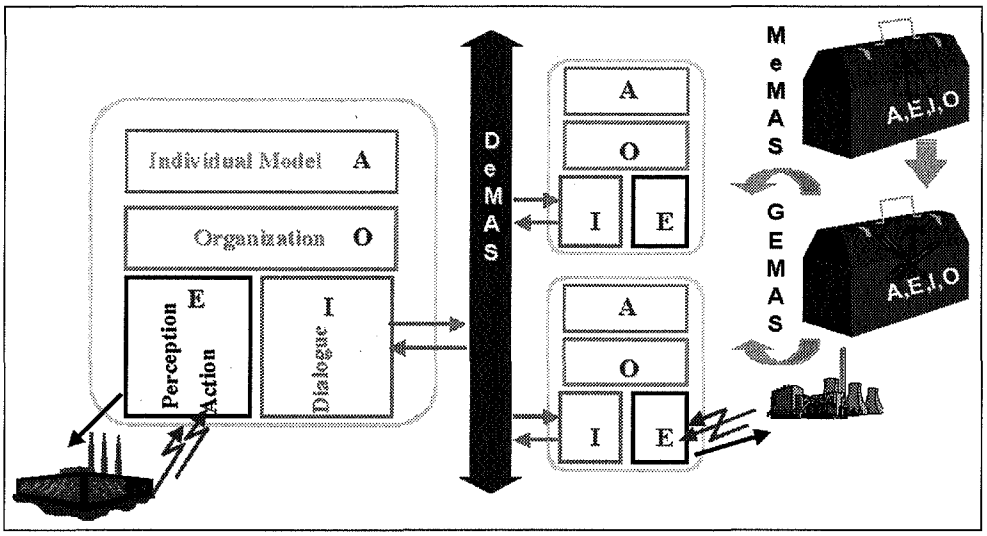


**Figure 7.2 :** Structure des agents dans MAST

La conception d'un agent avec MAST passe d'abord par la sélection de modèles des facettes qui le compose. Des interfaces de développement sont disponibles dans MeMas pour sélectionner des modèles, les spécifier ou les instancier et générer le code correspondant. A l'heure actuelle, il n'existe pas de modèle local de l'Environnement. Un agent peut être composé:

- d'une facette *Interaction.* Les agents peuvent utiliser par exemple le langage d'interaction temporel TACL [Carron et al., 1999]. La gestion de protocoles d'interaction et de conversations est également assurée.

- d'une facette *Organisation.* Des rôles et liens organisationnels sont définis suivant le modèle MOISE [Hannoun et al., 1999] ainsi qu'un moteur de calcul et de raisonnement sur les dépendances entre agents.

- d'une facette *Agent.* Un module et des connaissances d'accueil sont définis pour gérer l'intégration d'agents.

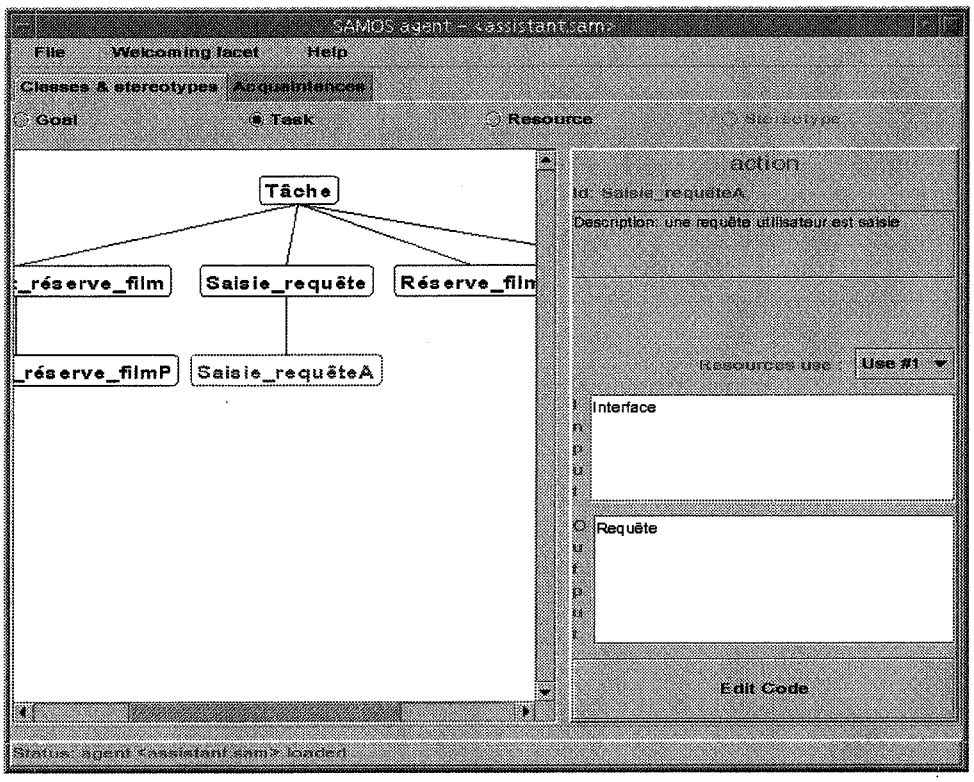


**Figure 7.3 :** Définition des classes de descripteurs

### 7.1.3 Module et connaissances d'accueil

Le modèle partiel d'agent, correspondant à la facette accueillante, est composé d'un module de connaissance et d'un module de raisonnement et de gestion des événements associés à l'ouverture du SMA. Le module de connaissance est mis à jour incrémentalement par le module de raisonnement au fur et à mesure des intégrations d'agents. Lors de la création d'un agent accueillant, le concepteur doit fournir des connaissances initiales à l'agent pour lui permettre d'assurer ses fonctions accueillantes (cf. figure 7.3).

L'agent doit connaître les classes de descripteurs qu'il utilise et sa propre description fonctionnelle. C'est la seule intervention du concepteur car les opérations liées à l'ouverture du SMA sont gérées automatiquement par la facette accueillante, ce qui les rend transparentes du point de vue des autres facettes de l'agent. L'interface représentée en figure 7.4 correspond à une étape de l'intégration d'un agent mais n'est habituellement pas présentée au concepteur. Une option de supervision du processus d'intégration nous permet néanmoins de la faire figurer ici afin de mieux comprendre le déroulement de l'intégration.

Cette fenêtre montre la représentation des autres d'un agent accueillant

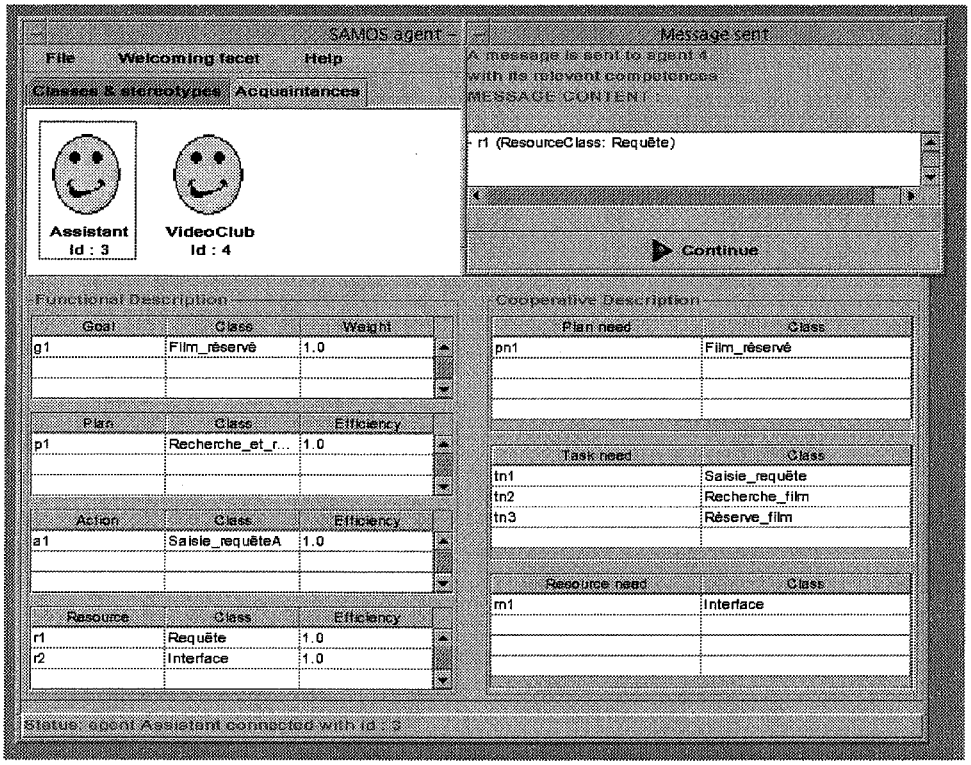


**Figure 7.4 :** Exemple d'une étape de l'intégration

"assistant" (voir les exemples du chapitre 4). Il est impliqué dans un protocole de présentation avec un agent Vidéo-Club qui lui a déjà fournit des informations sur lui-même. Ainsi il s'est contruit une représentation de cet agent et, au moment où la copie d'écran a été prise, il envoie ses propres compétences à l'agent Vidéo-Club (Ce message, représenté par la boîte de dialogue située en haut à droite, a pour contenu la description d'une ressource de classe *Requête*).

Ces interfaces, appartenant à la couche MeMas, génèrent les modules accueillants de l'agent. Comme nous l'avons mentionné précédemment, un agent est composé de plusieurs facettes instanciant chacune un des modèles disponibles. Chaque facette apporte des fonctions et connaissances mises à disposition du niveau application de l'agent (situé dans la couche AMas). Ainsi, un utilisateur de MAST peut développer un agent en le dotant des facettes qu'il souhaite et code l'utilisation et l'accès aux connaissances de ces facettes dans ce niveau application. Une facette fournit quelques services et des connaissances accessibles par le niveau application mais dispose également d'une partie cachée pour le traitement de ces connaissances ou la gestion d'événements particuliers. Par exemple, la facette Organisation four-

nit, entre autres, des informations sur les dépendances entre agents mais le processus de calcul de ces dépendances n'est pas directement accessible par l'application. De même, la facette Interaction fournit un ACL et définie des protocoles d'interaction mais elle gère automatiquement la transmission des messages ou la correspondance entre un nom d'agent et son identifant au niveau DeMas.

La figure 7.5 représente les liens d'accessibilité entre le module accueillant et le niveau application. Cette figure ne définit pas la composition entière d'un agent qui peut avoir d'autres facettes. Elle est centrée sur le module accueillant et sur ses relations avec l'application.

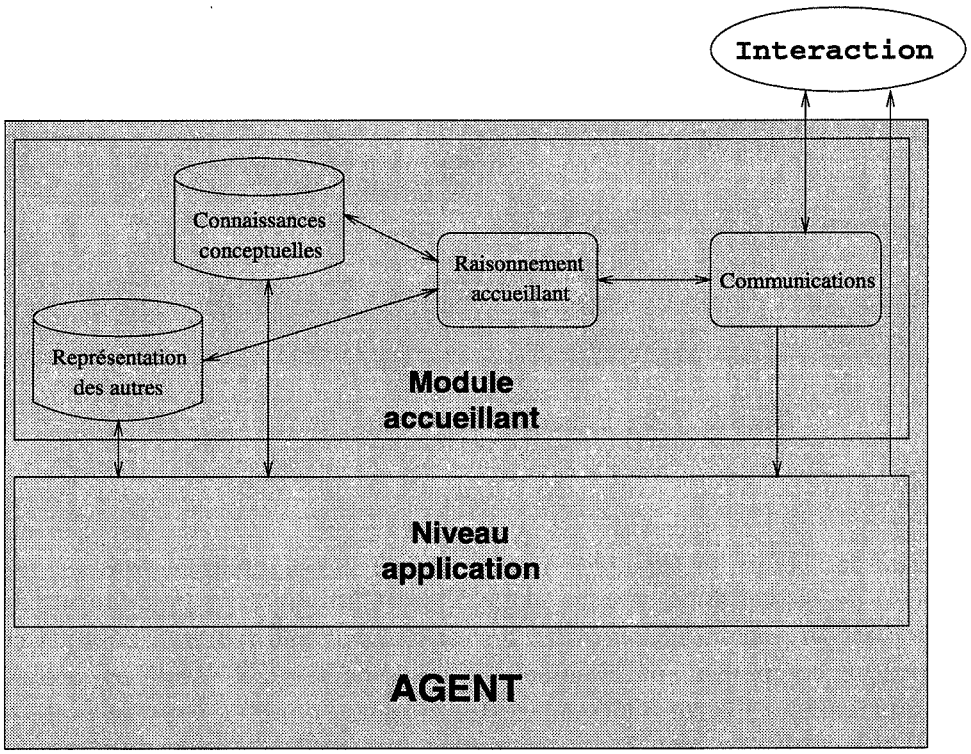


**Figure 7.5 :** Le module accueillant dans MAST

Les connaissances issues du module accueillant sont accessibles par le niveau application pour permettre un raisonnement de haut niveau sur la représentation des autres. Ces connaissances sont mises à jour par la partie cachée de ce module associée au raisonnement accueillant de l'agent. Le raisonnement accueillant se charge du traitement des messages utilisés pour un processus d'intégration et de leurs répercussions sur la représentation des autres ou l'envoi de messages.

La facette accueillante agit comme un filtre pour les interactions de l'agent. Les messages reçus au sujet d'une intégration d'un agent ne doivent

pas être transmis au niveau application, car ils ne seraient pas compris ou ils pourraient être mal interprétés. Le filtre du module accueillant transmet ces messages au raisonnement accueillant qui peut les traiter de manière appropriée et envoie tous les autres messages au niveau application. La réception d'un message symbolisée par une flèche émanant des interactions du système peut provenir d'une facette Interaction de l'agent, si elle existe, qui s'intercale alors entre la partie communications du module accueillant et l'extérieur de l'agent.

## 7.2 Domaines d'application d'un SMA ouvert

La plate-forme MAST ne privilégie a priori aucun type d'application. Un utilisateur de MAST développe une application dans la couche AMAS en tirant profit des modèles d'Agent, d'Interaction et d'Organisation de la couche GeMas.

On peut alors appliquer notre modèle d'agents accueillants à de nombreux systèmes informatiques distribués pour leur apporter une propriété d'ouverture. On peut citer plusieurs types d'applications pour lesquels l'ouverture est un réel apport :

- les systèmes distribués d'extraction de connaissances à partir de données (ECD) : différentes composantes sont associées à l'utilisation de techniques de data-mining, au stockage de connaissances liées au domaine, ... L'ouverture d'un tel système consisterait à ajouter de nouvelles techniques de data-mining ou de nouvelles connaissances, et elle permet aussi l'ajout de fonctionnalités nouvelles, telles qu'un processus d'auto-évaluation des résultats obtenus, d'apprentissage coopératif fédérant plusieurs classifieurs existants, ...

- Les systèmes de production : plusieurs machines sont ordonnées en une chaîne de production. L'ajout de machines, l'introduction de nouveaux traitements des produits, ou la modification des procédures de production sont des problèmes liés à l'ouverture du système de production.

- Les systèmes d'information distribués : ils sont composés de plusieurs sources d'information hétérogènes ainsi que de fonctions de traitement de l'information. Des systèmes d'information distribués ouverts permettent l'ajout de nouvelles sources d'information mais aussi celui de nouvelles fonctions telles que la gestion de requêtes complexes impliquant de nouvelles sources d'information, la traduction d'une ontologie vers une autre, ...

Nous avons choisi d'illustrer notre approche sur ce dernier exemple car c'est un cas d'application fréquent pour lequel les Systèmes Multi-Agents représentent un apport indéniable.

### 7.2.1 Système d'information orienté-agent

Il existe de nombreux travaux sur l'application de systèmes multi-agents aux systèmes d'information (SI) et sur le rapprochement de ces deux disciplines. G. Wagner [Wagner, 2000b] [Wagner, 2000a] distingue trois différentes approches rapprochant SMA et système d'information :

- l'extension des fonctionnalités d'un SI ou la définition d'architecture utilisant les technologies à agent;
- l'"agentification" d'un SI;
- la représentation explicite d'agents dans un SI.

Dans le premier cas, les systèmes multi-agents ne sont pas directement utilisés sinon comme source d'inspiration pour le développement de nouvelles techniques au sein d'un système d'information. De tels systèmes d'information sont dits coopératifs. Nous nous sommes plus particulièrement intéressés aux deux autres situations dans lesquels les agents sont explicitement représentés.

L'**agentification d'un SI** consiste à définir des agents que l'on assigne aux différents composants du SI distribué. Un composant peut être encapsulé dans un agent, qui a la capacité d'accéder aux informations ou d'activer les fonctions de ce composant, ou faire partie intégrante de l'agent. Nous détaillons cette approche plus loin.

La **représentation explicite d'agents dans un SI** fait cohabiter agents et objets dans un même système. L'architecture globale comprend les objets propres à un système d'information ainsi que des agents internes pouvant accéder à certains objets et accomplir certaines fonctions et des agents externes considérés comme des utilisateurs du SI interagissant avec celui-ci par des messages envoyés aux agents internes.

La plupart des travaux existants proposent une architecture combinant ces deux dernières approches. Certains agents sont définis par rapport à une entité du système d'information et cohabitent avec d'autres agents apportant d'autres services (par exemple un agent construisant des profils utilisateurs, un broker, ...). Le projet Carnot [Huhns and Singh, 1998] propose une architecture combinant des agents *Database resource* encapsulant chacun une source d'information et des agents *User interface* gérant les interfaces par lesquelles les utilisateurs formulent leurs requêtes. Ces agents interagissent avec des *brokers*, des agents *médiators* (qui décomposent les requêtes, les transmettent aux agents appropriés et coordonnent les activités des autres agents) et des agents *ontologies* (pouvant reconnaître et traduire les différentes ontologies utilisées). On peut trouver d'autres architectures du même type (tels Infosleuth [Bayardo et al., 1998] ou UMDL [Durfee et al., 1997]) définissant d'autres types d'agents et d'interactions entre agents suivant leur contexte d'application.

### 7.2.2 Un agent "système d'information"

Nous avons choisi l'approche consistant à "agentifier" un système d'information pour illustrer l'apport d'agents accueillants en termes d'ouverture du système. Nous avons accordé notre préférence à cette approche pour les raisons suivantes :

- Les composants d'un système d'information peuvent être décrits et gérés par des agents. En dotant les agents d'une facette accueillante, chaque aspect du SI peut être l'objet d'un ajout, retrait ou évolution d'une fonction ou d'une entité.
- Les agents sont définis suivant un modèle homogène regroupant plusieurs propriétés que l'on peut rapprocher de notre description fonctionnelle.
- Il n'y a pas de types d'agents ou de protocoles d'interaction prédéfinis, ce qui permet d'envisager l'ajout de fonctions non prévues initialement.

Selon cette approche, un agent a la charge d'un système d'information, et un SMA représente alors des systèmes d'informations distribués. G. Wagner a définit six caractéristiques nécessaires à un agent "système d'information" : ses **croyances/connaissances**, sa **perception**, ses **capacités**, sa **mémoire**, ses **engagements** et son **ACL (Agent Communication Language)**.

#### Croyances/Connaissances

Ce sont les informations dont dispose un agent. C'est la seule caractéristique des agents qu'on retrouve dans les systèmes d'information traditionnels. Par exemple, dans un modèle relationnel, les croyances ou connaissances d'un agent correspondent aux tuples des tables relationnelles.

#### Perception

Les perceptions d'un agent regroupent les messages qu'il reçoit d'autres agents et des événements issus de son environnement. Une perception peut activer des *règles de réaction* qui amènent un agent à entreprendre certaines actions.

#### Capacités

Les capacités d'un agent sont les actions qu'il peut effectuer. Ces actions peuvent être d'ordre communicatif (l'envoi de messages vers d'autres agents) ou physique (un moyen d'altérer l'environnement de l'agent).

### Mémoire

Un agent garde un historique de ses perceptions et actions passées sous la forme d'une mémoire. La mémoire est composée de messages reçus, des messages envoyés, d'événements perçus et des actions entreprises.

### Engagements

Un engagement implique deux agents. Il représente le fait qu'un agent va effectuer une action pour le compte d'un autre agent. Chaque agent connaît ses engagements en cours (qu'il ait à effectuer l'action ou qu'il en profite). Il existe également des engagements visant à satisfaire une condition plutôt qu'à effectuer une action.

### ACL

Les agents doivent connaître des langages communs pour s'échanger des messages et comprendre leurs contenus. Les performatifs utilisés sont de trois types : la modification d'information (performatif TELL) met à jour la connaissance d'un agent (cela correspond aux ajouts, suppressions ou modifications de tuples dans une table relationnelle), les requêtes d'information (performatif ASK) demande de l'information (comme une requête effectuée sur une table), et les requêtes d'action (performatif REQUEST) dont le but est de créer des engagements.

Parmi toutes ces caractéristiques, certaines sont intéressantes pour l'ouverture car elles décrivent des fonctions statiques alors que d'autres ont vocation à être utilisées pendant des exécutions de requêtes ou des coopérations, donc après l'intégration.

## 7.3 Utilisation des agents accueillants

Le modèle d'agent "système d'information", proposé par G. Wagner, associe les fonctionnalités d'un système d'information (des sources d'information, des méthodes de traitement de requêtes, ...) à des agents. Ce niveau d'abstraction supplémentaire permet aux agents de raisonner sur leurs interactions et sur leur propre fonctionnement.

Ainsi, en dotant les agents "système d'information" d'une facette accueillante, ils peuvent déduire leurs besoins en coopération et utiliser les différents plans, définis dans le chapitre 5, pour participer à l'ouverture du système. Nous expliquons, dans un premier temps, comment peut être définie la facette accueillante d'agents "système d'information". Puis nous nous intéressons à la conception d'un système d'information orienté-agent ouvert.

### 7.3.1 Adaptation à des agents accueillants

Parmi les caractéristiques d'un agent "système d'information" on peut distinguer deux types de caractéristiques: celles qui sont directement utilisées pour exécuter une fonction ou un service du système et celles qui assurent le bon fonctionnement de l'ensemble et coordonnent les différentes opérations.

La facette accueillante n'a besoin de représenter que les caractéristiques "durables" d'un agent. Ainsi, la mémoire, les engagements et l'ACL sont utilisés ou créés pendant une coopération d'un agent mais ne sont pas à l'origine de celle-ci. nous avons donc transcrit dans la facette accueillante les caractéristiques suivantes (cf. figure 7.6):

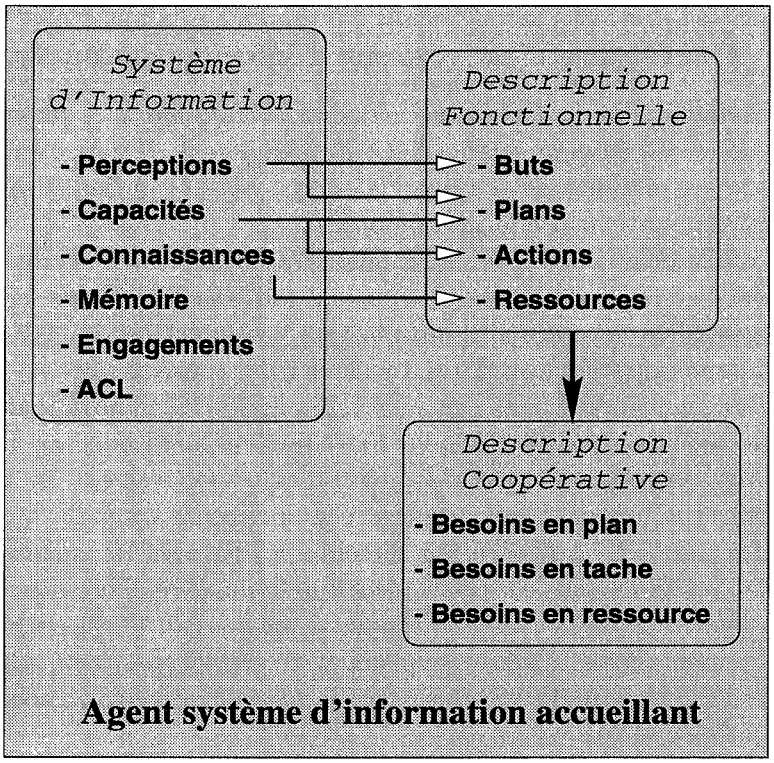


**Figure 7.6 :** relations entre système d'information et description d'agent

- Les connaissances sont représentées par des ressources dans la facette accueillante. Une classe de ressources peut correspondre par exemple à une table relationnelle et chaque instance de cette classe est un champ de cette table.
- Les capacités représentent ce qu'un agent "peut faire". C'est l'équivalent de notre notion de tâche. Ainsi, une capacité peut être traduite

par une action ou un plan.

- Les perceptions d'un agent régissent son fonctionnement. Un agent "système d'information" dispose de règles de réaction qui associe un type de perception à une série d'actions à entreprendre. Ce mécanisme est légèrement différent pour des agents accueillants : c'est l'activation d'un but qui amène un agent à chercher un plan satisfaisant ce but et à dérouler ce plan (ou à sous-traiter le déroulement de ce plan). Ainsi les règles de réaction sont transposées dans la facette accueillante sous la forme de plans et d'un but satisfait par ce plan. Il faut également que le but soit défini de manière à être activé par les mêmes événements qui déclenchaient la règle de réaction.

### 7.3.2 Conception d'un système d'information ouvert

La définition de la facette accueillante correspond à la phase descriptive mentionnée dans la section 5.2.1 du chapitre 5. Il faut encore mettre en œuvre les mécanismes d'ajout, de retrait ou d'évolution d'agents pour concevoir un système d'information ouvert.

Le concepteur a alors le choix entre une approche centralisée ou une approche distribuée de l'ouverture : soit il laisse la facette accueillante des agents gérer l'ouverture, soit il introduit un broker (cf. section 6.4.1) centralisant la représentation des autres. Ce choix a des conséquences sur le module de raisonnement accueillant des agents. La figure 7.7 montre le détail d'un agent accueillant sous MAST pour une approche distribuée de l'ouverture d'un SMA. Les figures 7.8 et 7.9 présentent respectivement les agents accueillants et le broker si le concepteur opte pour une approche centralisée.

D'une approche à l'autre, on retrouve à peu près les mêmes connaissances et fonctions mais elles sont réparties de manière différentes. Pour une gestion distribuée de l'ouverture, chaque agent connaît plusieurs représentations d'accointance et dispose de plans et de stéréotypes pour accomplir sa part dans l'ouverture du système (c'est le module accueillant présenté dans les chapitres 4 et 5)).

Avec une approche centralisée, la facette accueillante des agents est beaucoup plus "légère". Les agents peuvent connaître quelques représentations d'accointance (mais cette connaissance est facultative) et ils ne doivent connaître que quelques plans dans lesquels leurs interactions avec le broker sont définies. Parmi ceux-ci, un plan, n'apparaissant pas dans l'approche distribué, défini la recherche d'un partenaire de coopération en utilisant le broker.

Le broker centralise l'ensemble des descriptions d'agent du système et, dans son module de raisonnement accueillant, dispose de plans, accomplissant dans leur intégralité les tâches liées à l'ouverture. Il n'y a pas de connaissance sur des stéréotypes dans une approche centralisée car ceux-ci sont uti-

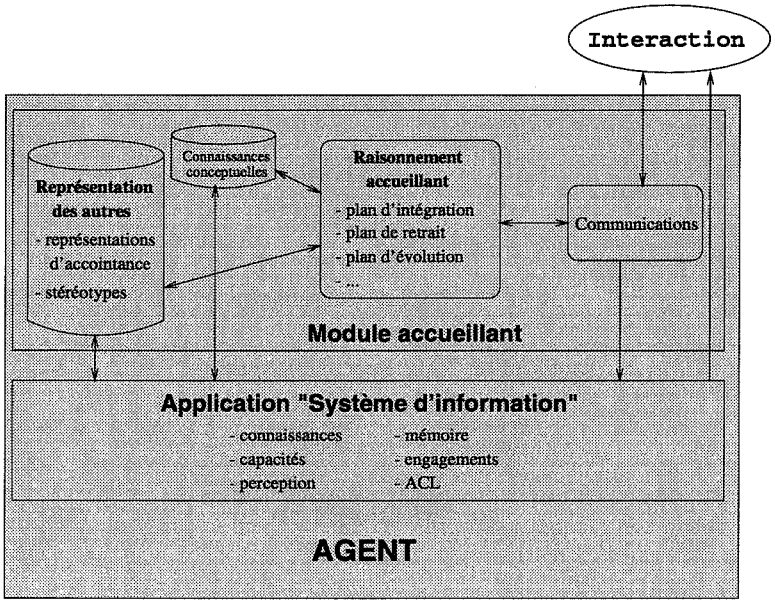


**Figure 7.7 :** module accueillant dans une approche distribuée

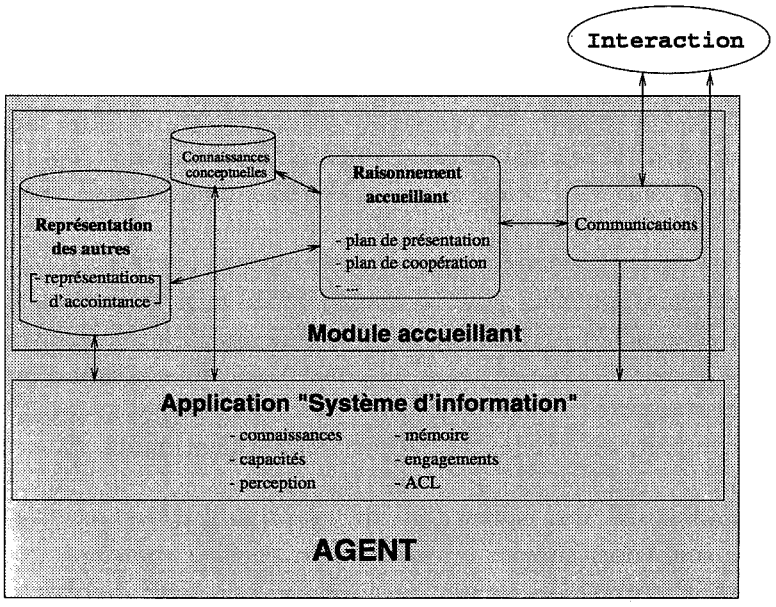


**Figure 7.8 :** module accueillant pour un agent dans une approche centralisée

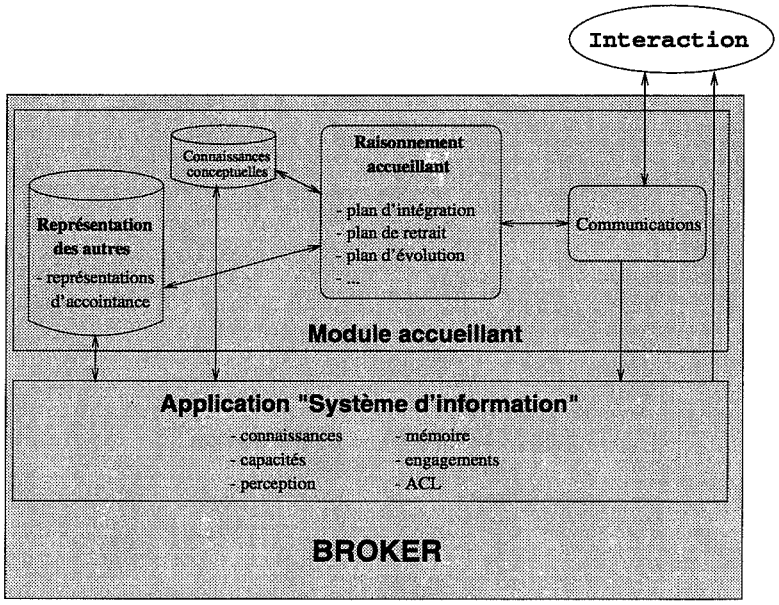


**Figure 7.9 :** module accueillant pour un broker dans une approche centralisée

lisés pour des recommandations qui n'ont pas lieu d'être si on dispose d'un broker.

Lors du développement d'un système d'information orienté-agent ouvert, le concepteur a maintenant le choix entre une approche centralisée ou distribuée de l'ouverture. La facette accueillante décrit d'une manière suffisamment générale les agents pour déduire leur potentiel de coopération (au moyen de leur description coopérative) sans tendre vers l'une ou l'autre approche de l'ouverture.

Dans les travaux existants jusqu'ici, l'approche centralisée est employée presque systématiquement pour sa simplicité de mise en œuvre. Le module accueillant disponible sous MAST met les deux approches à égalité au niveau de la mise en œuvre, et permet à un concepteur de système d'information orienté-agent ouvert d'opter pour l'une ou l'autre de ces approches en prenant en compte des critères de performance (mentionnés dans la section 6.3).

La combinaison des contraintes du système à développer et des performances souhaitées oriente le choix du concepteur dans son approche de l'ouverture. Par exemple, on optera pour une approche distribuée dans les contextes suivants :

- Les ajouts, retraits ou évolution de sources d'information ou de fonctions sont rares : la hausse du coût en communications pour maintenir l'intégration est limitée.

- Si les communications sont très coûteuses et si l'ouverture est également peu utilisée, l'approche distribuée est intéressante car elle réduit le nombre de communications pour établir une coopération (il n'y a pas de messages envoyés à un broker pour obtenir les informations nécessaires) et, dans un système d'information distribué, les interactions entre agents sont nombreuses (l'exécution d'une requête fait intervenir une interface de saisie, un planificateur, plusieurs sources d'information, ...).

- Le système est fragile, dans le sens où n'importe quel agent peut tomber en panne. En ne confiant pas l'ouverture à un seul agent, l'approche distribuée supporte la panne d'un agent et reste ouvert après celle-ci.

L'approche centralisée est plus adaptée dans les contextes suivants :

- Les sources d'informations et fonctions changent beaucoup (par ajout, retrait ou évolution de certaines d'entre elles).

- Le choix de partenaires de coopération est très simple et limité. Par exemple, si les requêtes d'un agent sont toujours dirigées vers les mêmes sources d'information, il n'est pas nécessaire de demander à chaque fois au broker quelles sont ces sources d'information. Cette information peut être initialement demandée et exploité pour plusieurs coopérations.

- Les utilisateurs doivent avoir accès à une vue générale du SMA regroupant les différentes compétences présentes à un moment donné dans le système. Un broker peut leur fournir cette information.

## 7.4 Conclusion

Dans ce dernier chapitre, nous avons présenté l'intégration de nos travaux à la plate-forme de développement d'applications multi-agents, MAST. Celle-ci met à disposition d'un utilisateur des modèles d'Agent, d'Interaction et d'Organisation utilisés pour concevoir un SMA. La facette accueillante y est représentée comme un modèle partiel d'agent apportant la propriété d'ouverture à une application.

Nous nous sommes ensuite intéressés à l'application de SMA ouverts. Parmi les domaines d'application possibles, nous avons choisi de prendre comme exemple les systèmes d'information distribués. En prenant comme base les travaux de G. Wagner sur l'agentification de systèmes d'information, nous avons expliqué de quelle manière on pouvait ajouter une facette accueillante aux agents et concevoir un système d'information orienté-agent ouvert. Les deux approches (centralisée et distribuée) de l'ouverture ont été étudiées pour déterminer les conséquences du choix de l'une ou de l'autre sur

les agents. Nous avons alors relevé des caractéristiques de contextes d'application susceptibles d'influer sur le choix d'une approche de l'ouverture.

# Chapitre 8

# Conclusion & perspectives

## 8.1 Problèmes abordés

Tout au long de ce mémoire, nous nous sommes intéressés au problème de l'ouverture dans les Systèmes Multi-Agents. Un Système Multi-Agent ouvert est évolutif dans le sens où l'ensemble des agents qui le composent n'est pas statique. Nous avons rapporté le problème de l'ouverture d'un SMA à la prise en charge de trois tâches par le système.

Celui-ci doit d'abord permettre d'ajouter de nouveaux agents. Cette opération nécessite, au-delà du simple ajout physique dans le système, d'établir des relations entre le nouvel agent et certains autres agents pour leur permettre de coopérer. La plupart du temps cette relation se traduit au moins par une représentation de l'autre décrivant ses caractéristiques qui peuvent justifier une coopération.

La seconde tâche que doit prendre en compte un SMA ouvert est le retrait d'agents qui peut déstabiliser les schémas d'interaction mis en place. Aussi, il faut gérer ce retrait et adapter les interactions entre agents.

Enfin, un SMA ouvert doit permettre l'ajout ou le retrait de compétences parmi ses agents. Cette modification locale entraîne une évolution des caractéristiques de cet agent produisant de nouvelles possibilités de coopération et en supprimant d'autres.

Ces trois tâches, que l'on regroupe sous le terme de tâche d'ouverture doivent être assurées par le SMA ouvert. Dans la quasi-totalité des travaux existants sur les SMA ouverts, les trois tâches liées à l'ouverture sont assignées à un agent, appelé broker. Celui-ci maintient une représentation de chaque agent présent dans le système. Lorsqu'un agent entre dans le SMA, il se présente au broker de manière à lui fournir une représentation de lui-même. De la même manière, si un agent quitte le SMA, il le signale au broker et si un agent évolue il notifie les changements au broker pour qu'il mette à jour sa représentation des autres.

Quand un agent cherche un partenaire pour une coopération, il envoie

une requête au broker en précisant certaines compétences requises pour la coopération. Le broker sélectionne alors les agents présentant ces compétences et envoie leurs adresses et/ou caractéristiques particulières à l'agent ayant émis la requête. Il peut aussi transmettre directement aux agents concernés la demande de coopération.

## 8.2 Notre proposition

Nous nous sommes démarqués de cette approche "classique" de gestion de l'ouverture par broker en répartissant la charge des tâches liées à l'ouverture parmi tous les agents du SMA. L'ajout, le retrait ou l'évolution d'agents sont alors traités collectivement par les agents du SMA. Suivant cette approche distribuée, nous avons défini un modèle partiel d'agent, que nous appelons la *facette accueillante* d'un agent, composé des connaissances et compétences nécessaires à la gestion collective de l'ouverture. Ainsi, un SMA ouvert *et distribué* est un système composé d'*agents accueillants* (des agents présentant une facette accueillante) dans lequel n'importe quel agent, à la perception d'un événement lié à l'ouverture du SMA, peut déclencher l'exécution collective d'une tâche assurant la prise en charge de cet événement. L'ajout d'une facette accueillante aux agents nous a amené à développer plusieurs modèles et mécanismes :

- Chaque agent dispose de sa propre **représentation des autres** dans laquelle il modélise ses accointances. Chaque représentation fait apparaître des buts et compétences d'un agent mais aussi des besoins en matière de coopération. Ainsi, un agent sait comment les autres agents peuvent l'aider dans ses activités, mais aussi comment il peut les aider. Notre modèle de représentation des autres est fondé sur la notion de **complémentarité** entre agents. La mise en relation des compétences et besoins de deux agents permet de déduire s'il existe des possibilités de coopération et, si c'est le cas, quelles sont réellement ces possibilités.

- Pour que chaque agent acquiert une représentation des autres, nous avons défini un **protocole de présentation**. Ce protocole structure une séquence de messages de manière à ce que deux agents s'échangent des informations sur eux-mêmes et se contruisent une représentation de l'autre. La caractéristique la plus intéressante de ce protocole est que chaque agent se construit une représentation **pertinente** de l'autre. Nous définissons la pertinence de représentation comme une mesure de l'exploitabilité des informations qu'elle contient. Dans le protocole de présentation, une première détection des complémentarités entre deux agents permet de filtrer les informations échangées pour ne communiquer à l'autre que ce dont il a besoin. En effet, nous ne souhaitons pas qu'un agent connaisse toutes les caractéristiques de tous les agents du

SMA. La représentation qu'un agent a des autres sera dite pertinente s'il ne connaît pas d'informations sur les autres à partir desquelles il ne peut pas déduire de possibilités de coopération ou qui ne lui sont pas indispensables pour remplir son rôle d'agent accueillant.

- En complément de ces deux premiers points, nous avons également développé un ensemble de **comportements accueillants**. Un modèle de représentation des autres et un protocole de présentation ne suffisent pas à gérer l'ouverture. Un agent accueillant doit savoir comment et avec qui utiliser le protocole de présentation, et quelles informations il peut déduire de sa représentation des autres à un moment donné, pour participer efficacement à l'ouverture du SMA. Un exemple de comportement accueillant est la **recommandation** pratiquée par un agent pour l'intégration d'un nouvel agent dans le SMA. Cette recommandation permet d'enchaîner plusieurs protocoles de présentation et dirige un nouvel agent vers les agents qui lui sont complémentaires. A côté de cet exemple nous avons également défini des comportements accueillants pour gérer le retrait d'agents, l'évolution d'agents, l'introduction de nouvelles compétences, ...

Nos travaux sur l'ouverture d'un SMA par des agents accueillants présentent l'avantage de répartir la gestion de l'ouverture et ainsi de ne pas dépendre du bon fonctionnement d'un agent centralisant cette fonctionnalité comme le fait un broker. En effet, une approche centralisée de l'ouverture présente l'inconvénient de concevoir un SMA vulnérable à la panne d'un agent. En cas de défaillance du broker, c'est non seulement l'ouverture du SMA qui est perdue mais aussi les possibilités de coopération entre les agents du SMA.

Notre objectif n'est cependant pas de prouver qu'une approche distribuée est meilleure qu'une approche centralisée. Nous en serions d'ailleurs bien incapable car chaque approche a ses avantages et ses inconvénients. Si une gestion centralisée de l'ouverture par un broker fragilise un SMA, elle est efficace tant que le broker fonctionne, peu coûteuse en temps et en nombre de communications échangées dans la gestion d'une tâche liée à l'ouverture, et facile à mettre en œuvre.

Le développement d'une approche distribuée nous a néanmoins permis de montrer qu'il existait une alternative à l'approche centralisée de l'ouverture d'un SMA. A notre connaissance, la plupart des travaux utilisant un SMA ouvert ne font pas de l'ouverture leur priorité de recherche et optent pour une gestion par broker car c'est une solution fréquemment employée et dont l'implantation est aisée.

Nous nous sommes alors intéressés au problème de l'ouverture d'une manière beaucoup plus générale en considérant les approches centralisée et distribuée comme des cas particuliers ou des instanciations de l'ouverture

d'un SMA. En généralisant notre approche, nous avons défini un certain nombre de concepts clés dans l'ouverture d'un SMA :

- Les ajouts, retraits et évolutions des agents peuvent se rapporter à un problème d'**intégration** d'un agent dans le SMA. Par intégration, nous désignons un état d'un agent par rapport au reste du SMA : un agent est intégré au SMA dès lors qu'il a des connaissances sur les autres suffisantes pour initier des coopérations avec les agents qui lui sont complémentaires. L'ajout d'un agent nécessite que l'intégration de cet agent soit créée; son retrait doit entraîner une mise à jour de la représentation que les autres agents ont de lui, pour qu'ils ne comptent plus sur des coopérations avec lui; et l'évolution d'un agent peut provoquer la perte de l'intégration de l'agent évoluant et il faut la restaurer.

- Les notions de **complémentarité** et de **pertinence** évoquées ci-dessus sont importantes dans la représentation des autres. Quelque soit l'approche choisie, il n'est pas souhaitable que tous les agents connaissent toutes les caractéristiques de tous les autres agents. La complémentarité permet de détecter si une information est utile à se représenter alors que la pertinence filtre les informations pour éliminer celles qui n'ont pas d'intérêt pour un agent donné.

- Enfin l'intégration d'un agent et l'acquisition d'une représentation des autres se fait par des échanges de messages qu'il faut structurer. Il est nécessaire de définir des **protocoles de présentation** propres à l'ouverture dont l'objectif est de fournir des informations sur les caractéristiques d'un agent à un autre agent (que l'un d'entre eux soit un broker ou non). Les protocoles de présentation s'appuient sur les notions d'intégration, de complémentarité et de pertinence pour gérer au mieux les échanges de messages et supprimer les communications inutiles.

Ces considérations générales, communes à tout SMA ouvert, nous apportent un recul suffisant pour comparer différentes approches de l'ouverture selon des critères portant sur le coût de l'ouverture, de la coopération, sur le modèle de représentation des autres et sur la robustesse du SMA. Ainsi, d'après les caractéristiques et besoins d'une application multi-agent, le concepteur d'un SMA ouvert a les moyens d'évaluer quelle approche de l'ouverture est la plus appropriée et peut définir sa mise en œuvre.

## 8.3 Perspectives

Dans la suite de nos travaux, nous allons nous intéresser à l'application de notre modèle théorique de l'ouverture d'un SMA sur un cas concret. Il

est prévu que soit développé un système multi-agent ouvert dans le cadre du projet E-Alliance, financé par la région Rhône-Alpes. Le contexte traité est celui de la gestion de plusieurs ateliers d'impression. Ces ateliers d'impression ne forment pas un ensemble homogène dans le sens où chacun peut offrir des services différents ou de qualités différentes (par exemple, un atelier permettra l'impression en couleur, un autre des impressions au format A3, ...). L'impression d'un document est une tâche complexe qui peut faire intervenir plusieurs ateliers différents (on peut exprimer des contraintes sur la qualité du papier utilisé pour certaines pages, sur l'orientation de pages, la page de couverture peut être imprimée sur du papier cartonné, ...). Des agents supplémentaires interviendront alors pour gérer le déroulement d'une tâche d'impression complexe et sous-traiter plusieurs impressions à des ateliers d'impression différents.

L'ouverture de ce réseau d'impression est intéressante car elle permet d'ajouter de nouveaux ateliers d'impression, qui peuvent apporter de nouveaux services. De plus, les agents attachés à des ateliers d'impression sont également amenés à interagir. Dans le cas où un atelier a une charge de travail trop importante, il peut sous-traiter certaines de ces tâches à un atelier proposant les mêmes services. Il est alors nécessaire que chacun de ces agents se représentent d'autres agents de manière pertinente pour qu'il sache à quel agent il doit demander une sous-traitance.

Une autre perspective de notre travail consiste à positionner nos considérations générales sur l'ouverture d'un SMA dans un contexte plus large. Notre objectif a été de décrire des critères de choix lors de la conception de SMA ouverts et de proposer des mécanismes et modèles généraux pour un SMA ouvert. Actuellement, une problématique importante de la communauté SMA est la définition de méthodologies de conception ou d'analyse de systèmes multi-agents ([Gutknecht and Ferber, 1999], [Hilaire et al., 2000], [Müller, 1998], [Wooldridge et al., 1999], ...). Sans vouloir nous attaquer à ce vaste problème, il serait intéressant de situer les choix de conception concernant l'ouverture dans une méthodologie globale. L'ouverture pourrait être considérée dès l'analyse générale du SMA, lors de la définition d'une architecture d'agent (par l'intégration d'une facette accueillante), ou des schémas d'interaction entre agents (en considérant les protocoles de présentation), ou même pendant toutes ces étapes.

# Annexes

# Mécanisme de stéréotypage par GBM

La stéréotypage est une adaptation de la formation de concept dans une Mémoire Fondée sur la Généralisation (GBM) [Lebowitz, 1986]. L'ajout d'une nouvelle instance (une représentation d'accointance) est décrite ci-dessous :

**Ajout d'une représentation d'accointance.**

1. Réception de la nouvelle représentation d'accointance
2. Une recherche sur la hiérarchie est effectuée pour trouver les stéréotypes les plus spécifiques caractérisant la nouvelle représentation d'accointance. La fonction SEARCH(racine des stéréotypes, descripteurs de la nouvelle représentation d'accointance) est appelée.
3. La représentation d'accointance est ajoutée par un appel à la fonction UPDATE(stéréotype, nouvelle représentation d'accointance) pour chaque stéréotype renvoyé par la fonction SEARCH.

La recherche des stéréotypes caractérisant la nouvelle représentation d'accointance suit la fonction suivante :

**SEARCH (STEREO, liste).**

1. La confiance accordée à chaque descripteur de *STEREO* qui est aussi présent dans *liste* est augmentée. (le stéréotype est potentiellement acceptable).
2. Y a-t-il des descripteurs de *STEREO* qui ne sont pas dans *liste* ?

   **Oui** La confiance de ces descripteurs est diminuée. La fonction renvoie NIL.

   **Non** L'algorithme continue à l'étape 3.
3. Pour chaque stéréotype $st \in sub(STEREO)$ (les représentations d'accointance de $sub(STEREO)$ ne sont pas traités ici) ayant au moins un descripteur présent dans *liste*, on rappelle la fonction *SEARCH(st, liste)*.
4. L'étape précédente a-t-elle renvoyée des stéréotypes par les appels récursifs de *SEARCH* ?

   **Oui** L'union de ces stéréotypes est renvoyée.

   **Non** STEREO est renvoyé.

L'ajout final d'une représentation d'accointance est géré par la fonction suivante :

**UPDATE (STEREO, représentation d'accointance).**

1. Pour chaque nœud $n \in sub(STEREO)$ appeler la fonction *CREATE_STEREOTYPE(n,représentation d'accointance,STEREO)*

**CREATE_STEREOTYPE (node, représentation d'accointance, STEREO).**

1. Y a-t-il des descripteurs de *node* qui sont présents dans *représentation d'accointance* ?

   **Oui** (a) Un nouveau stéréotype *st* est créé représentant l'intersection des descripteurs de *node* et de *représentation d'accointance.*

   (b) *node* et de *représentation d'accointance* sont attachés comme des spécialisations de *STEREO.*

   (c) Si *node* était une spécialisation directe de *STEREO* le lien est supprimé.

   (d) Pour chaque nœud $n \in sub(node)$ appeler la fonction *CREATE_STEREOTYPE(n,représentation d'accointance,st)*

   **Non** Pour chaque nœud $n \in sub(node)$ appeler la fonction *CREATE_STEREOTYPE(n,représentation d'accointance,STEREO)*

Nous avons adapté cet algorithme au stéréotypage en modifiant deux points précis :

- Un stéréotype n'est pas retenu comme une généralisation d'une représentation d'accointance dès lors qu'il manque à celle-ci un des descripteurs du stéréotype. L'algorithme de GBM teste une contradiction de descripteurs, pas une absence.

- La fonction *CREATE_STEREOTYPE* est un ajout spécifique pour les stéréotypes. Elle permet de construire des stéréotypes intermédiaires entre deux représentations d'accointance et n'existe pas dans l'algorithme de GBM.

# Bibliographie

[Adam et al., 1999] Adam, E., Mandiau, R., and Kolski, C. (1999). Approche holonique de modélisation d'une organisation. In *Ingénierie des systèmes multi-agents (JFIADSMA'99)*, pages 121–134, Saint-Gilles, Ile de La Réunion. Hermès.

[Agha and Hewitt, 1985] Agha, G. and Hewitt, C. (1985). Concurrent programming using actors: Exploiting large-scale parallelism.

[Austin, 1962] Austin, J. L. (1962). How to do things with words. Oxford University Press.

[Axelrod, 1984] Axelrod, R. (1984). The evolution of cooperation. Basic Books, New York.

[Bakam, 1998] Bakam, I. (1998). Modèles multi-agents pour la gestion de ressources renouvelables : vers un couplage simulation et approches formelles de modélisation. In *Modèles et systèmes multi-agents pour la gestion de l'environnement et des territoires*, pages 417–432, Clermont-Ferrand, France. Cemagref.

[Barbuceanu and Fox, 1995] Barbuceanu, M. and Fox, M. S. (1995). Cool: a language for describing coordination in multi-agent systems. In *Proceedings of the First International Conference on Multi-Agent Systems*, pages 17–24, San Francisco, California. AAAI Press/The MIT Press.

[Basu et al., 1998] Basu, C., Hirsh, H., and Cohen, W. (1998). Recommendation as classification: Using social and content-based information in recommendation. In *AAAI-98*, pages 714–726, Menlo Park, California. AAAI Press.

[Bayardo et al., 1998] Bayardo, R. J., Bohrer, W., Brice, R., Cichocki, A., Fowler, J., Helal, A., Kashyap, V., Ksiezyk, T., Martin, G., Nodine, M., Rashid, M., Rusinkiewicz, M., Shea, R., Unnikrishnan, C., Unruh, A., and Woelk, D. (1998). Infosleuth: Agent-based semantic integration of information in open and dynamic environments. In Huhns, M. and Singh, M., editors, *Readings in Agents*, pages 205–216. Morgan Kaufmann.

[Bendou, 1999] Bendou, M. (1999). Approche sma pour un forum dynamique. Master's thesis, Université Savoie, Ecole des Mines St Etienne.

[Berthet et al., 1992] Berthet, S., Demazeau, Y., and Boissier, O. (1992). Knowing each other better. In *Proceedings of the 11th International Workshop on DAI*, pages 23–42, Glen Arbor.

[Boissier, 2000] Boissier, O. (2000). *A para tre*, chapter 4. Modèles et architectures d'agents.

[Boissier et al., 1998] Boissier, O., Beaune, P., Sayettat, C., Carron, T., Hannoun, M., Proton, H., and Vercouter, L. (1998). Multi-agent system toolkit. Technical report, Equipe SMA, Centre SIMMO, ENSM.SE.

[Bothorel, 1998] Bothorel, C. (1998). Des communautés dynamiques d'agents pour des services de recommandation. In *Journées Francophones d'Intelligence Artificielle Distribuée et de Systèmes Multi-Agents*, Pont-à-Mousson. Hermès.

[Bradshaw, 1997] Bradshaw, J. M. (1997). Kaos: Toward and industrial-strength open agent architecture. In Bradshaw, J. M., editor, *Software Agents*, chapter 17, pages 375–418. AAAI Press/The MIT Press.

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

[Carmel and Markovitch, 1998] Carmel, D. and Markovitch, S. (1998). How to explore your opponent's strategy (almost) optimally. In *Proceedings of the Third International Conference on Multi-Agent Systems*, pages 64–71, Paris, France. IEEE Computer Society.

[Carron et al., 1999] Carron, T., Proton, H., and Boissier, O. (1999). A temporal agent communication language for dynamic multi-agent systems. In *Modelling Autonomous Agents in a Multi-Agent World (MAAMAW'99)*, pages 115–127, Valenica, Espagne.

[Castelfranchi et al., 1992] Castelfranchi, C., Miceli, M., and Cesta, A. (1992). Dependence relations among autonomous agents. In Werner, E. and Demazeau, Y., editors, *Decentralized A. I. 3*, pages 215–227. Elsevier Science Publishers.

[Cavedon and Sonenberg, 1998] Cavedon, L. and Sonenberg, L. (1998). On social commitment, roles and preferred goals. In Demazeau, Y., editor, *Proceedings of the third International Conference on Multi-Agent Systems*, pages 80–87, Paris, France. IEEE Computer Society.

[Chang and Woo, 1991] Chang, M. and Woo, C. (1991). Sanp: a communication level protocol for negociations. In Werner, E. and Demazeau, Y., editors, *Decentralized A.I. 3*, pages 31–54, Kaiserslautern.

[Cohen et al., 1998] Cohen, P. R., Cheyer, A., Wang, M., and Baeg, S. C. (1998). An open agent architecture. In Huhns, M. N. and Singh, M. P., editors, *Readings in Agents*, chapter 3.1, pages 197–204. Morgan Kaufmann.

[Collinot et al., 1996] Collinot, A., Ploix, L., and Drogoul, A. (1996). Application de la méthode cassiopée à l'organisation d'une équipe de robots. In *Intelligence Artificielle Distribuée et Systèmes Multi-Agents*, Port Camargue. Hermès.

[Decker, 1996] Decker, K. S. (1996). Task environment centered simulation. In Prietula, M. and Carley, K., editors, *Simulating Organizations: Computational Models of Institutions and Groups*. AAAI Press / MIT Press.

[Decker and Lesser, 1992] Decker, K. S. and Lesser, V. R. (1992). Generalizing the partial global planning algorithm. *International Journal of Intelligent Cooperative Information Systems*, pages 319–346.

[Delgado and Ishii, 1999] Delgado, J. and Ishii, N. (1999). Formal models for learning of user preferences, a preliminary report. In *Learning About Users, IJCAI-99 workshop*, Stockholm, Sweden.

[Demazeau, 1997] Demazeau, Y. (1997). Steps towards multi-agent oriented programming. In *1st International Workshop on Multi Agent Systems, IWMAS'97*, Boston.

[Descartes, 1647] Descartes, R. (1647). *Méditations Métaphysiques - Objection et Réponses*. Garnier-Flammarion.

[Doan and Beigbeder, 1999] Doan, B.-L. and Beigbeder, M. (1999). Virtual www documents: a concept to explicit the structure of www sites. In *21st Annual Colloquium on IR Research*, Glasgow, Ecosse.

[Durfee, 1995] Durfee, E. H. (1995). Blissful ignorance: Knowing just enough to coordinate well. In *Proceedings of the First International Conference on Multi-Agent Systems*, pages 406–413, San Francisco, California. AAAI Press/The MIT Press.

[Durfee et al., 1997] Durfee, E. H., Kiskis, D. L., and Birmingham, W. P. (1997). The agent architecture of the university of michigan digital library. In Huhns, M. H. and Singh, M. P., editors, *Readings in Agents*, pages 98–108. Morgan Kaufmann, San Francisco, California.

[FIPA, 1997] FIPA (1997). *Agent Communication Language*. Foundation for Intelligent Physical Agents, Geneva, Switzerland.

[FIPA, 2000a] FIPA (2000a). Agent management specification. Technical Report XC00023G, Foundation for Intelligent Physical Agents, Geneva, Switzerland.

[FIPA, 2000b] FIPA (2000b). Agent software integration specification. Technical Report XC00079A, Foundation for Intelligent Physical Agents, Geneva, Switzerland.

[Fisher, 1987] Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139–172.

[Foss, 1998] Foss, J. D. (1998). Brokering the info-underworld. In Jennings, N. R. and Wooldridge, M., editors, *Agent Technology - foundations, applications and markets*, chapter 6, pages 105–124. Springer.

[Franc and Sanders, 1998] Franc, A. and Sanders, L. (1998). Modèles et systèmes multi-agents en écologie : état de l'art et comparaison avec les approches classiques. In *Modèles et systèmes multi-agents pour la gestion de l'environnement et des territoires*, pages 17–34, Clermont-Ferrand, France. Cemagref.

[Gasser, 1991] Gasser, L. (1991). Social conceptions of knowledge and action: Dai foundations and open systems semantics. In Huhns, M. N. and Singh, M. P., editors, *Readings in Agents*, chapter 4.3, pages 389–404. Morgan Kaufmann.

[Gasser et al., 1987] Gasser, L., Braganza, C., and Herman, N. (1987). Implementing distributed ai systems using mace. In *Proceedings of the 3rd IEEE Conference on Artificial Intelligence Applications*, pages 315–320, Orlando.

[Giroux, 1995] Giroux, S. (1995). Open reflective agents. In Wooldridge, M., Muller, J. P., and Tambe, M., editors, *Intelligent Agents 2 (ATAL'95)*, pages 315–330.

[Gleizes and Glize, 2000] Gleizes, M.-P. and Glize, P. (2000). Abrose : Des systèmes multi-agents pour le courtage adaptatif. In *Journées Francophones d'Intelligence Artificielle Distribuée et de Systèmes Multi-Agents*, pages 117–132, Saint-Jean-La-Vêtre, Loire. Hermès.

[Gmytrasiewicz, 1995] Gmytrasiewicz, P. J. (1995). On reasoning about other agents. In Wooldridge, M., Müller, J. P., and Tambe, M., editors, *Intelligent Agents 2, Agent Theories, Architectures and Languages*, pages 143–155, Montréal, Canada. Springer.

[Gmytrasiewicz and Durfee, 1995] Gmytrasiewicz, P. J. and Durfee, E. H. (1995). A rigorous, operational formalization of recursive modeling. In *Proceedings of the first International Conference on Multi-Agent Systems*, pages 125–132, San Francisco, California. AAAI Press/MIT Press.

[Gnutella, 2000] Gnutella (2000). http://gnutella.wego.com/.

[Gutknecht and Ferber, 1999] Gutknecht, O. and Ferber, J. (1999). Vers une méthodologie organisationnelle de conception de systèmes multi-agents. In *Ingénierie des systèmes multi-agents (JFIADSMA'99)*, pages 93–104, Saint-Gilles, Ile de La Réunion. Hermès.

[Gutknecht et al., 2000] Gutknecht, O., Ferber, J., and Michel, F. (2000). Madkit : Une expérience d'architecture de plate-forme multi-agent générique. In Pesty, S. and Sayettat-Fau, C., editors, *Journées Francophones pour les Systèmes Multi-Agents et l'Intelligence Artificielle Distribuée (JFIADSMA'00)*, pages 223–236, Saint-Jean-La-Vêtre. Hermès.

[Haddadi and Sundermeyer, 1993] Haddadi, A. and Sundermeyer, K. (1993). Knowledge about other agents in heterogeneous dynamic domains. In *Proceedings of the International Conference on Intelligent and Cooperative Information Systems*, number ISBN 0-8186-3135-X, pages 64–70, Rotterdam, Netherlands. IEEE-CS Press.

[Hannoun et al., 1999] Hannoun, M., Boissier, O., Sichman, J. S., and Sayettat, C. (1999). Moise : un modèle organisationnel pour la conception de systèmes multi-agents. In Gleizes, M.-P. and Marcenac, P., editors, *Ingénierie des systèmes multi-agents (JFIADSMA99)*, pages 105–118, Saint-Gilles, Ile de la Réunion. Hermès.

[Hannoun et al., 1998] Hannoun, M., Sichman, J. S., Boissier, O., and Sayettat, C. (1998). Dependence relations between roles in a multi-agent system. In *Multi-agent systems and Agent-Based Simulation*, Paris.

[Hewitt, 1986] Hewitt, C. (1986). Offices are open systems. *Communications of the ACM*, 4 (3):271–287.

[Hewitt and Inman, 1991] Hewitt, C. and Inman, J. (1991). Dai betwixt and between: from "intelligent agents" to open systems science. In Huhns, M. N. and Singh, M. P., editors, *Readings in Agents*, chapter 4.3, pages 405–415. Morgan Kaufmann.

[Hewitt and Jong, 1982] Hewitt, C. and Jong, P. D. (1982). Open systems. Technical Report AI Memo 691, MIT Artificial Intelligence Laboratory.

[Hilaire et al., 2000] Hilaire, V., Koukam, A., Grüer, P., and Müller, J.-P. (2000). Vers une méthodologie formelle de spécification des systèmes multi-agents. In Pesty, S. and Sayettat, C., editors, *Actes des 8èmes Journées Francophones d'Intelligence Artificielle Distribuée et Systèmes Multi-Agents*, pages 209–221, Saint-Jean-La -Vêtre, Loire, France. Hermès.

[Huhns and Singh, 1998] Huhns, M. and Singh, M. (1998). Multi-agent systems in information rich environment. In *Cooperative Information Agents II*, pages 79–93, La Villette, Paris, France. Springer.

[Kaminka et al., 1998] Kaminka, G. A., Tambe, M., and Hopper, C. M. (1998). The role of agent-modeling in agent robustness. In *AI Meets Real World (AIMTRW-98)*, Stamford, CT.

[Kayser, 1997] Kayser, D. (1997). *La représentation des connaissances.* Hermès.

[Labrou and Finin, 1994] Labrou, Y. and Finin, T. (1994). A semantics approach for kqml - a general purpose communication language for software agents. In *Third International Conference on Information and Knowledge Management.*

[Langley, 1996] Langley, P. (1996). *Elements of Machine Learning.* Morgan Kaufmann.

[Lashkari et al., 1998] Lashkari, Y., Metral, M., and Maes, P. (1998). Collaborative interface agents. In Huhns, M. N. and Singh, M. P., editors, *Readings in Agent*, pages 111–116. Morgan Kaufmann.

[Lebowitz, 1986] Lebowitz, M. (1986). Concept learning in a rich input domain: Generalization-based memory. In Michalski, R. S., Carbonell, J. G., and Mitchell, T. M., editors, *Machine Learning - an artificial intelligence approach*, volume 2, chapter 8, pages 193–214. Morgan Kaufmann.

[Lieberman, 1995] Lieberman, H. (1995). Letizia: An agent that assists web browsing. In *International Joint Conference on Artificial Intelligence*, Montreal.

[Müller, 1998] Müller, J.-P. (1998). Vers une méthodologie de conception de systèmes multi-agents de résolution de problèmes par émergence. In Barthès, J.-P., Chevrier, V., and Brassac, C., editors, *Journées Francophones d'Intelligence Artificielle Distribuée et de Systèmes Multi-Agents*, pages 355–371. Hermès.

[Quinlan, 1986] Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1:81–106.

[Ramat et al., 1998] Ramat, E., Preux, P., Seuront, L., and Lagadec, Y. (1998). Modélisation multi-agents de systèmes naturels - réflexion générale et application en biologie marine. In *Modèles et systèmes multi-agents pour la gestion de l'environnement et des territoires*, pages 35–50, Clermont-Ferrand, France. Cemagref.

[Rao and Georgeff, 1991] Rao, A. S. and Georgeff, M. P. (1991). Modeling rational agents within a bdi architecture. In Allen, J., Fikes, R., and Sandwall, E., editors, *Proceedings of the Int. Conf. on Principles of Knowledge Representation and Reasoning, KR-91*, pages 473–484, San Mateao. Morgan Kaufmann.

[Resnick et al., 1994] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. (1994). Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of ACM 1994 Conference on CSCW*, pages 175–186, Chapel Hill.

[Rich, 1979] Rich, E. (1979). User modeling via stereotypes. In *Cognitive Science*, volume 3, pages 335–366.

[Rognon, 2000] Rognon, J.-L. (2000). Meta-annuaire. *Le Monde Informatique*, (no 862):pp. 4–6.

[Searle, 1969] Searle, J. R. (1969). Speech act: An essay in the philosophy of language. Cambridge University Press.

[Shoham, 1990] Shoham, Y. (1990). Agent oriented programming. Technical report, Stanford University.

[Sian, 1990] Sian, S. S. (1990). Adaptation based on cooperative learning in multi-agent systems. In Demazeau, Y. and Müller, J.-P., editors, *Decentralized A.I. 2*, pages 257–272, St Quentin en Yvelines.

[Sichman, 1995] Sichman, J. S. (1995). *Du Raisonnement Social chez les Agents*. PhD thesis, Institut National Polytechnique de Grenoble.

[Stone et al., 2000] Stone, P., Riley, P., and Veloso, M. (2000). Defining and using ideal teammate and opponent agent models. In Press, A. P., editor, *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, pages 1040–1045, Austin, Texas.

[Tambe, 1997] Tambe, M. (1997). Towards flexible teamwork. In *Journal of Artificial Intelligence Research*, volume 7, pages 83–124. Morgan Kaufmann.

[Tidhar et al., 1998] Tidhar, G., Sonenberg, E. A., and Rao, A. S. (1998). On team knowledge and common knowledge. In Demazeau, Y., editor, *Proceedings of the third International Conference on Multi-Agent Systems*, pages 301–308, Paris, France. IEEE Computer Society.

[Veloso and Stone, 1998] Veloso, M. and Stone, P. (1998). Individual and collaborative behaviors in a team of homogeneous robotic soccer agents. In Demazeau, Y., editor, *Proceedings of the third International Conference on Multi-Agent Systems*, pages 309–316, Paris, France. IEEE Computer Society.

[Vercouter, 1997] Vercouter, L. (1997). Coordination multi-agents pour la résolution de problèmes de gestion locale sous contrainte collective. Master's thesis, Université Paris 9.

[Vercouter et al., 1998] Vercouter, L., Beaune, P., and Sayettat, C. (1998). Apprentissages dans les sma. In *Journées Francophones des Systèmes Multi-Agents et de l'Intelligence Artificielle Distribuée (JFIADSMA'98)*, Pont-à-Mousson, France. Hermès.

[Vercouter et al., 2000] Vercouter, L., Beaune, P., and Sayettat, C. (2000). Towards open distributed information systems by the way of a multi-agent conception framework. In Lespérance, Y., Wagner, G., and Yu, E., editors, *Agent-Oriented Information Systems, Seventeenth National Conference on Artificial Intelligence*, Austin, Texas.

[Vidal and Durfee, 1995] Vidal, J. and Durfee, E. H. (1995). Recursive agent modeling using limited rationality. In *Proceedings of the first International Conference on Multi-Agent Systems*, pages 376–383, San Francisco, California. AAAI Press/MIT Press.

[Wagner, 2000a] Wagner, G. (2000a). The agent-object-relationship metamodel: Towards a unified conceptual view of state and dynamics. Technical report, Institut für Informatik, Freie Universität Berlin.

[Wagner, 2000b] Wagner, G. (2000b). Towards agent-oriented information systems. Technical report, Institut für Informatik, Freie Universität Berlin.

[Werner, 1996] Werner, E. (1996). Logical foundations of distributed artificial intelligence. In O'Hare, G. M. P. and Jennings, N. R., editors, *Foundations of DAI*, chapter 2, pages 57–118. Wiley-Interscience.

[Wilkins and Myers, 1995] Wilkins, D. E. and Myers, K. L. (1995). A common knowledge representation for plan generation and reactive execution. *Journal of Logic and Computation*, 5:731–761.

[Wooldridge and Jennings, 1994] Wooldridge, M. and Jennings, N. R. (1994). Towards a theory of cooperative problem solving. In Demazeau, Y., Muller, J.-P., and Perram, J., editors, *Proceedings of the 6th European Workshop on Modeling Autonomous Agents in a Multi-Agent World*, pages 15–26, Odense, Danemark.

[Wooldridge et al., 1999] Wooldridge, M., Jennings, N. R., and Kinny, D. (1999). A methodology for agent-oriented analysis and design. In Etzioni, O., Muller, J. P., and Bradshaw, J., editors, *Agents'99: Proceedings of the third International Conference on Autonomous Agents*, Seattle.