Proposition d'un modèle organisationnel générique de systèmes multi-agents et examen de ses conséquences formelles, implémentatoires et méthologiques

Olivier Gutknecht

ACADÉMIE DE MONTPELLIER

UNIVERSITÉ MONTPELLIER II

— **S**CIENCES ET **T**ECHNIQUES DU **L**ANGUEDOC —

THÈSE

présentée à l'Université des Sciences et Techniques du Languedoc
pour obtenir le diplôme de DOCTORAT

SPÉCIALITÉ : **INFORMATIQUE**
Formation Doctorale : **Informatique**
École Doctorale : **Information Structures Systèmes**

Proposition d'un modèle organisationnel générique de systèmes multi-agents

Examen de ses conséquences formelles, implémentatoires et méthodologiques

par

Olivier GUTKNECHT

Soutenue le 14 septembre 2001 devant le Jury composé de :

M. Jean-Pierre BRIOT, Directeur de Recherche, Université Paris 6, ... Rapporteur
M. Les GASSER, Professeur, University of Illinois, ... Rapporteur
M. Pierre MARCENAC, Professeur, Université de la Réunion, ... Rapporteur
Mme. Thérèse LIBOUREL, Maître de Conférence, CNAM, ... Examinateur
M. Jean-Pierre MÜLLER, Cadre de Recherche, CIRAD, ... Examinateur
M. Joël QUINQUETON, Professeur, Université Montpellier III, ... Président
M. Jacques FERBER, Professeur, Université Montpellier II, ... Directeur de Thèse

Un lecteur de rêves khazar, encore élève dans un monastère, reçut en cadeau un vase qu'il rangea dans sa cellule. Le soir il y déposa sa bague. Mais lorsqu'il voulut la reprendre le lendemain matin, elle n'y était plus. Vainement il enfonçait son bras dans le vase, il n'arrivait pas à en toucher le fond. Cela le surprit car le récipient semblait moins haut que son bras n'était long. Il le souleva mais, dessous, le sol était plat, et il n'y avait aucune ouverture dans le vase, pas plus que dans n'importe quel autre. Il prit un bâton et essaya d'atteindre le fond, mais toujours sans succès ; le fond du vase semblait se dérober. Il se dit "Là où je suis, là est ma limite" et il s'adressa à son maître Mokadasa Al Safer , lui demandant d'expliquer ce que signifiait un tel phénomène. Le maître prit un caillou, le jeta dans le vase, et compta. Lorsqu'il arriva à 70 on entendit à l'intérieur du récipient un bruit de plongeon, comme si un objet était tombé dans l'eau et le maître dit :

— Je pourrais t'expliquer ce que représente ton vase, mais demande-toi d'abord si c'est bien utile. Dès que je t'aurai dit ce qu'il en est, le vase prendra, pour toi et les autres, une valeur inférieure à celle qu'il a maintenant. En effet, quelle que soit sa valeur, elle ne peut être supérieure à celle du tout. Et dès que je t'aurai dit ce qu'il est, il ne sera plus tout ce qu'il n'est pas, et donc plus ce qu'il est encore maintenant.

L'élève fut d'accord avec son maître. Mais ce dernier prit un bâton et cassa le vase. Stupéfait, le jeune homme lui demanda le motif de ce dommage et le maître répliqua :

— Le dommage aurait consisté à te dire à quoi servait ce vase avant de le casser. Mais puisque tu ne connais pas son usage, le dommage n'existe pas, car le vase te servira toujours, comme s'il n'était pas cassé...

En effet, le vase khazar sert encore, bien qu'il n'existe plus depuis longtemps.

— Milorad Pavić, *Le Dictionnaire Khazar*

Remerciements

Arrive in Neurotica - Through Neon Heat Disease
— King Crimson

Merci à Les Gasser de m'avoir fait l'honneur de rapporter sur ce travail. J'ai été plus d'une fois admiratif devant la portée de ses travaux fondateurs sur les organisations multi-agents. J'espère qu'il n'en trouvera pas les reflets développés ici trop pâles.

Merci à Jean-Pierre Briot d'avoir accepté de rapporter sur cette thèse[1], avec sa rigueur, sa gentillesse et sa finesse d'analyse. J'avais parfois à l'esprit la belle simplicité de certains de ses modèles objets quand j'essayais péniblement d'élaguer AGR.

Merci à Pierre Marcenac d'avoir accepté de rapporter sur ce travail, malgré la distance. J'ai encore souvenir de discussions à propos de modèles organisationnels, un peu irréelles sous un soleil tropical.

Merci à Jean-Pierre Müller de m'avoir fait le plaisir de participer à ce jury. J'ai l'impression d'avoir joué une longue, intermittente et passionnante partie de ping-pong conceptuel pendant ces quelques années.

Un grand merci à un autre membre de ce jury, Thérèse Libourel, dont la rigueur scientifique n'a d'égale que la générosité, et que j'ai parfois terrifiée par certaines audaces diagrammatiques un peu, ahem, contestables.

Merci à Joel Quinqueton pour nos échanges scientifico-agentisto-informatiques, son aide, et son enthousiasme tout au long de ma présence au LIRMM, jusqu'à finalement ce jury de thèse.

Merci pour tout, Jacques. Quand il y a une dizaine d'années je fouillais dans une pile déjà un peu poussiéreuse de magazines pour lire quelques curieuses chroniques qui parlaient d'I.A., d'objet et autres choses amusantes, j'étais loin de penser que j'aurais le plaisir de travailler avec l'auteur. Certaines discussions pendant lesquelles ces modèles se sont construits ont été pour moi des moments de pure jubilation intellectuelle.

Merci à Alexis Drogoul et toute la bande du LIP6 pour avoir accueilli aussi gentiment un 'squatteur officiel' pendant de longs mois de rédaction exilée. La fin de ce travail vous doit beaucoup.

Un grand, grand merci à Jean César. Sans une tortue pour dessiner des docacres au soleil, strictement rien de tout cela ne serait arrivé.

Merci au passage à quelques maîtres à penser : Seymour Papert, qui m'a indirectement fait découvrir la beauté des abstractions informatiques et donné cette passion, Patrick Henry Winston dont les livres m'ont fait découvrir Lisp et I.A., Mitchell Resnick qui a déclenché cette fascination pour l'émergence et les systèmes complexes, Pattie Maes dont certains papiers m'ont fait basculer dans les agents.

[1] Et de m'avoir fait le plus beau compliment qu'on pouvait sans doute me faire à propos de ce travail

Merci à Nils Ferrand, Remy Courdier, Vincent Chevrier, Robert Lifran, Stéphane Garin et bien d'autres, pour toutes nos discussions sur AGR ou MadKit.

Merci à Francois de Coligny, Emmanuel Lieurain, Marc Dupont, Guillaume Vidon, Louis Peyras, Pierre Bommel, Fabien Michel, Thomas Cahuzac, qui ont sans doute dû maudire plusieurs fois l'auteur d'une certaine plate-forme mais l'ont rendu assurément moins bancale. J'en profite également pour remercier tous les utilisateurs de ladite plate-forme pour leurs remarques et leur aide. Le sentiment d'avoir pu faire un outil utile à d'autres recherches est toujours grisant.

Merci aux habitants du hall. Youssef[2], David, Isa, Mathieu, Jean-David, Fred, Olivier, Eugène, Manu, Vincent, avec qui les échanges furent toujours fructeux, même entre la Bretagne et le Languedoc, et Luc, quoi (qui avec Sophie, m'ont fait me sentir bien à Montpellier, au tout début). Laurent & Laurent, compagnons de mauvais esprit[3].

Merci à Jean-Marie Hullot et Bertrand Guiheneuf pour m'avoir donné l'occasion de m'aérer la tête à la toute fin de ce travail, et d'y penser un peu moins pour y penser un peu mieux.

Merci à la fine équipe des Graves, pour votre amitié et votre soutien, fut-il à des milliers de kilomètres. Vous avez vu venir ce travail de très loin.

Merci Odile. Tu fais partie des personnes qui savent pourquoi tout ceci s'est fait.

Merci à Luc S. Heinrich qui, en plus d'être un ami et le dandy-geek par excellence, m'a fait cristalliser au moment idoine ce que pourrait être une intéressante programmation par agents, avec un de nos projets jamais commencé. Nasty ? Yeah.

Merci aux petits mondes de `fcsm`, des bruiteurs, trolleurs et autres testeurs[4]. J'informe.

Merci à Murphy, Twin-Peaks, Broadway et Discipline qui ont tout supporté sans trop rechigner.

Merci le Diag', dealer indispensable en images mouvantes. Grand merci et force bises à Juliette, alter ego cinéphilique, pourvoyeuse officielle en bons plans, idées folles, et échanges scientifiques improbables [RMP, 1999].

Merci à toi, Lou. Les thèses parallèles ont été quelque chose de marquant et tu sais ce que ce travail te doit, mais c'est d'autres moments dont je me souviendrai bien plus.

Merci enfin à ma famille, mes parents et ma frangine, pour leur soutien sans faille, leur patience face aux grommellements et doutes de fin de rédaction.

Merci Véro, pour un presque-Orsay et tout ce qui s'ensuit. Et merci de m'avoir aidé à remettre la tête dans tout cela.

Merci enfin à L. - forcément

[2] $DCSP \subset SMA$, j'en ai la preuve, mais la marge n'est pas assez grande

[3] Même si on ne sait même pas faire, en fait

[4] Versum

In girum imus nocte et consumimur igni – A mes grands-parents

Table des matières

Première partie

Positionnement

Chapitre 1

Introduction

1.1 Sur ce travail

Comment comprendre les systèmes informatiques ? Nous cherchons sans cesse à bâtir de nouvelles manières de concevoir, construire et analyser ces systèmes. La thématique du discours que nous allons entreprendre gravite autour de cette idée : comment réduire le fossé entre une idée, un besoin et l'élaboration du système qui l'incarnera et quels cadres adopter pour servir de base à l'intuition puis à l'analyse.

Montée vers l'abstraction, objet, modularisation, apprentissage, adaptation, interactions, toutes ces approches ne sont finalement que l'expression d'une même volonté : se placer à un niveau d'expression adéquat et trouver le meilleur rapport entre liberté laissée au système et maintien d'un contrôle sur celui-ci. Les agents sont l'une de ces expressions possibles : un parti pris d'autonomie, d'interactions souples et d'(auto-)organisation pour modéliser et concevoir des systèmes toujours plus adaptables, modulaires et décentralisés. Ce sera le cadre de notre travail.

Le thème général que le lecteur entendra en parcourant cet ouvrage est celui de la représentation de l'organisation dans les systèmes multi-agents. Nous montrerons que garder un point de vue essentiellement social peut apporter structuration et ouverture dans ces systèmes à agents qui pourraient à terme devenir un paradigme de programmation comme un autre. La finalité de ces recherches est de proposer des pistes pour aider à comprendre et concevoir des systèmes multi-agents de taille importante, ouverts et hétérogènes. Le lecteur y croisera plusieurs modèles, implémentations et méthodes qui auront tous la caractéristique d'adopter comme point de départ une vue organisationnelle de leur champ d'action.

Nous voulons également par avance prévenir que l'organisation sera vue ici comme un guide générique, un cadre de représentation et d'analyse. Notre propos ne sera donc pas structuré par une définition spécifique - sociologique, éthologique, purement informatique ou autre - mais par le souci d'identifier et d'y rapporter un ensemble minimal de concepts

génériques et d'analyser leur portée. Notre hypothèse essentielle sera alors de voir l'organisation multi-agents comme notion primitive, au même titre que les entités qui la construisent et sont façonnées par elle.

1.2 Sur ce document

Ce travail se structure en deux parties. On trouve en premier lieu la définition d'un modèle organisationnel générique pour l'expression de systèmes multi-agents. Cela s'accompagne ensuite d'une série de réflexions sur ce modèle, selon divers points de vue : formel, implémentatoire, méthodologique ou applicatif.

Certains de ces points de vue se complètent, d'autres sont plus parallèles, mais tous seront des facettes de ce modèle d'organisation. La figure 1.1 illustre cet agencement.

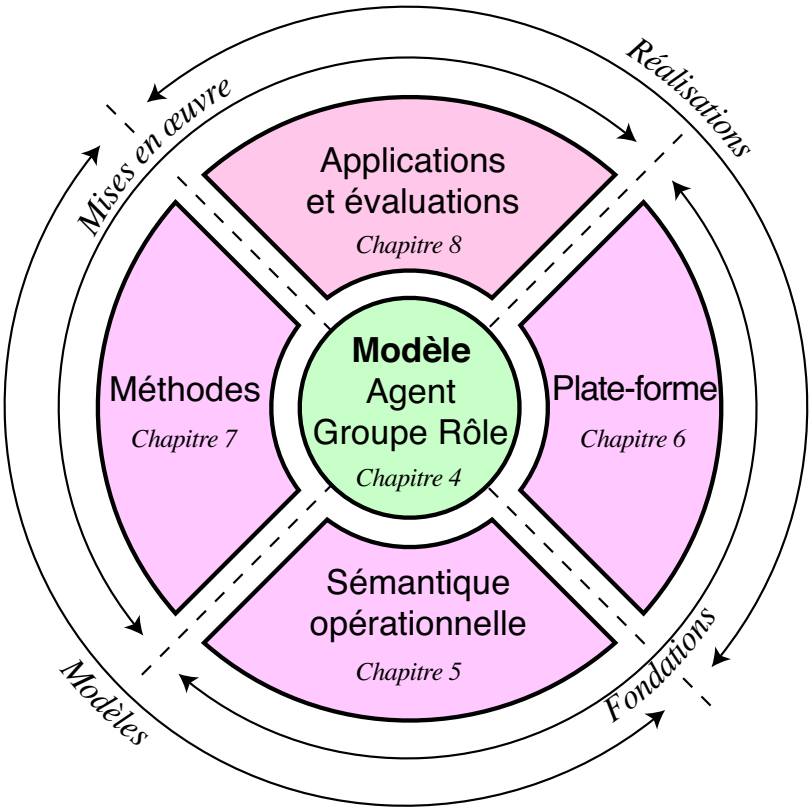


FIG. 1.1 – Structure de la présente contribution

Ceci fait qu'une fois après avoir entendu le thème principal le lecteur saura que les différentes parties qui suivront ne seront que variations sur un même thème qu'il pourrait en

fait aborder à sa convenance. On pourrait donc imaginer sur ce principe plusieurs grilles de lecture :

Un parcours thématique, en abordant ces variations individuellement, ou par certaines combinaisons, comme :

L'aspect applicatif, par la présentation de la plate-forme d'implémentation et des expérimentations qui y ont été réalisées.

Les modèles proposés, en voyant comment la sémantique formelle élaborée peut exprimer les opérations primitives du modèle initial, et comment le modèle peut être abstrait pour exprimer des notions de conceptions.

Un parcours structurel, en ayant une lecture plus verticale, où l'on pourrait se focaliser sur :

Les fondations, où le souci d'opérationalisation se retrouve entre bases formelles et noyau implémentatoire.

Les réalisations, où l'on pourrait se contenter de n'examiner le modèle proposé que comme support de conception et de construction d'application, par les contributions en termes de plate-forme ou de méthodologie.

Notons pour finir que certains thèmes reviendront tout au long des chapitres, en particulier la mise en place du méta-niveau, et seront systématiquement réexaminés à la lumière de chaque variation.

Cette lecture éclatée n'est évidemment pas une obligation : reprenons une vue plus classique et linéaire, et revenons sur l'organisation de ce document.

La première partie pose le cadre de notre recherche. Après la présente introduction, le second chapitre situera le contexte de cette thèse : les systèmes multi-agents. Nous montrerons d'abord pourquoi les architectures logicielles actuelles exhibent des propriétés et des contraintes qui justifient des approches à base d'agents. Nous verrons néanmoins que les systèmes multi-agents ne sont pas une réponse absolue et qu'en particulier des systèmes réellement ouverts et hétérogènes restent problématiques.

La seconde partie forme la partie centrale de la proposition : des modèles sociaux pour les systèmes multi-agents. Nous verrons en quoi il semble possible de résoudre les points délicats que nous avons identifiés par l'utilisation de mécanismes organisationnels.

Le troisième chapitre évoquera ensuite différents modèles sociaux informatiques préexistants, que ce soit dans le domaine multi-agents ou à sa lisière, et nous verrons pourquoi ce champ reste encore disparate et spécialisé. Nous identifierons la plupart des concepts et métaphores sociales manipulés par la suite et croiserons plusieurs cas justifiant l'intérêt de ce type de modèles.

Le quatrième chapitre est le coeur de notre contribution. Un modèle d'organisation générique basé sur trois notions clés d'agent, de groupe et de rôle y est présenté. Nous l'illustrons par quelques exemples, et défendrons son orginalité et ses points forts. Cette partie se conclut sur quelques remarques sur la dynamique et les aspects réflexifs de ce modèle.

Ceci fait, nous entrerons dans la troisième partie de ce document : le retour sur le modèle proposé selon différentes approches.

Le premier angle de vue est celui des bases formelles. Le chapitre cinq propose l'expression d'une sémantique opérationnelle de celui-ci exprimée en π-calcul. Un modèle d'agent possible y est exposé. Nous y montrons comment la dynamique de l'organisation peut se définir au méta-niveau.

Le sixième chapitre reprend l'idée de fondation pour notre modèle, mais cette fois-ci selon une perspective d'implémentation. Une plate-forme d'exécution et de développement de systèmes multi-agents appelée MADKIT y est décrite en détail. Nous y montrons l'intérêt d'une vue organisationnelle au plus profond niveau des systèmes et les conséquences sur l'architecture et les utilisations.

Ayant traité de modèle et d'implémentation, il parait judicieux de se poser la question du processus de conception. Le chapitre sept répond à ce souci en décrivant différents modèles méthodologiques et notations, et propose quelques solutions pour faire le lien avec des méthodologies existantes.

Enfin, il reste à savoir ce que l'on peut faire à partir du modèle et de la plate-forme proposée. Le huitième chapitre en fait donc l'illustration par quelques systèmes et applications parmi d'autres. Cela nous donne alors l'occasion d'évaluer la volonté de généricité avancée lors de la présentation initiale du modèle et de la plate-forme d'implémentation.

Il ne nous restera donc plus qu'à revenir sur ce travail, pour en questionner le champ d'action et envisager quelques pistes pour le chemin restant (forcément) à parcourir.

Chapitre 2

Problématique des systèmes informatiques complexes

Les travaux de recherche présentés dans cette étude reposent sur le questionnement suivant : les systèmes multi-agents pourraient-ils devenir à terme un *modèle de programmation* réaliste de systèmes informatiques ?

Faire cette hypothèse entraîne forcément d'en assumer les conséquences. Soutenir que le modèle agent puisse être rangé à côté des modèles procéduraux, objets, ou componentiels implique d'être capable de se confronter aux mêmes styles de problèmes que ceux abordés par les procédés existants, ce qui est déjà une ambition intéressante. Mais il faut encore, pour en justifier l'intérêt, pouvoir montrer que l'on peut se confronter avec succès à des écueils *actuels* que les approches classiques savent peu ou mal traiter.

Formuler des propositions dans ce domaine nécessite entre autres de se positionner par rapport au génie logiciel, pour pouvoir en préciser les préoccupations, et montrer en quoi les approches à base d'agents peuvent répondre à certains de ces besoins. Il nous faudra malgré tout préciser pourquoi ces modèles ne sont pas encore utilisables dans un cadre général de façon réaliste, et quelle gamme de solutions pourrait être envisagée.

C'est ce à quoi nous allons nous employer dans ce chapitre.

2.1 Evolution des architectures logicielles

2.1.1 Changements de perspectives

Il y a à peine plus d'une quinzaine d'années, l'informatique était pour beaucoup un environnement de systèmes informatiques très homogènes, monolithiques et centralisés. A cela s'ajoutait une myriade de machines peu puissantes, déconnectées, dont les interactions se faisaient beaucoup plus par le réseau social de leurs utilisateurs [Breton, 1990] que par quelque architecture informatique.

Bien que des solutions d'interconnexion et d'interopérabilité existent depuis longtemps [Cerf et Kahn, 1974], la mise en place et l'exploitation de la distribution naturelle des systèmes n'a jamais été immédiatement pertinente. Néanmoins, progressivement le consensus s'est formé autour de quelques solutions techniques d'infrastructure ou d'implémentation (comme les réseaux IP pour les communications, des représentations de données comme l'ASCII, l'Unicode ou XML, des protocoles de transports comme HTTP, ou des paradigmes de programmation comme l'objet). Même si on pouvait contester la valeur individuelle de chacun de ces choix, on ne peut en réfuter la valeur en tant que fondation admise et partagée.

L'avènement de ces nouveaux *standards de fait*, ainsi que l'explosion de nouveaux usages [Berners-Lee et Fischetti, 1999] ont été les précurseurs - ou plutôt les déclencheurs - de nouveaux besoins, puis d'architectures plus adaptées à cette évolution des systèmes informatiques. L'acceptation de l'hétérogénéité, les archipels de données, les liaisons souples entre systèmes sont maintenant plus souvent la règle que l'exception. Cela ne va pas sans mal, car à la simple difficulté "technique" de mise en oeuvre de tels systèmes s'ajoute une difficulté conceptuelle, pour les envisager, représenter, ou valider.

Un des objectifs actuels important est donc déjà d'identifier et comprendre ces *besoins*, d'y répondre par les *infrastructures* ou *applications*, et d'élaborer *méthodes* et *outils* pour les bâtir, évaluer et faire évoluer.

2.1.2 Nouvelles infrastructures

On a pu observer assez récemment, dans des contextes de recherche ou industriels, l'importance portée au travail sur les infrastructures logicielles[1]. Cet effort de construction d'une base d'intégration est intéressante du point de vue architectural par le changement de perspective qu'elle implique. L'objectif n'est plus de bâtir une application unique, ne reposant que sur les services d'un système d'exploitation ou de bibliothèques isolées, mais de minimiser l'effort en s'appuyant sur des frameworks [Jacobson et Nowack, 1999] spécialisés.

Le travail sur les infrastructures devient donc une composante à part entière des systèmes. Pour preuve, on mentionnera l'accent mis sur des fondations d'interconnexion telles que COM [Gray et al., 1998] ou CORBA [OMG, 1992], qui offrent un modèle objet indépendant des implémentations mais néanmoins opératoire, ou encore les serveurs d'applications [Monson-Haefel, 1999] [Platt, 2001].

Mais par rapport à une approche plus "classique" du logiciel, on se rend compte aussi que ces nouvelles possibilités mettent en lumière des difficultés conceptuelles nouvelles. On peut par exemple évoquer :

– L'inquiétude face à des systèmes d'informations moins centralisés. Vouloir construire des applications fédérant des systèmes informatiques en provenance de *différentes* organisations implique d'admettre la perte de contrôle sur l'ensemble du système. En

[1] Nous ne voulons pas parler ici des systèmes d'exploitation, mais bien d'infrastructures applicatives

n'étant plus maître de l'ensemble de l'application *en fonctionnement*, il faut partir du principe que le système avec lequel on collabore peut se révéler peu fiable, voire malveillant. Il faut donc pouvoir garder une intégrité fonctionnelle dans un environnement plus imprévisible.

- Des modèles de conception plus adaptés, où l'on cherche à s'affranchir des détails d'implémentation mais où l'on doit absolument garder une vision opératoire et flexible. On peut mentionner dans ce cadre les patterns de conception [Gamma et al., 1995], mais aussi la transition à des architectures focalisées sur les modèles d'applications et non les infrastructures [OMG, 2001]. C'est également un des aspects assez caractéristique de nouvelles méthodologies de développement comme Extreme Programming [Beck, 1999], qui prend l'acceptation du changement comme référent principal.
- Le besoin de nouvelles structurations, là où le modèle objet "pur" se révèle trop peu expressif car mettant trop les différentes classes et instances au même niveau de conception ou d'opération. On cherche à accroître la réutilisation, monter dans l'abstraction en décrivant directement événements, propriétés ou dépendances, mais donc hors du modèle en tant que tel. Les modèles de composants se rattachent aussi à ces préoccupations. Même si le langage du modélisateur permet d'exprimer des structures d'ordre de haut niveau, le passage au système opératoire fait perdre quasiment systématiquement cette sémantique (modularisation, aggrégations, ...).
- Une prise en compte du déploiement des entités logicielles. Là où les applications sont de plus en plus modulaires, le contrôle et la localisation des instanciations, la prise en compte de la tolérance aux pannes et de la répartition de charge demandent à être envisagées de bout en bout, de la conception à l'exécution.

2.1.3 Sécurité et modularité

Force est de constater également que la sécurité d'une application complexe est de moins en moins réductible à la sécurité de l'ensemble de ses parties. La dynamique de plus en plus présente dans les interactions entre composants complique souvent singulièrement la tâche.

- La validation formelle devient de plus en plus complexe quand il s'agit de qualifier non pas une entité mais plusieurs systèmes en coopération (voir par exemple [Yoo, 1999] pour une proposition prenant en compte ces contraintes et montrant la difficulté de l'entreprise).
- La maîtrise conceptuelle complète de l'application n'est pas toujours possible, et même non souhaîtable pour conserver le plus de possibilités d'évolution possibles.

Ceci ne va pas sans risques. Pour preuve, on peut citer le commentaire récent d'experts en sécurité informatique. Ils évoquaient des failles dans des architectures logicielles de plus en plus modularisées dans des applications (comme Netscape) ou des systèmes (tel que

Windows) [Ellis et Lasser, 2000] :

> *These new bugs exist because the people who understood the different components did not or could not see how the interaction of these various pieces could cause trouble. The most dangerous, well-concealed, complex, and noteworthy security flaws in the future will be of this sort [...] In the past, many security holes were problems in the software itself : the latest server module does not check the length of a string and allocates too much memory, or some new killer app does not check the permissions of the file, and, having too much privilege on the system, overwrites it. This new class of holes is different : a programmer doesn't understand what somebody else was thinking or expecting, and fails to make a smooth transition between the two pieces of code. Sometimes the transition appears smooth, but the underlying complexity scuttles system security in the end.*

Notons au passage qu'il s'agissait encore dans ce commentaire que d'une application locale : les problèmes inhérents à d'authentiques architectures distribuées ne font que compliquer encore les choses.

2.1.4 Première synthèse

On pourrait résumer, d'une manière un peu grossière, les besoins de ces architectures en deux points : abstraction et adaptation. Abstraction, par la volonté de se concentrer de plus en plus sur le coeur d'un problème applicatif, et non plus dans des bases mille fois reprises. Adaptation, également, pour pouvoir établir des architectures "agiles" capables d'être remises en cause selon les besoins ou les risques et en minimisant les ruptures fortes. Si l'on dégageait des points plus concrets, on pourrait mentionner les préoccupations suivantes :

Une granularité trop fine. L'objet remplace maintenant la procédure ou la fonction, mais au delà, on n'a pas observé de véritable changement de niveaux - ou plutôt de changements d'échelle - dans les modèles admis.

Une rigidité des interactions. Les mécanismes de liaison entre entités restent très majoritairement assimilables à l'appel à un service auquel l'entité ne peut se soustraire. Les contraintes en matière de synchronisme, de localité et d'adaptabilité s'y expriment souvent maladroitement.

Peu de modèles globaux. On ne dispose que de peu de représentations pouvant aider à la compréhension de vastes systèmes logiciels *à un niveau opératoire*.

L'évolution à long terme est incommode. Les architectures se doivent de prendre en compte les changements futurs, que ce soient dans les protocoles, les besoins, les infrastructures ou même l'architecture, et rares sont les modèles qui prennent en compte réellement ces aspects.

On verra dans la section suivante que les agents fournissent une réponse possible à ces problèmes.

2.2 Les modèles à agents

2.2.1 Une vision décentralisée

Le principe fondamental des modèles à agents est celui d'une décentralisation, d'une distribution du comportement et des connaissances (ou du contrôle et des données, selon le point de vue retenu). Une telle hypothèse a l'avantage d'être particulièrement proche des problèmes et de champs d'applications réels, qu'ils soient purement informatiques, biologiques, sociaux ou physiques. Comme le résume [Meyer, 1989], *"Real systems have no top"* : les systèmes dépassant une certaine taille ou complexité ont de multiples points de contrôle ou d'action. Cette remarque vaut aussi pour la distribution : la plupart des systèmes réels sont naturellement distribués, par décentralisation, réduction du couplage entre composants ou entités, ou par le besoin de prise de décision locale.

Mais ce qui fait essentiellement la force de ce principe, c'est le choix de voir dans cette décentralisation non pas un élément négatif car difficile à maîtriser mais bien le biais de compréhension et de mise en oeuvre de tels systèmes. Ceci conduit alors à l'étude des mécanismes de coopération, d'interaction, d'émergence de comportement, etc.

On pourrait rétorquer - à juste titre - que ce point de vue plutôt général va conduire à une variété de modèles et de systèmes quelque peu intimidante.

C'est en effet le cas, mais on ne rentrera pas ici dans le débat sans fin de qu'est-ce *exactement* qu'un agent [Franklin et Graesser, 1997]. Il nous parait plus utile de simplement souligner l'étendue des inspirations : biologiques [Di Caro et Dorigo, 1998], informatiques, à base de connaissances, sociales...

Dans ce travail, nous supposerons une définition de l'agent la plus générale possible. Les a-priori que nous retenons par contre comme essentiels sont les suivants :

Les décisions sont prises localement. L'agent est l'entité d'observation et d'analyse primitive, et ses actions sont guidées par des observations et des raisonnements locaux, établis par l'agent par rapport à son contexte immédiat.

Le comportement est supposé autonome. En tant que concepteur, analyste ou observateur, on fait l'hypothèse d'une autonomie de contrôle de l'agent. Cette autonomie peut s'avérer plus ou moins importante en pratique (par exemple en la restreignant par construction), mais on lui *attribue* cette autonomie.

On suppose un environnement. Il est essentiel pour ancrer l'agent dans l'application, de permettre de définir correctement les actions et mécanismes de communications possibles. On ne suppose pas par contre un type d'environnement particulier, comme un environnement physique, ou un contexte social.

L'agent est une entité sociale. Pour permettre une action de notre part sur le système (en tant qu'observateur ou analyste) on fait l'hypothèse de départ que les agents ont un

comportement collectif. Que celui-ci soit volontaire ou non, symbolisé ou non, n'entre pas en ligne de compte.

Ce qui permet d'obtenir par la construction même du système, des propriétés en accord avec nos préoccupations : robustesse, adaptabilité, flexibilité.

2.2.2 Intérêt d'un niveau social

[Jennings, 2000] va plus loin en mettant en avant les approches multi-agents comme solution privilégiée pour aider à analyser, concevoir et construire des systèmes (logiciels) complexes. Pour étayer son hypothèse, il fait plusieurs remarques que nous reprendrons ici, en premier lieu sur les similitudes entre architectures complexes et organisations d'agents :

- Une correspondance forte existe entre la notion de sous-système et d'organisation d'agents : dans les deux cas, il s'agit d'entités constituantes, agissant (et interagissant) selon leur rôle dans une structure d'échelle supérieure.
- L'inter-relation entre les sous-systèmes et leurs composants constituants s'envisage d'une manière plus simple et naturelle si on les décrit en tant qu'interactions de haut niveau. Pour citer G. Booch : *At any given level of abstraction, we find meaningful collections of entities that collaborate to achieve some higher level view*. Cette vue et ce niveau d'abstraction sont exactement ceux offerts par les agents.
- Un trait caractéristique des systèmes complexes est la nécessité de pouvoir modifier le réseau de relations entre les composants. Ceci peut également être une question d'angle d'observation, pour mettre en exergue des jeux de propriétés distincts des entités du système. Cela peut-être aussi une question d'échelle, pour par exemple voir une seule unité conceptuelle depuis divers niveaux d'abstraction.

Au delà de ces points de convergence, le fait de retenir un modèle multi-agent implique d'accepter des différences par rapport à un modèle centralisé, beaucoup plus profondes, dans la nature des liaisons entre les entités, et sur la structuration globale du système, en particulier :

Flexibilité des interactions : il est impossible de connaître a priori tous les liens possibles : les interactions ont lieu à des moments non prévisibles, et pour des raisons non prévisibles. Les entités *doivent* s'adapter. L'implicite ici est que cette obligation est finalement une question de survie : la première tâche pour un agent, c'est de réussir à maintenir son intégrité et sa cohérence dans un environnement non contrôlable.

Contrôle et organisation : l'apparition d'un "ordre" dans le système peut arriver bien plus tard qu'à la conception, par émergence dans le système en fonctionnement, et doit pouvoir être gérée *dans le contexte courant*.

Pourquoi alors ne pas voir les systèmes multi-agents comme une étape supplémentaire après le "Knowledge Level" de [Newell, 1982] : la mise en place d'un niveau social entre

entités relevant du niveau inférieur (voir le tableau 2.1 pour une comparaison des deux niveaux).

Dimension	**Description**	**Knowledge Level**	**Social Level**
Système	Entité à décrire	Agent (asocial)	Organisation d'agents
Composants	Eléments primitifs du système	Buts, actions	Agents, interactions, dépendances, relations organisationnelles
Loi de composition	Comment les entités sont assemblées	Variables	Rôles, normes sociales
Comportement	Comment le comportement du système dépend de sa composition et de ses entités	Principe de rationalité	Principe de rationalité organisationnelle
Medium	Eléments à traiter pour obtenir le comportement désiré	Connaissance	Organisation, obligations sociales, influence, évolution de l'organisation

TAB. 2.1: Comparaisons entre Knowledge et Social Level, d'après [Jennings, 2000]

2.2.3 Des systèmes délicats à maîtriser

Cette liberté d'expression et ce confort de modélisation ne vont pas sans inconvénients. N. Jennings ne cache pas dans sa proposition les difficultés soulevées par ce type de modèles :

Versatilité des interactions et de leurs effets. Les schémas d'interactions sont de haut niveau, et le déclenchement d'iceux est décidé dans l'immense majorité en fonction de changements locaux dans le système. Cela donne donc lieu à des échanges complexes (négociation, ...) entre entités. La prévision, la traçabilité et l'analyse des dialogues sont particulièrement difficiles. En fait, déterminer par avance la complexité apparente (simple requête ou long échange de coopération) et le résultat final d'une interaction est souvent ardu.

Imprévisibilité du comportement global du système. Les systèmes multi-agents font classiquement apparaître des comportements *émergents* où l'action globale du système n'est pas réductible aux lois des entités individuelles. La prévision ou la validation du système est là encore particulièrement difficile [Parunak et Vanderbok, 1997].

Pour éviter ces effets difficilement maîtrisables, la proposition faite est de *rigidifier* des parties bien précises de ces systèmes : ontologies, protocoles d'interactions précis, objectifs

attribués à priori. Tout ceci permettant de ramener l'organisation globale vers une forme homogène plus facile à maîtriser.

2.3 Les écueils des modèles centrés agent

On doit néanmoins tempérer l'analyse de N. Jennings quand il s'agit de résoudre ces points supposés négatifs des systèmes multi-agents. En effet, les reproches formulés aux architectures agents sont un peu paradoxaux car pour beaucoup inhérentes à cette classe de systèmes. Il est certain que revenir à un seul point de contrôle peut éviter l'apparition de comportement émergent. Mais cela est-il souhaitable ? A ce titre le système mériterait-il vraiment le qualificatif d'"orienté agent" ?

En fait, les remèdes proposés par N. Jennings sont à la base un retour à une situation classique et donc supposée maîtrisable. Par exemple, l'auteur évoque l'utilisation de protocoles d'interactions dont les propriétés sont analysables formellement, ou bien une structure rigide ou prédéfinie pour le système afin de mieux contrôler son évolution. D'une part, la définition de schémas d'interactions suffisamment souples pour être intéressants dans un contexte agent et définissables dans un cadre formel, s'est avérée particulièrement délicate en pratique [Pitt et Mandani, 1999]. D'autre part, l'émergence elle-même peut être un outil de résolution de problème singulièrement puissant et tout à fait désirable dans le système [Muller, 1998].

2.3.1 La prise en compte de l'hétérogénéité

En effet, les systèmes multi-agents classiques prônent une autonomie totale des agents, sans faire intervenir de contraintes organisationnelles. En parlant, comme souvent, *d'agent autonome*, on fait - plus ou moins implicitement - le choix d'une vue centrée sur l'agent. Les interrogations du concepteur en découlent : Quels objectifs l'agent doit réaliser ? Quels sont ses possibilités de raisonnement ou d'apprentissage ? Quels sont ses modèles de connaissances ? Comment mettre en place ses mécanismes de communication ? Si une coopération entre agents est envisagée dans l'application, quelles interactions faut-il mettre en place ? Quel mécanisme d'interprétation et de représentation de ces interactions doit être présent dans l'architecture de l'agent ?

Cette autonomie est prônée comme le mode le plus souple pour la construction de systèmes collectifs, vu que l'on suppose que le concepteur du système a toutes les latitudes possibles concernant le choix des techniques de communications et des architectures envisagées. Mais cette position n'est pas tenable dans le cas général : si les agents communiquent dans n'importe quel langage et n'importe comment, sans savoir qui sait faire quoi, alors le système multi-agent ne peut pas fonctionner. La structuration est anarchique.

Classiquement, on fait alors la supposition que l'on n'aura à construire qu'un système

bien particulier, dans lequel on mettra en place un modèle homogène, unique, de communication ou d'action. L'avantage de cette approche est que l'on a, à l'issue de la conception de l'application agent, une connaissance extrèmement précise et fiable de l'agent - ou du système multi-agent - que l'on a construit.

Néanmoins, il faut prendre garde à une caractéristique majeure de cette façon de concevoir ces systèmes : la force de l'intégration entre entités vient avant tout des implicites du concepteur. Revenons sur les conséquences de ce style de construction de systèmes multi-agents : quand, en tant que concepteur, nous faisons le choix *préalable* d'un modèle d'architecture interne et de communication, nous supposons nécessairement cette homogénéité entre nos agents dans leur manière d'interpréter informations et interactions. Si nous cherchons à concevoir des systèmes ouverts (où nous ne sommes plus forcément les seuls concepteurs à agir) ou à faire évoluer un système (où nous pouvons avoir à revenir sur les choix de modèles), nous sommes tenus de respecter scrupuleusement la sémantique choisie. Adopter un point de vue centré sur un agent nous contraint alors dans nos choix pour tous les autres agents.

Malgré tout, nous devons fréquemment intégrer nos agents avec d'autres. Par exemple, on voit fleurir des projets de négociation pour le commerce électronique [Karjoth, 2000], de services de proximités évolués à base d'agent [Willmott et Burg, 2000]. Toutes ces approches supposent l'intégration d'agents hautement spécialisés créés par des fournisseurs distincts. Comme nous l'avons vu, les interactions entre agents se situent à un niveau d'abstraction beaucoup plus élevé que des systèmes logiciels classiques. Il faut alors, pour garantir l'interopérabilité, se replier sur des définitions *normatives* des échanges entre agents, et *extrinsèques* à l'application. Bref, on en revient au besoin d'une standardisation.

Ce n'est néanmoins pas une panacée, car une standardisation "agent" pose des problèmes d'expression particulièrement délicats. Il faut ainsi spécifier des protocoles d'interactions singulièrement plus souples que des protocoles informatiques standards, mais tout en devant vérifier que l'interprétation des échanges reste cohérente entre les diverses entités : sans sémantique commune, point d'échanges fructueux.

On se trouve finalement pris en tenaille entre la volonté d'obtenir en pratique des systèmes *opérationnalisables*, et dans le même temps de conserver les bonnes propriétés d'adapation, de souplesse, de couplage faible qui sont la justification des systèmes multi-agents.

Cette idée de normalisation a donc introduit des contraintes sur la construction de systèmes multi-agents. Mais ces contraintes n'étant pas directement présentes au sein du système multi-agents[2] en tant que concepts manipulables, le système est figé.

[2]Ne serait-ce que parce que les agents n'ont plus l'opportunité de se mettre d'accord pour adapter les règles

2.3.2 Contraintes de conception

Une autre faiblesse que l'on peut identifier dans les modèles de systèmes multi-agents classiques est le peu de prise en compte de problèmes de génie logiciel. Nous parlions plus avant du besoin d'en respecter les contraintes, et on pourrait mentionner en particulier les points suivants :

Sécurité des applications. Dans les systèmes multi-agents classiques, un agent peut communiquer avec n'importe quel agent. Cela comporte donc des failles de sécurité importantes. Par exemple, il appartient au concepteur d'un agent de vérifier que celui qui lui demande des informations a bien les qualités requises pour obtenir cette information. Aucune vérification n'est faite au niveau du système d'une manière quelconque.

Construction d'applications par frameworks ou composants. La construction de logiciels modernes passe par le développement d'architectures fondées sur la relation entre frameworks et composants. Les frameworks définissent des architectures abstraites dans lesquels des composants viennent "s'enficher". Ce type de construction est rendu impossible dans des systèmes multi-agents classiques. Souvent, il n'existe qu'un seul "framework", celui de la plate-forme d'exécution elle-même.

Modularisation. Dans les techniques de développement logiciel on veut aussi pouvoir regrouper les entités qui vont ensemble sous la forme de modules. Ces modules définissent, comme pour les communautés d'utilisateurs, des règles de visibilités : certains aspects des entités ne sont visibles par toutes les entités du même module alors que d'autres sont accessibles par tous les modules. Notons simplement que cette notion de module n'est généralement pas dynamique dans les démarches de génie logiciel. Par exemple, dans les langages à objets, une classe ne change pas de module au fur et à mesure de son utilisation : son module fait partie de sa définition intrinsèque, tout comme un objet ne changera pas de classe (i.e. ses actions, réactions et interactions restent les mêmes).

Séparation des ontologies. Ceci est particulièrement complexe dans le cas des systèmes à agents où chaque sous-système peut avoir à manipuler sa propre ontologie. Pour préciser cette idée, on peut faire le parallèle avec la transition progressive qui s'amorce dans le Web tel qu'on le connait depuis 1991 (dont les contenus accessibles se caractérisent par une expression structurée par leur forme, et non leur fond). C'est ce que l'on commence à appeler le "Web sémantique" [Berners-Lee et al., 2001] : exprimer les contenus directement dans le langage de données adéquat. Bref, évoluer vers des modèles de représentation de la connaissances plus explicites, et permettant un traitement automatique et une interprétation ne nécessitant pas systématiquement une intervention humaine.

En résumé, plus on utilise un point de vue "centré-agent", plus l'on fait de suppositions sur l'interprétation et la mise en oeuvre de l'activité collective du système, et plus on complique l'intégration d'agents structurellement différents. Les systèmes deviennent fermés et homogènes dans des applications de taille réaliste, alors que l'on promettait justement de nouvelles possibilités d'adaptation et d'intégration.

2.4 Vers un retour à la socialité ?

Comment se dégager de ce mauvais pas ? Une manière de faire pourrait être de revenir finalement à l'essence de ces systèmes. Si nous parlons de systèmes multi-agents, de modèles décentralisés, d'interaction ou de de coopération, c'est que nous voulons exprimer des concepts qui n'apparaissent pas directement dans des systèmes classiques (orientés objet ou d'intelligence artificielle). Ce que nous cherchons à exprimer, outre cette délocalisation de l'action, c'est une approche "sociale" dans l'interaction entre nos entités. Pourquoi alors ne pas s'appuyer sur ce point fondamental pour résoudre ces problèmes d'ouverture ?

Si nous revenons sur notre champ d'étude, que pouvons-nous constater en examinant de manière très générale les travaux effectués ? Ce qui frappe avant tout, c'est l'hétérogénéité, qui peut se décliner sous plusieurs axes.

Hétérogénéité des domaines applicatifs Les systèmes multi-agents et l'intelligence artificielle distribuée ont été appliqués à une gamme étonnante de champs : simulation orientée entités, négociation, logistique et systèmes de production, personnalisation de la relation à l'utilisateur, support au travail collectif, contrôle temps-réel adaptatif, ...

Hétérogénéité des ontologies manipulées Cela fait suite au point précédent, on s'aperçoit vite que l'on devrait gérer des domaines de connaissances disparates. Et parfois des agents doivent en être parties prenantes de plusieurs simultanément. Comme il apparaît illusoire de disposer d'une ontologie unique, on s'achemine vers une information structurée en îlots, tous cohérents individuellement.

Hétérogénéité des modèles d'interactions On y trouve des mécanismes d'enchères ou de négociation, des modèles économiques, des systèmes de communication non symbolique par l'environnement, des protocoles auto-adaptatifs, des reconnaissances d'intention, ...

Hétérogénéité des architectures d'action Il peut s'agir de systèmes purement réactifs ou situés, d'entités à planification non symbolique, d'architectures de contrôle par règles, de modèles à états mentaux, ...

Modularisation et sécurisation Afin de permettre des activités simultanées indépendantes sans risque d'interférénces ou de malveillance, même si elles obéissent au même modèle d'application.

Comment concilier une telle variété avec la volonté de pouvoir capitaliser au besoin sur ces différentes réalisations ? L'objectif est bien de se placer dans le cas de systèmes *ouverts* et *hétérogènes*. L'hypothèse que nous ferons est d'utiliser une structuration exprimable *au niveau du système multi-agents et non de l'agent* : un modèle social.

Deuxième partie

Vers un modèle organisationnel

Chapitre 3

Les approches sociales des architectures logicielles

Nous avons vu que prendre un point de vue social pouvait être une clé de la conception et de la mise en oeuvre de systèmes multi-agents de taille conséquente. Nous allons parcourir ici quelques approches ayant été déjà proposées pour adopter et exprimer ce niveau social. Nous verrons dans ce chapitre que ces modèles, bien que tous orientés vers la conceptualisation du lien "social" entre agents, vont prendre des points de vue fort différents sur ces relations entre entités. On pourra distinguer des modèles à forte connotation *structurelle*, où l'agencement des entités (et donc de l'application) est avant tout affaire d'expression d'un modèle d'organisation bien défini, pour pouvoir y exprimer les interactions et situations de coopérations possibles.

On pourra trouver également des approches plus *fonctionnelles*, où chaque entité devra être conforme à une certaine attente quant à ses objectifs, manières d'interagir, obligations ou compétences. Nous verrons aussi que l'application du principe d'autonomie permet de situer la représentation et l'utilisation de ces concepts de manière intrinsèque ou non au système. Le concepteur ou l'agent conçu, voire les deux, peuvent manipuler ces représentations. Enfin, un autre point de différenciation que nous remarquerons est le champ d'action de ces modèles : certains se placent au niveau des connaissances des agents, d'autres étudient l'ensemble du système. Il nous a paru intéressant de mentionner également des propositions en rapport dans le monde de l'objet, même quelque peu excentrées par rapport à notre domaine d'étude multi-agents.

Nous allons donc brosser un aperçu de quelques modèles représentatifs (partant des inspirations sociologiques et descendant jusqu'au à un niveau d'implémentation), et tenter de proposer, en conclusion de ce second chapitre, une vue plus synthétique de ces modèles d'inspiration organisationnelle.

3.1 Repères, analogies et inspirations

Pour rassembler en un mot les modèles qui vont suivre, on dira simplement que l'étude des *organisations* guidera notre discours. Pourquoi ce parti-pris plutôt que parler de coopération, d'intention ou d'interaction ? Ce qui nous motive ici est la relative neutralité induite. Si l'on prend une définition classique de l'organisation, celle de [Morin, 1977] :

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

Ce qui est manifeste ici est la possibilité d'étudier l'organisation en soi, mais indissociablement de son état d' "agencement" d'entités primitives, son caractère à la fois structuré et adaptatif, l'influence réciproque de l'organisation et de ses constituants. Nous prendrons donc l'organisation comme cadre d'analyse, point de départ de l'examen que nous allons faire de différents modèles sociaux informatiques.

L'organisation qui nous préoccupe est donc à la fois un modèle, décrivant les échanges potentiels entre composants, mais également le schéma explicatif que l'on appliquera sur ce système. C'est là où des métaphores sociologiques [Schillo et al., 2000] ou éthologiques [Drogoul et Collinot, 1997] prennent tout leur sens.

Mais revenons un instant sur notre avertissement liminaire : nous nous plaçons dans ce travail dans un cadre purement informatique. Notre objectif est bien d'explorer des solutions computationnelles, qui doivent se plier bon gré mal gré aux contraintes des machines. Nous nous voyons donc forcés, quelles que soient nos inspirations, d'en affadir les termes pour en tirer des modèles qui soient - au final - opérationnalisables. Si nous nous permettons d'utiliser des termes comme organisation, rôle, interaction, communication, c'est bien parce que nous les situons dans ce contexte et non pas dans les acceptations d'origine de leurs domaines comme la sociologie, l'éthologie ou les sciences cognitives.

Ceci dit, rien ne nous empêche malgré tout de revenir sans prétention sur certaines de ces définitions originelles. On verra que les grands courants d'analyse se reflètent finalement plutôt fidèlement dans les modèles informatiques qui vont suivre.

3.1.1 La structure sociale comme primitive

L'intérêt pour l'organisation en tant qu'angle d'analyse directeur n'est pas une approche récente. En fait, on pourrait remonter à plus d'une centaine d'années, quand le sociologue E. Durkheim [Durkheim, 1895] puis ses successeurs envisageaient la structure collective comme point d'étude du fait social. Chez Durkheim, l'individu accomplit ses engagements

en remplissant des devoirs définis en dehors de lui, par son environnement moral ou normatif.

Pour schématiser, on pourrait dire que dans cette approche, la sociologie se doit d'étudier les structures sociales à part entière, indépendamment du fait que des individus y participent. Le sociologue doit aller traiter ces structures sociales (groupes, institutions, symboles, valeurs, normes, statuts, et dynamiques) comme si elles étaient des faits physiques ou naturels.

Bien que l'émergence soit parfois évoquée comme une base ontologique de ces hypothèses, leurs mécanismes spécifiques n'en sont généralement pas étudiés. La structuration sociale étant le principal objet d'analyse, les modes individuels d'explicitation d'un phénomène émergent y seraient vus comme réductionnistes. De même, si l'existence de mécanismes comme l'internalisation (par laquelle les entités internalisent la représentation d'une structure sociale externe) est supposée et utilisée, ces modèles sont souvent peu intéressés par la nature même de ces mécanismes, en dehors du bouclage qu'ils permettent.

Une exception est peut-être la question de la reproduction de l'ordre social, avec l'étude du maintien (ou de la survie) de ces structures. Dans la sociologie anglo-saxonne, cet aspect est principalement tributaire de l'influence de T. Parsons [Parsons, 1951], l'école française étant plus marquée par l'analyse de M. Crozier et E. Freidberg sur les organisations comme systèmes d'actions [Crozier et Freidberg, 1981] :

> *L'organisation est un construit politique et culturel, un moyen dont les acteurs sociaux se dotent pour 'régler' leurs interactions afin d'obtenir le minimum de coopération nécessaire pour atteindre des objectifs collectifs*

Bref, l'analyse est là avant tout structurelle et globale. Même si la "conscience collective" permet la perception individuelle du fait social, elle guide et formate les contraintes qui pèseront sur les individus.

3.1.2 La structure sociale comme construction individuelle

A l'opposé, le courant subjectiviste suppose que les structures sociales n'existent pas en tant que telles, que seuls les individus existent, étant entendu que les groupes ne sont formés d'autre chose que d'individus. Le sociologue doit alors se focaliser sur les perceptions subjectives de l'individu et ses interprétations de la structure sociale[1].

La définition (subjective) de l'action sociale se fait via la prise en compte de l'autre : de par son existence avant tout, mais aussi par sa symbolisation et la reconnaissance des actes, intentions et attentes. Le tout, pris dans une boucle rétroactive, va structurer les actions de l'individu. L'approche remonte à Weber [Weber, 1995] et a été raffinée après 1945 [Winch, 1958] via la phénoménologie. Les théories subjectivistes n'évoquent que rarement

[1] D'où d'ailleurs un certain réductionnisme implicite

les phénomènes de construction émergentes par cet accent porté sur l'individu.

Pour résumer, la structure sociale est ici internalisée et interprétée par les individus, à partir de leurs motivations et comportements propres comme à partir que ceux des autres.

3.1.3 Modèles fédérateurs et analyse des organisations

Bien entendu, il ne s'agit là que de grands traits, la réalité est plus nuancée, et on a vu dans les dernières décennies plusieurs propositions pour réconcilier ces deux approches, comme [Giddens, 1984]. Une manière d'éviter cette opposition est de déplacer l'analyse vers les interactions effectives, ce qui permet de subsumer individus et structures sociales [Collins, 1981] [Knorr-Cetina et Cicourel, 1981], en rassemblant des approches de sociologie classique, d'ethnométhodologie, de l'analyse conversationnelle. Il n'est pas surprenant, par l'accent mis sur l'interaction (mais également sur l'émergence), que des études en systèmes multi-agents y aient vu des points de convergence. En particulier, pouvoir isoler l'interaction comme niveau d'analyse à part entière (comme base pour les structures sociales ou comme préalable aux comportements des individus) est une préoccupation très proche des concepteurs de langages de communication agent.

Il reste alors à faire le lien entre ces modèles génériques et les structures que l'on trouvera très concrètement dans ces organisations qui cadrent l'activité collective dans les sociétés humaines. Prenons une définition classique de [Parsons, 1951] :

> *Les organisations sont des systèmes, fonctionnellement différenciés, du système social englobant. Elles ont donc les mêmes propriétés formelles que les autres systèmes sociaux même si elles sont conçues en vue de la réalisation de buts spécifiques tels que la production des biens et des services, l'éducation ou la défense. Ces buts, les organisations les réalisent à travers les relations qu'elles entretiennent avec l'environnement au sein duquel elles opèrent. Elles ne sont donc pas isolées, mais évoluent au sein d'un univers composé d'autres sous-systèmes avec lesquelles elles sont fonctionnellement en rapport.*

Une des difficultés d'analyse des organisations est justement la déviance d'organisations réelles, observée par rapport à un modèle abstrait ou un idéal-type[2]. L'enjeu est alors de savoir prendre en compte à la fois la structure officielle, souvent figée et hiérarchique et la présence de groupes officieux, dynamiques et non formalisés. C'est ce que les groupes gravitant autour de T. Parsons résumaient en trois règles :

- Toute organisation crée des structures informelles
- Ses buts sont modifiés, déviés, abandonnés par des processus internes
- Ces modifications se font par des processus informels.

Cette multiplicité de structures qui se superposent dans une organisation (au sens strict) a souvent été observée : [Lafaye, 1996], citant [Mayo, 1933], remarque :

[2]Pour reprendre les termes de Weber

> *Il existe une vie de groupe au sein des ateliers, [...] l'individu n'existe pas seul, il est pris dans une pluralité d'appartenances collectives internes à l'entreprise. Ils identifient toute une organisation informelle qui ne coïncide pas exactement avec l'organisation formelle et technique de l'entreprise.*

Ayant accepté ces structures enchevêtrées, on peut alors s'en servir pour catégoriser comportements et règles sociales, comme l'a fait [Gouldner, 1954] sur les règles ayant cours dans les bureaucraties :

- Règles artificielles, extérieures, qui ne sont que peu respectées en pratique.
- Règles punitives, imposant contraintes et obéissances sous peine de sanction.
- Règles représentatives, fondées sur l'élaboration collective et le consensus.

Dans cette optique, l'apport essentiel de [March et Simon, 1958] est que via leur notion de rationalité limitée[3], il devient possible de réintégrer les comportements non rationnels (si on les envisage par rapport au modèle global) dans le cadre d'analyse, en les resituant systématiquement par rapport au contexte des individus.

A partir de cette séparation des préoccupations, l'examen des normes régulant, maintenant ou définissant la fonction des structures se simplifie. L'interaction est ainsi structurée à partir de normes collectives, mais celles-ci peuvent relever de différentes "sous-sociétés". L'individu est donc le "produit des tribus" [Rocher, 1968].

3.1.4 Des organisations artificielles ?

Même si certaines théories de l'action peuvent donner l'impression d'une opérationalisation aisée dans des entités informatiques, la pratique s'avère délicate si l'on veut sortir de la simulation d'un phénomène social et passer à une véritable conception de systèmes sociaux artificiels.

[Heitsch et al., 1999] se sont ainsi posé la question de l'opérationalisation d'un modèle sociologique, le cadre étant la construction de sociétés artificielles ou l'étude de communautés hybrides agents/humains. Le problème est de répondre à une double contrainte : trouver un modèle social suffisamment formel et à la fois opératoire pour pouvoir être exécutable mais également validable par les sociologues. Le modèle d'organisation aboutit effectivement à une solution opératoire permettant simulation et analyse de comportement du système, mais le lien entre les comportements et compétences identifiées globalement et les comportements à mettre en oeuvre au niveau des entités n'est pas fait. En effet, il s'agit ici d'une expression informatique d'un modèle sociologique dans la problématique de la sociologie et non l'inverse.

Cela explique que la grande majorité des approches sociales dans les systèmes complexes logiciels parte d'un modèle mathématique ou informatique et d'une simple inspiration -

[3] Qui, il faut le rappeler, a bien été forgée face à ce problème d'analyse de structures organisationnelles humaines.

plutôt que d'un modèle - sociologique. Il faudrait même, et nous terminerons cette parenthèse sociologique ici, se défier d'une transposition trop hâtive entre l'analyse de l'action sociale humaine et la conception de systèmes collectifs artificiels, écueil bien résumé par [Ferrand, 1996] :

> *L'objectif premier de la sociologie est l'analyse et la compréhension de l'action sociale, et non la construction. On peut donc s'interroger sur la pertinence des distinctions opérées entre les deux niveaux d'explication de Weber et Durkheim, lorsqu'elles sont appliquées à l'ingéniérie des systèmes collectifs artificiels. En particulier, le développement de la "conscience d'un agent" n'a a priori pas de sens dans la mesure où leur construction doit être achevée pour envisager l'utilisation effective du système collectif artificiel.*

3.2 Modèles sociaux dans les systèmes à agents

Posons nous maintenant le problème de l'élaboration de modèles sociaux pour des systèmes purement informatiques et multi-agents. Assez logiquement, l'accent mis sur l'interaction et la coopération dans les systèmes multi-agents a entraîné la proposition de nombreux modèles et expressions spécialisés de concepts sociaux depuis l'émergence de la discipline. On trouvera d'ailleurs une synthèse des premières approches de modèles d'organisations dans les systèmes multi-agents dans [Bouron, 1992].

On pourrait dégager des travaux du domaine deux grandes manières d'envisager les sociétés d'agents :

Une vision principalement *comportementale* et fonctionnelle, où l'agent est à la source de la structure sociale, et où l'activité dans l'organisation multi-agents s'identifie à de grands axes fonctionnels, avec des expressions en termes de tâches, objectifs, contrats, ...

Une approche plus *structurelle*, où l'on essaie d'identifier ou de concevoir une ossature organisationnelle pour le système, qui permette d'exprimer plus facilement les interactions, communautés, dépendances entre agents. L'agent est vu plutôt comme partie prenante d'une organisation, devant y respecter les normes et guidé par celle-ci.

A cela s'ajoutent des caractéristiques plus transversales, comme l'étude de la dynamique de ces organisations, et en particulier les phénomènes d'émergence ou de réorganisation. On pourrait aussi citer les différentes manières de situer la connaissance organisationnelle : posée par le concepteur [Carle et al., 1994] ou bien construite par les agents. Avant d'entamer ce parcours des modèles sociaux, notons les caractéristiques essentielles relevées par [Foisel, 1998] dans son analyse des organisations multi-agents :

- La dualité entre structure statique de description de société abstraite et dynamique du processus d'organisation ou de réorganisation.
- L'origine non définie du processus d'organisation, qui peut être imposé par le concepteur ou initié, construit et adapté par le système lui-même.

– La nécessité d'adaptation d'une organisation, due à l'absence d'une structure optimale : la société d'agents doit pouvoir remettre en cause son fonctionnement pour mieux s'adapter à un environnement changeant.

Comme les modèles sociaux en systèmes multi-agents sont légion et ne peuvent être catégorisés de façon absolue, nous allons plutôt dégager quelques familles de modèles, en y retraçant les grandes caractéristiques que nous avons évoquées.

3.2.1 Approches comportementales et agent-centrées

Modèles de tâches et connaissances

Une première façon d'envisager l'aspect social d'un système multi-agent est de le situer au niveau de l'agent lui-même. On fait la supposition d'un agent cognitif, avec un modèle de représentation de connaissance. Le contexte social de l'agent fait alors partie de sa représentation du monde : accointances, motivations, protocoles d'interaction à observer, évaluation d'autres agents ...

L'organisation est alors définie comme l'ensemble des connaissances et hypothèses formées par les agents sur les croyances et actions des autres agents. Ces interrogations, qui peuvent être remises en cause, forment la base de la dynamique de l'organisation. Il y a donc réévaluation permanente de la société d'agents à partir des actions de ses membres, actions venant elle-mêmes des connaissances sociales et de leur évolution, et des activités à assurer :

> *Si l'on se place par rapport au concept de tâche, l'organisation désigne les processus de coordination qui permettent la décomposition, l'allocation et l'accomplissement des tâches de façon cohérente. [...] Si l'on se place par rapport au concept d'agent, l'organisation détermine les statuts et comportements sociaux des agents (les rôles et les relations qui permettent d'unir les agents au sein d'un groupe que ce soit vis à vis de la décision (par l'autorité) ou la coordination (via l'engagement) [Bouron, 1992]*

On peut citer également [Lechilli et Chaib-draa, 1996], qui définissent le rôle par l'ensemble des tâches qu'effectue un agent. Cela leur permet d'effectuer la mesure d'une structure proto-sociale (uniquement relationnelle) en étudiant la différenciation entre rôles et les dépendances impliquées. La difficulté apparaît dans des systèmes à architectures hétérogènes, où cette mesure nécessite de faire appel à une évaluation externe du concepteur.

L'organisation n'a d'ailleurs pas forcément d'unité d'objectifs : il s'agit plus généralement d'un ensemble d'intérêts non nécessairement convergents. C'est à chaque agent de trouver compromis entre le respect des obligations et ses aspirations, quitte à réévaluer son appartenance ou sa position. On peut alors voir l'organisation comme une *politique de choix* : étant donné un problème à résoudre, comment trouver l'agent le plus apte à prendre en charge la résolution d'une partie de ce problème et comment organiser la coopération avec les autres agents responsables d'autres sous-parties [Cammarata et al., 1983]. Ainsi,

dans [Durfee et Lesser, 1991] l'organisation représente le contrôle de la coordination, par les connaissances qu'ont les agents sur les rôles, objectifs et responsabilités des membres du système dans une résolution de problème.

Dans MACE [Gasser et al., 1987], l'organisation n'existe que dans les croyances et engagements de ses agents membres. Ici, le concept est utilisé comme abstraction, pour fédérer les activités, et les traiter comme un même travail coopératif de plusieurs agents ayant des rôles déterminés. La structure sociale est présente au niveau de l'architecture interne de l'agent, mais exprimée dans un modèle bien défini, faisant appel aux concepts de membres, rôles, compétences, etc. On retrouve ici une propriété importante d'appartenance possible à plusieurs organisations simultanément.

On peut aussi mentionner le modèle SAM [Bouron, 1992], qui exprime une population d'agent devant réaliser des tâches sociales, mais en la situant uniquement au niveau des engagements distribués dans l'ensemble du système :

> *Dans cette étude, nous ne considérons pas une organisation comme une structure globale, mais comme l'ensemble des engagement tenus par différents agents. La tâche sociale est définie comme une tâche nécessitant une résolution collective, et sert de base à la notion d'engagement de l'agent.*

On est ici typiquement dans le cas d'un modèle où le point de vue social est adopté par le concepteur, mais où la structure n'apparaît plus explicitement au niveau de l'agent sinon par l'expression qui en est faite dans les connaissances propres de l'agent. Les définitions d'une organisation sont assez restrictives, et sans doute influencées par l'exemple-type du modèle, les poursuites proies-prédateurs, ainsi :

- une seule organisation a pour objet l'accomplissement d'une tâche donnée,
- le concept de "leader" est pré-défini dans le modèle,
- l'agent ne peut appartenir qu'à une organisation,
- l'organisation est dynamique, et n'a comme durée que celle de la tâche dont elle assure la base de coopération,
- les rôles assignés aux agents ne le sont que dans la durée d'une organisation.

Un autre exemple de système construisant le système global principalement à partir de l'action individuelle se trouve dans le modèle STEAM de travail collectif [Tambe, 1997], plus précisément dans [Tambe et Zhang, 2000], qui en étudie la répartition de cette activité. Là encore, le comportement social est structuré avant tout par les comportements et objectifs individuels via des plans (partiellement) partagés, le tout étant basé sur la théorie de l'intention jointe. Le modèle STEAM permet de travailler avec des structures d'équipes homogènes ou hiérarchiques. Par exemple, une compagnie d'hélicoptères (le cas d'étude du modèle) se décomposant en deux équipes de respectivement 4 et 8 engins.

Le modèle se base sur une hiérarchisation d'équipes, avec des opérateurs définis pour caractériser les comportements (engagement, retrait, ...), associés à des préconditions d'ap-

plication ou de terminaison. Un rôle peut donc contraindre les opérateurs possibles. Un point important selon Tambe de cette approche est l'accent mis sur les rôles pour structurer l'activité, et considérés comme relevant de deux types possibles :

les rôles persistants Ce sont des associations à long-terme de rôles aux individus ou sous-équipes. Par exemple, on peut trouver une équipe de surveillance, un hélicoptère comme commandant de groupe.

Les rôles spécifiques à la tache Il s'agit des besoins et comportements à court terme, basés sur la tâche courante et la situation. Par exemple le leader d'un vol en formation est l'hélicoptère en tête. Ces rôles serviront à l'identification des accointances.

L'association entre rôles et individus se fait alors sur les compétences propres et par rapport à la situation. Cela ne peut donc pas être fait *a priori* à tout moment ; les individus peuvent donc avoir à se porter volontaires pour remplir un certain rôle. L'évolution de la situation peut d'ailleurs provoquer des réattributions. Comme le comportement social est explicité en tant que tel, une séparation est néanmoins possible au moment de la conception, et même de la mise en oeuvre, qui associe un framework pour la définition de 'teams', et un framework général pour l'architecture de contrôle. On pourra comparer aussi cette approche à la planification collective de [Kinny et al., 1994].

Relations de dépendances

Une autre approche importante en relations sociales dans les systèmes multi-agents est celle de [Castelfranchi, 1992], qui fait l'étude des relations de dépendances et du pouvoir social. L'auteur distingue deux grandes relations de dépendances :

- La dépendance sur une ressource, dans le cas où elle est détenue par un autre agent mais nécessaire pour atteindre un but.
- La dépendance sociale, où un agent dépend d'un autre pour l'exécution d'une action nécessaire à un de ses buts.

Un apport très intéressant est la démonstration que l'on peut réécrire la dépendance de ressource en une dépendance sociale (en la ramenant à un *contrôle* de la ressource). Dans [Castelfranchi et al., 1990], l'auteur va en fait un peu plus loin en distinguant le "pouvoir de" (l'agent peut atteindre un but b) du "le pouvoir sur" (où l'agent peut empêcher ou permettre à un agent d'atteindre b). Tout ceci permet de créer un réseau social expressif, basé sur plusieurs relations de dépendances distinctes (gêne, aide, compétence, ...).

Dans l'approche de [Sichman, 1995], les agents ont une représentation de leur modèle mental propre et des autres encore plus riche, exprimée en terme de de buts, d'actions, de ressource ou de plans. Cela permet à l'auteur d'identifier six situations-type de dépendances pour structurer le réseau social.

On peut maintenant voir la notion de rôle sous deux angles. Dans une première vision, il se définit comme un regroupement de différentes composantes (dépendances sociales, en-

gagements, conventions sociales), comme chez Castelfranchi et Sichman où l'ensemble des dépendances définit le rôle (quasiment par calcul).

Dans un deuxième cas, le rôle est prédéterminé par un ensemble de dépendances, d'engagements ou de conventions. [Castelfranchi, 1995] s'y rallie : le rôle d'un agent dans une organisation est définit par l'ensemble des engagements de cet agent envers l'organisation.

3.2.2 Structurations sociales

Le risque de ces approches centrées sur l'agent ou son comportement est d'aboutir à une représentation complètement implicite de l'organisation, ce qui empêche une analyse fine des phénomènes d'interaction ou de structuration dans les systèmes. Cette mise en garde n'est d'ailleurs pas nouvelle. Dès 1990 [Castelfranchi, 1992] met en garde contre la perte de vue du point de vue social dans les systèmes multi-agents. Il proposait alors quelques axes d'analyse en réponse aux premières interrogations sur la validité de métaphores sociales pour les agents intelligents, en faisant finalement une distinction sur *l'objet* de l'étude :

- Les projets dans lequel on envisage l'intelligence du point de vue de la mécanique d'une résolution de problème, avec l'architecture des systèmes, leur structuration, et catégorisé sous le thème de "l'approche D.A.I."[4].
- Les expérimentations centrées sur l'interaction et l'organisation sociale *en tant que tels*, avec en particulier le choix délibéré d'envisager le système social comme objet d'étude.

C'est dans ce dernier cas que la mise en garde suivante était faite : se focaliser trop tôt sur un modèle de raisonnement particulier est dangereux, car on court le risque d'aboutir à un système explicatif trop *rationnel* du comportement social. Et ce justement parce qu'on ne peut pas forcément attribuer à la société un comportement rationnel, pris dans le sens de celui accordé au niveau des individus. On retrouve d'ailleurs en filigrane le problème du changement de niveau entre individu et groupe, à manier avec précaution quand l'on suppose un système récursif unifié décrivant les comportements.

C. Castelfranchi remet également en cause le présupposé trop commun d'un monde en harmonie dans lequel tous les agents sont bienveillants et coopèrent, en soulevant deux problèmes concrets au niveau de l'entité ou de l'organisation :

- Pourquoi un agent entrerait-il dans une interaction socialisée ? Ce problème de la "sociabilité" est à mettre en rapport avec des expressions de l'intérêt d'une action de l'agent dans un cadre social, et ce du point de vue de l'agent ou de la société.
- Comment transformer une résolution de problème en une résolution de problème social, c'est à dire une situation où d'autres agents adopteraient tout ou partie de la tâche à leur compte ? Le problème est ici ce lui de la construction d'une situation de coopération.

[4] Le terme est sans doute un peu daté, l'essentiel est l'accent mis sur la filiation de l'I.A. "classique"

Il faut d'ailleurs rappeler que la modélisation de ce comportement d'agent est souvent purement cognitive, et n'implique pas forcément de communication au sens strict dans les faits. C. Castelfranchi aboutit alors à une définition en rapport de l'agent autonome, comme :

- ayant des buts propres,
- ayant un pouvoir de décision sur lesdits buts,
- ayant la possibilité, par décision propre, d'adopter les buts d'autres agents,
- définissant l'adoption comme une manière de permettre la satisfaction d'un but,
- contrôlant l'acquisition de sa connaissance, en faisant intervenir la confiance ou la crédibilité pour contrôler l'influence des autres entités.

Comme on le voit, la frontière entre modèle individuel (perçu et agit par l'agent) de la société et modèle global de l'organisation (observé et respecté par l'agent) devient singulièrement ténue. On trouve donc un certain nombre de modèles qui partent du principe que la structure sociale peut être étudiée, modélisée et appliquée aux agents en tant que telle.

La position frontière de l'interaction

La première façon d'affranchir le modèle d'agent de celui de l'organisation est d'exprimer l'activité sociale à travers sa manière de se manifester dans les échanges, c'est à dire via les interactions entre entités. [Foisel, 1998] définit ainsi l'interaction dans un contexte de coopération comme :

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

Concrètement, cela consiste à faire l'hypothèse que les comportements sociaux ne sont pas seulement décrits au niveau des agents mais aussi au niveau du système, via des schémas d'interactions. Les dits schémas sont définis au niveau abstrait comme une motivation, un ensemble de participants et un protocole d'interaction. Les participants abstraits sont en fait mis en correspondance avec un ensemble de rôles concernés que les agents peuvent avoir à tenir dans la mise en oeuvre réelle. Ces rôles servent alors de points d'accroche pour l'expression des interactions. La distinction est alors nécessaire entre la société organisée et la description abstraite de la structure organisationnelle.

Cela peut être utilisé en tant que modèle de conception, ou bien a posteriori, en analyse du système. Par exemple, [Baeijs et Demazeau, 1996] analysent l'impact d'une organisation sur les interactions en l'étudiant sous l'angle de la coopération et coordination observée. [Sichman, 1995] propose d'expliciter la dynamique des organisations multi-agents en la réexprimant dans les mêmes modèles d'interaction. Il s'agit d'utiliser des protocoles d'interaction types, non pas pour remplir un objectif, mais bien pour modeler la société elle-même : entrer dans la société, en sortir, construire des coalitions, et cela à l'initiative des agents.

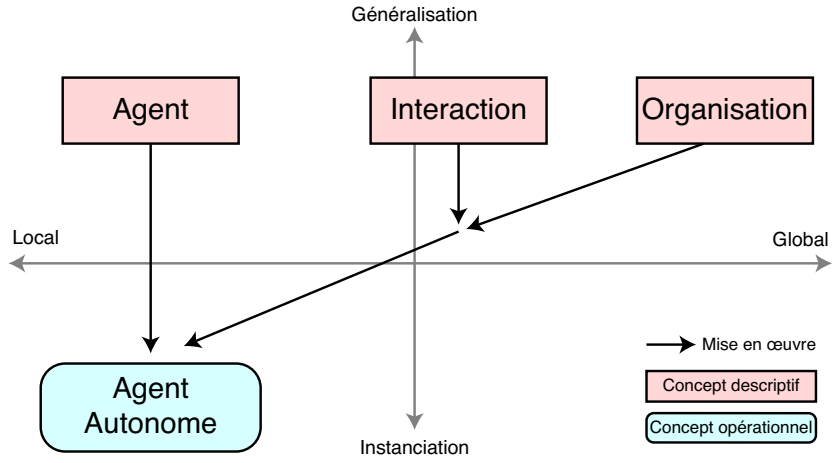


FIG. 3.1 – Construction de systèmes-multi-agents, d'après [Foisel, 1998]

Modèles d'organisations

De ce fait, il est naturel de passer d'une étude en soi des mécanismes d'interaction à des structurations de plus haut niveau, même si une des premières questions qui se pose est de savoir *à quel niveau* l'effectuer. Ainsi, dès [Gasser, 1992], la question de la *délimitation* de l'espace propre d'un agent est posée, avec en particulier l'étude de :

- La notion d'encapsulation : comment définir la nature d'un agent par rapport à son contexte (en raisonnant en fonction du problème de la clôture des bases de connaissances)
- La sémantique de l'aggrégation : comment faire le lien entre l'agent et ses modèles (la définition de l'identité, des connaissances, les savoirs-faire ou services, la notion d'engagement à utiliser)

Comment envisager l'organisation hors du modèle d'agent ? On peut la voir comme une limitation de l'autonomie de l'agent : la norme sociale à laquelle se soumet l'entité est une *contrainte extérieure à l'agent qui indique avec qui interagir, dans quel contexte et selon quel mode*. L'organisation est bien ici un principe actif : *lorsqu'un agent est amené à interagir, l'organisation lui indique généralement avec quel autre agent il doit le faire, lui évitant ainsi de chercher et décider comme il le ferait s'il utilisait, par exemple, un protocole tel que le contract net"*. Ceci étant posé dans le but de limiter le nombre de conflits et augmenter l'efficacité globale de la société, l'autonomie perdue par l'agent est considérée comme un moindre mal. L'organisation n'est donc pas qu'une structure (statique) fondant l'activité mais bien un processus (dynamique) qui permet d'y aboutir, en guidant les agents (à la conception ou l'exécution) vers un état désiré du système.

Une des structures organisationnelles les plus connues en I.A.D. est le réseau contractuel défini par [Smith, 1980], imaginé pour résoudre le problème de l'allocation de tâches entre

agents. Le schéma d'interaction dans cette structure est bien connu : un agent gestionnaire envoie un appel d'offre avec des critères de sélection ; des agents contractant et répondant aux critères évaluent l'offre et proposent leurs services. Le gestionnaire fait alors son choix et répond aux candidatures. Un point parfois oublié de ce modèle est la non-désignation préalable des agents dans les rôles gestionnaires ou contractants : c'est dans un contexte donné que s'établiront ces relations. Cette structure pouvant se déployer de multiples fois simultanément, le nombre de contrats gérés simultanément n'est évidemment pas borné.

Une mise en application de ces principes montrant clairement la séparation entre organisation et agents apparaît dans [Werner, 1992] : dans cette application de marché, la structure de l'organisation, une hiérachie, est établie préalablement puis mise en place dans les agents.

La mise en oeuvre est directe :

- Chaque agent a un jeu de connaissances minimales pour agir socialement : nom du groupe d'agents avec qui coopérer, préférences sociales.
- La communication est uniquement point-à-point et structurée via les connaissances sociales.
- Les structures d'organisations n'existent *ensuite* qu'en tant que croyances partagées par les agents impliqués.

Le groupe est vu alors comme un "Special Interest Group", une communauté d'intérêt : des agents, un nom de groupe et des restrictions. Le responsable du groupe est alors celui qui définit les contraintes et valeurs du groupe, il informe les agents candidats à l'admission du nom du groupe et des restrictions ayant cours. Les agents candidats doivent alors connaître - outre l'identification du groupe en question - les protocoles à observer, les services fournis par le groupe. Il est possible de faire de la diffusion de message au sein du groupe, par exemple un marché avec acheteur et tous les fournisseurs vus comme un seul groupe. Cette notion de service est importante puisqu'elle permet d'établir une *utilité* sociale : quels sont les avantages pour un agent à venir dans le groupe ?

Dans [Prasad et al., 1995] également, l'organisation des agents est construite de façon à fournir un meta-niveau d'expression à une résolution de problèmes. On se place dans le cas où une application où l'on veut rechercher une solution par composition progressive de solutions partielles. Chaque agent est potentiellement partie prenante de plusieurs activités de construction d'une solution, en fonction de ses compétences intrinsèques et de ses connaissances concrètes à ce moment de la résolution. Des propositions émises par un agent sont raffinées, étendues et critiquées avant d'être adoptées. Pour structurer cette activité, des rôles "organisationnels" ont été définis : l'`initiator`, l'`extender` ou le `critic` d'une solution, ainsi que des opérateurs spécialisés. Par exemple l'initiateur d'un problème peut, outre émettre la proposition de base, relâcher certaines contraintes sur la solution au fur et à mesure du travail collectif. L'objectif est donc d'identifier les agents les plus à mêmes de participer efficacement à certaines catégories de problèmes. Dans ce cas précis, un mécanisme

d'apprentissage permet cette identification.

Ce qui nous intéresse particulièrement dans cette approche est avant tout l'idée de *structuration* d'un modèle applicatif par l'expression d'un modèle social spécialisé. Les auteurs concluent sur l'observation de meilleures performances du système par rapport à un précédent qui ne prenait pas en compte cet aspect collectif explicite.

Cette structuration peut aussi s'exprimer par l'explicitation des relations ou des dépendances entre agents. Ainsi, [Bouzouba et Moulin, 1997] étudient l'impact des relations sociales sur les modes d'interactions. Il s'agit par exemple de conversations entre supérieur hiérachique et employé ou entre père et fils : des situations qui sont structurées et influencées par les relations de dominance. L'idée est ici de reporter au niveau agent un fait mis en évidence par les sociologues du contexte social lors de l'étude des interactions en conversation. L'influence de ce contexte est très importante dans la conversation mais souvent *implicite* (sauf en cas de rupture de la règle sociale, où il faut réaffirmer la relation d'autorité).

Appliqué aux systèmes multi-agents, ce travail a permis de rendre compte de l'implicite (en définissant structuration ad hoc) et de l'importance des relations sociales (en tant que telles). Par contre, l'approche est focalisée sur l'interaction conversationnelle et non étendue à d'autres modes (par exemple dans l'action). Il s'agit de faire la part des relations sociales dans les conversations, en remarquant que les informations à caractère social sont très souvent non verbalisées, et donc en les représentant par des états mentaux de type social qui figurent explicitement leurs connaissances des relations et des conventions sociales.

Par exemple, on décrit comme suit une relation interpersonnale donnée (`etat1`) comme un jeu de couples de rôles potentiels, que l'on définit ensuite par la liste des schémas de conversations qui peuvent s'y rattacher

```
REL-INTERP (etat1, père-fils,
            [(père-normal, fils-normal), (père-faible, fils-défiant),
             (père-autoritaire, fils-soumis)])
ROLE(etat1, père-normal,
           [sch-conv "requête avec position d'autorité,
            sch-conv "refus d'un défi social, rest-sch-conv])
ROLE(etat1, fils-normal,
           [sch-conv "requête polie", rest-sch-conv])
ROLE(etat1, fils-défiant,
           [sch-conv "refus d'un but à cause d'une position sociale
            anormale", rest-sch-conv])
```

Le but ici est d'expliciter les situations de discours par le contexte social (à travers l'attribution des rôles) et de pouvoir identifier et adapter les schémas d'interactions dans les cas où la position sociale sort de la norme établie.

Cela peut aboutir d'ailleurs à définir une typologie des organisations, comme le fait [Ouzrout et al., 1996] qui en dégage plusieurs styles de structures, comme :

La structure organisationnelle hiérarchique qui traduit la hiérarchie de responsabilité modelée dans le système à partir de la structure sociale humaine.

La structure organisationnelle de dépendance qui définit les protocoles de communication et synchronisation pour constituer la fondation de la coopération potentielle.

La structure organisationnelle en "effet de groupe" qui correspond à l'idée moderne d'entreprise en réseau : des objectifs définis au niveau d'un groupe autonome et éventuellement transversal. A noter que dans ce travail précis, le groupe est en fait agentifié dans un meta-agent (dont la sémantique exacte n'est pas précisée).

Même non formalisés, on voit poindre dans le discours des auteurs des traits marquants que nous retrouverons dans nos abstractions, tel que :

> *Ces structures organisationnelles superposées vont faciliter la spécification des protocoles de coordination et de synchronisation des agents cognitifs.*

Normes et obligations

MOISE [Hannouns et al., 1999] est un autre exemple de travail focalisé sur l'organisation. Ici, l'organisation est avant tout *normative* : l'organisation agit comme un système de règles qui contraindra l'activité des agents, ou plus exactement leurs possibilités d'actions individuelles. L'expression des règles est faite par la notion de rôle, qui permet de définir un des comportements *attendus* d'un agent dans sa structure sociale. Le groupe se définit plus simplement comme un agrégat de rôles, c'est à dire un jeu de règles cohérentes. Cette cohérence est assurée par des liens spécifiés entre rôles, qui permettent d'exprimer les communications possibles, les accointances attendues ou les relations d'autorité entre rôles.

L'expression opérationnelle du modèle n'est néanmoins pas générique : elle est volontairement restreinte aux agents de types BDI. Ceci permet de définir les rôles dans des concepts manipulables directement dans les architectures internes (buts, plans à suivre, actions et ressources utilisables, ...). Dans le même esprit, les liens organisationnels établis entre rôles permettent d'aboutir à une description opérationnelle des protocoles d'interactions à utiliser, ou des priorités à favoriser entre buts.

Notons pour finir que le modèle permet une séparation bien définie entre modèle d'organisation (appelé *structure organisationnelle*) et organisation effective (notée *entité organisationnelle*), comme le montre la figure 3.2. Cela offre une facilité de représentation du point de vue conception, même si le contrôle de la dynamique de l'organisation ou bien la sémantique exacte d'une instance de rôle ou de groupe n'est pas définie.

Toujours dans ce cadre, on peut évoquer le travail de [Dignum et al., 2000], qui présente comment intégrer un raisonnement social dans un modèle d'agent BDI. Il s'agit ici d'intégrer explicitement normes et obligations sociales dans les influences pesant sur le comportement d'un agent. Cette part de l'influence sociale se justifie par la nature même des agents BDI.

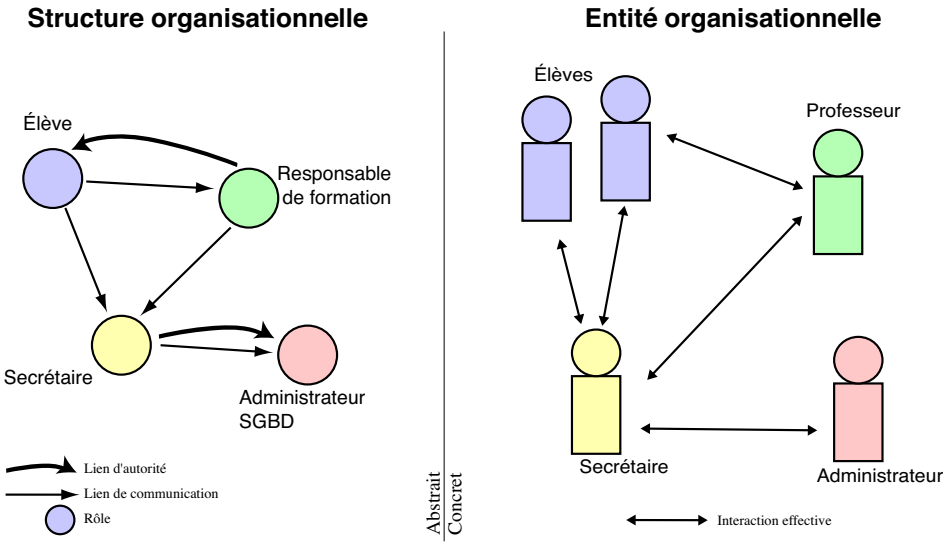


FIG. 3.2 – Un exemple de modèle d'organisation, d'après [Hannouns et al., 1999]

Les auteurs font en effet remarquer qu'une norme sociale n'est ni représentable comme un filtre sur le comportement de l'agent (car sinon l'obéissance à la norme serait absolue et l'agent ne pourrait en avoir conscience), ni comme une intention directe (puisqu'il est très fréquent de devoir composer avec plusieurs obligations sociales antagonistes). En pratique, normes et obligations sont représentées formellement par une logique déontique dyadique et une mesure d'utilité sociale permet d'établir une hiérarchisation entre normes à observer, et d'aboutir à une intention sociale exprimée.

Il est intéressant de noter que, même si dans ce cas la structure sociale est intégrée *préalablement* au coeur de l'architecture de l'agent, la justification du modèle est externe : *Obligations and norms are an important tool to "glue" autonomous agents together in a multi-agent system*.

[Fox, 1981] a été un des pionnier de ces approches structurantes, en envisageant un empilement de structures sociales, lesquelles étant contrôlées par des analogies comme l'emploi ou le marché. La structuration reste principalement hiérarchique et met l'accent sur la structure de marché comme cadre intégrateur. Ces travaux ont eu, entre autres, le mérite d'intégrer au coeur des préoccupations la question du choix de la structure d'organisation et, en corollaire, la nécessité d'évaluer les performances d'une organisation.

[Singh, 1991] remarque que souvent les théories des systèmes multi-agents ignorent la *structure interne* du groupe d'agents, en supposant une harmonie, une homogénéité d'action, ou en faisant l'hypothèse d'un monde parfaitement coopératif. Il propose de voir un groupe comme un simple agent monolithique non structuré, tout en supposant que le savoir-faire d'un groupe d'agents est du même type que le savoir-faire des individus. La seule différence

est qu'un groupe coopératif aura "plus" de savoir-faire qu'un de ses éléments. La définition est récursive et en conséquence, même si les groupes semblent homogènes, ils ne le sont pas en interne. L'opérationalisation du modèle associe une logique des mondes possibles avec une construction dynamique des dépendances sur les conditions et actions possibles. La structure de groupe est alors identifiée comme $G = ((x_1, ..., x_n)Is, Ir)$, un jeu de restrictions sur les interactions stratégiques (sociale et apprenables) et réactives. Une extension possible identifiée, mais non poursuivie de ce modèle est le lien aux rôles que les agents peuvent jouer dans un groupe.

Modèles hiérarchiques

On trouve aussi des modèles axés sur une forme particulière d'organisation, dont ils exploitent toutes les possibilités. On peut par exemple trouver un modèle caractéristique dans [Occello et Demazeau, 1997] pour le cas des systèmes multi-agents hiérarchiques. Le modèle donne des règles bien définies pour encapsuler un système multi-agents de niveau n en un agent de niveau $n + 1$:

1. Trouver une description fonctionnelle en agents pour le système multi-agent encapsulé.
2. Tenir compte de la nécessité pour un agent A_n^i membre de ce système multi-agents de communiquer avec des agents de niveau supérieur $(n + 1)$, c'est à dire de mettre en oeuvre les protocoles, de connaître les agents et de percevoir les environnements de niveau n et $n + 1$.

Cette approche n'est néanmoins pas véritablement généralisable, ne serait-ce que parce que le point 1 n'est pas toujours possible, et que le point 2 fait peser une contrainte structurale importante sur l'entité. Les auteurs précisent d'ailleurs bien que le mécanisme de décomposition fonctionnelle qu'ils illustrent n'est valide que si l'on se place dans un contexte de multi-agents situés (il faudrait sinon raisonner en tâches), et cette l'approche n'est valable qu'en tant que description *statique* d'un problème.

Un autre modèle hiérarchique apparait dans GEAMAS [Calderoni et al., 1997], mais avec une restriction importante : il s'agit là d'une modèle à nombre fixe de niveaux, avec trois échelles de représentation. Chaque niveau incorpore des aspects fonctionnels (par le rôle joué par chaque entité), organisationnels (par les schémas d'interactions) et individuels (via l'organisation des connaissances et des comportements). Ce modèle est plus adapté à un cadre de modélisation-expérimentation par son attribution prédéfinie des différents niveaux (éléments atomiques, couche d'intégration, vue système). Il se distingue néanmoins par son approche : structurer fortement l'application globale et les plans de communication (intra ou inter-niveaux), mais dans le but de mettre en place une structure favorisant l'auto-organisation et une multiplicité de points de vue pour l'analyse.

Les modèles holoniques sont une autre manière d'envisager des modèles fortement structurés et hiérarchisables. Un holon est un agent qui peut être fait d'holons plus petits, et qui peuvent se joindre à d'autres holons pour former des holons de niveau plus élevé. Le modèle holonique permet donc explicitement aux "groupes" d'agents d'être des agents à part entière dans des groupes de niveau supérieur. Le treillis résultant de holons est alors appelé une holarchie et forme la métaphore organisationnelle de base dans les systèmes holoniques.

Le modèle holonique a pour principal attrait la facilité de représentation de systèmes d'aggrégations principalement hiérarchiques très courants dans les systèmes réels, et qui permettent de garder une vue unifiée malgré une diversité d'échelles spatiales ou temporelles. Par exemple, [Parunak et Odell, 2001] citent une usine automobile actuelle rassemblant quelques centaines de milliers d'outils spécialisés (dont chacun va être un candidat naturel à l'agentification) et regroupée en une douzaine de lignes de production. Les ingénieurs peuvent définir, construire et gérer ce type de systèmes en focalisant leur attention sur tel outil, telle ligne ou telle ligne de production selon le problème posé, et en voyant à un moment donné une entité de haut niveau comme une agrégation abstraite d'éléments qui ne rentrent pas directement en ligne de compte.

L'intérêt de ce point de vue est de pouvoir simultanément analyser un système en se concentrant sur certains agents $a_1, a_2, ..., a_n$ que l'on supposera atomiques (agents humains, systèmes informatiques préexistants, agents logiciels), mais en traitant dans le même temps d'autres groupes comme des boîtes noires, et en conservant ce point de vue tant que l'on peut ignorer la structure interne des membres des groupes. A ce moment, on pourra revenir vers une analyse en groupe et rôles de ces sous-systèmes. Si l'on se restreint à l'étude des sociétés humaines, il est plus difficile de dissocier la notion d'organisation de celle de tâche ou d'échange d'information. [Adam et al., 1999]

En résumé, différentes situations demandent différents styles d'organisation dans les systèmes multi-agents. Mais comme ladite situation est susceptible d'évolution, il faut aussi être capable de remodeler la structure sociale.

3.2.3 Processus d'organisation

Réorganisation

Pourrait-on établir une organisation idéale pour un problème ? C'est peu probable : l'environnement, le contexte des agents n'est jamais complètement maîtrisé, et l'adaptation est obligatoire. De fait, la dynamique des organisations est quasiment toujours évoquée dans les modèles sociaux d'agents. Ce qui varie par contre beaucoup plus, c'est l'importance des changements que l'on autorise dans une organisation : certains travaux présentent même cette capacité de restructuration comme essentielle, à travers l'émergence de structures sociales ou des réorganisations permanentes.

Dynamique et émergence

La structure sociale influence les comportements de chaque agent vers un possible "équilibre social". La stabilité ou simplement la convergence vers cet équilibre peut varier selon les modèles, entre les sociétés figées fixées par le concepteur, et les approches entièrement émergentistes. Il ne faudrait donc pas voir une dichotomie absolue dans ces approches : une action individu-centrée peut être une une action influencée par le contexte social (voir figure 3.3. Il suffit pour cela que la structure sociale impose des contraintes (les institutions ou normes sociales) sur les agents. Cette relation double entre interaction et structure sociale a notamment été modélisée par [Ossowski, 1999]. L'auteur remarque d'ailleurs avec justesse que cette liaison est fonction de la complexité de l'individu mais aussi de la société :

> *The influence of socialisation on the self-interested behaviour of individuals decreases with the complexity of society. By contrast, within simpler societies the socially conditioned sphere of individual consciousness is predominant.*

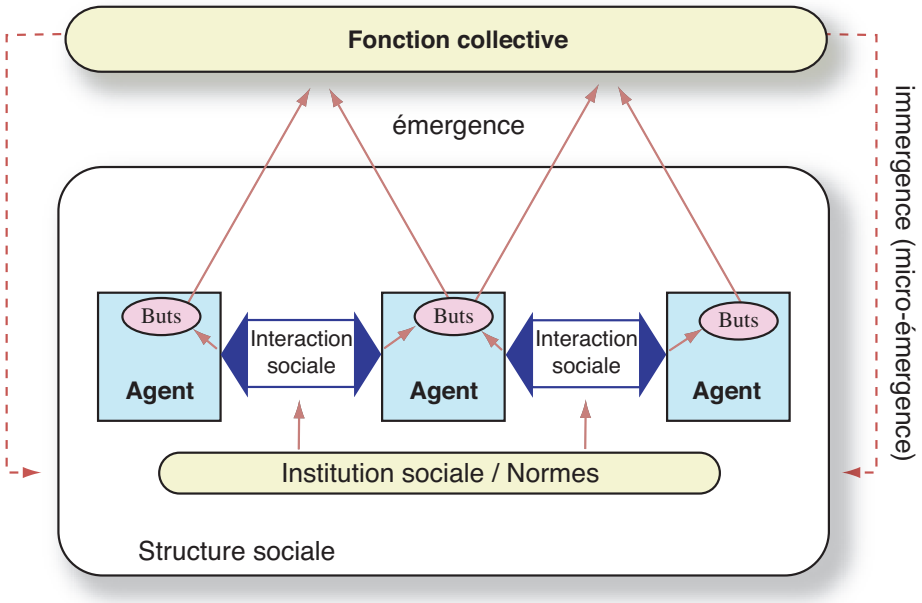


FIG. 3.3 – Coordination émergente, d'après [Ossowski, 1999]

Plus qu'une simple coordination des comportements des agents permettant d'identifier une "fonction" pour l'ensemble de l'organisation, cela peut également permettre également un ajustement (par rétroaction) les comportements individuels [Haynes et al., 1995], et par conséquent l'efficacité de la coopération globale.

On peut comparer cette approche aux mécanismes de réorganisation beaucoup plus radicaux définis par [Glance et Huberman, 1995], imaginés sur une métaphore de réorganisation

de matière, au sens *physique*.

Schémas d'organisation

Le modèle de "schémas d'organisation" de [Durand, 1996] est sans doute un des efforts les plus complets à ce niveau. Bien que définie initialement dans un cadre de simulation multi-agent (l'étude d'épizooties de fièvre aphteuse), le modèle proposé est en fait générique et ses conséquences dépassent le simple problème de la modélisation et mise en oeuvre de simulation multi-agent.

L'auteur remarque la complexité et la multitude de domaines à prendre en compte lorsqu'on envisage la modélisation de systèmes à large échelle. Se pose alors le problème de la synthèse des différents domaines de connaissance pour pouvoir mettre en oeuvre une entité de simulation suffisement complète (prenant en compte par exemple dans le cadre d'étude les points de vue économique, sanitaire, législatif, spatial, ...).

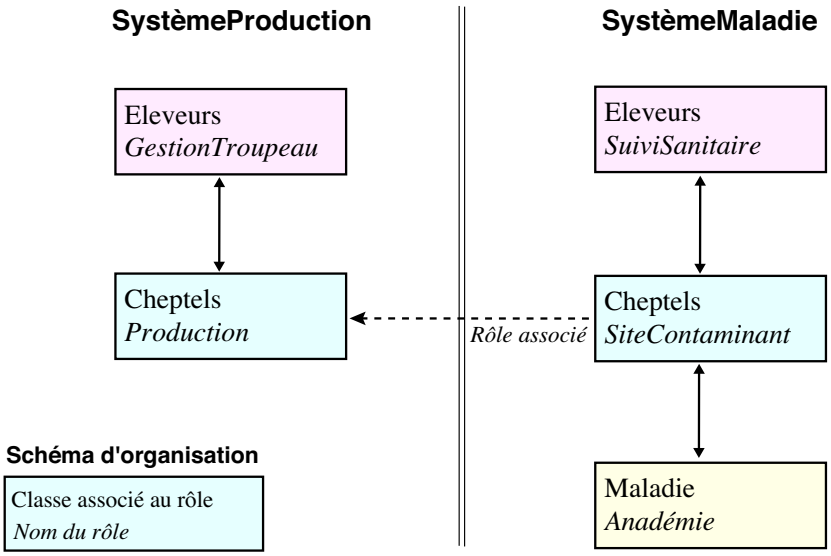


FIG. 3.4 – Un exemple de réseau de liens d'accointances exprimé par des schémas d'organisation, d'après [Durand, 1996]

Le point de départ de la proposition est la simple question du rattachement des activités aux entités : faut-il définir le comportement comme faisant partie intégrante de l'entité (ou plutôt de sa classe, puisque B. Durand utilise la modélisation objet comme base de départ). Cette possibilité est rapidement écartée comme trop statique (pas de possibilité d'acquérir un nouveau comportement) et pénalisant pour les besoins de conception et simulation incrémentale. L'autre approche envisagée est de rattacher les activités possibles aux entités concrètes elle-mêmes (et donc à leur instances), ce qui satisfait aux besoins d'adaptabilité mais complique singulièrement la conception à un niveau plus abstrait.

Le modèle finalement proposé *associe* (et non définit) les comportements et activités au sein des modèles d'entités (leur classes). Ces activités sont alors définies comme étant des rôles, et permettent d'agir à deux niveaux :

– En associant à chaque agent les tâches qu'il *sait* et *veut* faire.
– En associant à chaque définition d'agent les tâches que les instances *peuvent* faire.

Les schémas d'organisations sont alors définis comme des points de vue aggrégeables, représentant chacun un ensemble de rôles conçus pour s'exécuter d'une façon coordonnée. On est bien en présence d'une modélisation structurelle et fonctionnelle d'un système.

Néanmoins, l'accent mis sur la simulation fait que d'autres parties du modèles proposé sont nettement moins génériques. Par exemple le rôle est défini avant tout comme un comportement, via une classe, plutôt que comme un point d'interaction, et tout comportement est par principe rattaché à un rôle. Cela aboutit à une définition très complète du rôle (où les comportements peuvent être réactifs, proactifs) mais qui obligent parfois à transiger quand on veut définir des connaissances ou comportement de base au niveau de l'agent. Ainsi un comportement spécifique (appelé comportement fonctionnel) est défini que pour permettre l'exécution d'autres comportements selon l'activité du système. On est a ce niveau très proche d'un niveau implémentatoire du système (et particulièrement de l'architecture de simulation) ce qui pose problème si l'on veut adapter le modèle à d'autres domaines.

Un autre aspect notable de la proposition est le modèle d'exécution réflexif d'un système de simulation conçu selon la proposition. Les agents, schémas d'organisation, système ou rôles sont réifiés et décris dans le même modèle. Par exemple, le rôle "agent" est capable de réagir aux perturbations ou notifications pour activer des comportements, d'établir un lien d'accointance vers le schéma d'organisation concerné. De même, le rôle "simulation" contrôle les transitions et l'état d'un cycle d'exécution. Même si l'action au meta-niveau n'est pas possible de façon générique, le contrôle de l'exécution d'un système simulé est tout à fait intégré au modèle.

3.3 Approches opératoires

Comme nous l'avons vu, de nombreux modèles sont proposés pour faciliter la modélisation d'une application multi-agent par une approche sociale. Cette inspiration a également un impact dans un contexte bien précis : les modèles d'implémentation ou d'architectures de systèmes. La problématique n'est pas si éloignée de ce que l'on a vu jusque là : adaptation, souplesse, puissance d'expression ou évolutivité, mais l'accent y est clairement opératoire.

On notera aussi que l'approche - plus pragmatique - s'échappe un peu du cadre des systèmes à agents. En particulier, les modèles à objets ont suscité toute une gamme de proposition d'extension par des rôles, tant au niveau modélisation qu'implémentation.

3.3.1 L'organisation multi-agent comme implémentation

Commençons par l'examen de modèles plutôt ciblés agents. L'utilisation de structures sociales comme moyen d'implémentation multi-agent n'est pas une idée neuve. En fait, il est même frappant de voir que la majorité des contributions sur ce thème datent des débuts des systèmes à agents. C'est le cas en particulier de plusieurs système d'IAD/SMA précurseurs, comme Pandora [Maruichi, 1989], ACTORS/ORGS [Hewitt et Inman, 1991], la réflexivité organisationnelle de [Giroux, 1993], ou encore MACE.

En effet, on y retrouve souvent ces principes de structuration d'agents en organisations, d'action sur des structures sociales, en tant qu'outils de base des système.

Comment expliquer cet intérêt précoce pour les système multi-agents en tant que technique d'implémentation ? Cela vient peut-être du fait que le débat initial sur la définition exacte et la comparaison des notions d'acteurs, agents, ou objets concurrent n'était pas tout à fait clos, et que les frontières entre ces différents niveaux d'expression étaient plus diffuses que dans les travaux actuels. L'étude de ces modèles n'est néanmoins pas sans intérêt, vu les points de concours avec le thème actuel de l'agent comme architecture logicielle.

Evidemment, la gamme couverte par ces modèles est assez vaste : de modèles très expressifs d'organisations multi-agent comme la réflexivité organisationnelle à de simples questions de partage de ressources ou de coordination bas-niveau. C'est par exemple ce que faisait ABCL/R2 [Matsuoka et al., 1991] pour exprimer l'ordonnancement (et plus pour structurer le méta-niveau que le dénoté) ou encore comme dans ORGS, où l'objectif premier est de faciliter la composition de systèmes d'acteurs.

Le groupe comme moyen de communication

Un modèle fréquent d'utilisation de l'organisation multi-agent comme approche opératoire est la fédération des moyens de communication. Comme une application multi-agent implique souvent d'interagir avec plusieurs agents simultanément, en se basant sur leur type, fonction, ou objectif, pourquoi ne pas fédérer certaines entités pour factoriser une partie des interactions.

Ce principe de structuration par le groupe se trouve par exemple illustré très tôt dans PANDORA-II [Maruichi, 1989] [Maruichi et al., 1990]. On trouve déjà dans ce système une distinction claire entre objet actif et agent, définie par la présence ou non d'un mécanisme d'*interprétation locale* des messages. L'agent est alors défini comme un acteur qui possède son propre interprète de message. À partir de cela, le groupe (ou environnement) n'est qu'un ensemble d'agents coopérants et qui met en commun les fonctions de communications du groupe. C'est même cette entité qui fera l'aiguillage de messages, via des opérations primitives comme `(send-all environment-name message &key :except me)` ou `(send-to environment-name agent-name message)`.

Ce mécanisme de communication permet la définition d'un système complexe comme une collection de groupes d'agents, une organisation. En fait, les agents travaillant dans un même environnement sont considérés automatiquement comme formant un groupe, avec surtout la possibilité pour un agent d'appartenir à plus d'un groupe. L'environnement est vu simplement comme le medium de communication, il est en fait confondu avec la notion de groupe. Notons également que le modèle a également été motivé par le souci de fonctionner dans un contexte de traitement distribué, avec une implémentation en KCL [Yuasa, 1990].

Le point distinctif de ces travaux réside dans cette approche de la communication qui amène le concepteur à penser à la structuration de son système comme une organisation, dynamique et à qui l'on attribuera un trait fonctionnel :

- Un agent n'a pas à spécifier les noms de toutes les cibles. La communication est avant tout définie par un couple *(environnement, message)* bien qu'il soit également possible d'intervenir en point à point par *(environnement, message, agent-cible)*.
- Le groupe ou environnement est une entité dynamique qui peut être instanciée, détruite, qu'on peut rejoindre ou quitter et qui peut être interrogée (pour obtenir le nombre ou le nom des agents).

On est donc bien là en présence de structures organisationnelle définies comme des sous-systèmes clos, cohérents individuellement, peuvant être restructurées par et pour les agents.

On pourra retrouver une mise en oeuvre de ce même principe de structuration plus tard, dans le contexte des agents mobiles [Baumann et Radouniklis, 1997]. Là, le groupe est défini comme une collection d'agents travaillant ensemble sur une tâche commune : à partir de cette notion de groupe, des rôles fonctionnels sont identifiés et attribués. Dans certains cas, cela peut être implicite, par exemple si l'on est simplement "membre" du groupe créé par un "initiateur" de la tâche.

Cette identification fonctionnelle permet alors de communiquer de façon quasiment anonyme, puisqu'il n'est pas besoin de connaître directement les accointances hors de leur rôle. Toujours selon le même principe, un coordinateur identifié définira les dépendances entre le groupes et l'extérieur. Comme chaque groupe est équivalent à une tâche, la structure sociale est en fait une manière d'exprimer implicitement un objectif global pour l'organisation.

C'est d'ailleurs sans doute l'écueil de ce modèle. Cela impose de concevoir l'organisation à deux niveaux complètement distincts : une activité multi-agent "classique" au niveau du groupe (identifiée fonctionnellement en terme de rôles) et des chaînes de tâches au niveau de l'organisation. Cette dichotomie oblige d'ailleurs à composer avec deux mécanismes de communications distincts, inter-groupe et intra-groupe, ce qui empêche toute unification du modèle ou structures hiérachiques.

La réflexivité organisationnelle

Un des travaux les plus complets à avoir pris l'organisation comme principe d'implémentation de systèmes est sans doute la proposition de notion de réflexivité organisationnelle de [Giroux, 1993]. L'idée est ici d'avoir une approche organisationnelle de la réflexivité (exprimée en termes d'acteurs plutôt qu'agents), en procédant par trois niveaux :

- L'acteur, que l'on exprimera comme une organisation d'acteurs.
- le méta-acteur, décrivant un aspect de son comportement opératoire : traitement des message, comportement somatique.
- Le système global : interactions avec le milieu, comportement social, situation dans le monde. Une organisation est la réification d'un ecosystème du point de vue des interactions entre les entités qui y évoluent.

L'auteur résumait tout cela par l'exemple suivant, emprunté à [Minsky, 1985] :

> *Un dîner entre deux personnes au restaurant est modélisé par deux écosystèmes : deux sociétés de l'esprit en interaction dans le contexte du restaurant, vues comme un écosystème de plus haut niveau.*

Du modèle sont définis les niveaux de l'acteur, du meta-acteur qui en assure le comportement implantatoire (en particulier son interprétation des messages et des interactions), et de la société, vue comme le meta-meta-acteur. Cela revient à dire que l'agent est réifié par une organisation qui prend en charge le traitement des messages. En tant que membre d'un système, la réification de l'agent a lieu à travers la réification du système dont il fait partie.

De façon un peu paradoxale, on pourrait reprocher à ce modèle d'être à la fois trop et pas assez radical dans sa démarche. Trop radical d'abord, parce que l'expression sous forme d'organisation de tous les agents individuels d'une société pose de nombreux problème de contrôle sur le système et de lourdeur de la conception. Il s'agit bien là d'avoir un modèle social pour caractériser une entité individuelle, avant d'envisager le niveau supérieur qui représente la société dans son ensemble. On peut évoquer d'autres systèmes ayant essayé d'architecturer un monde multi-agents sur des principes similaires, comme PALADIN [Ferber, 1989] : la maîtrise conceptuelle et opérationnelle d'un système réflexif à autant de niveaux touche vite à l'ingérable.

A l'inverse, on peut regretter que la réflexivité organisationnelle telle qu'elle est définie ici ne traite vraiment que l'implémentation de l'entité (réactions aux messages, comportements, ...) et ne s'attaque pas aux structures sociales en général ou à la dynamique de la société.

3.3.2 Objets et modèles de rôles

En fait, ces approches sociales ne sont pas l'apanage des systèmes multi-agents. Identifier fonctionnellement, structurer des systèmes sont des besoins qui en dépassent largement

le cadre : assez logiquement, on en retrouve les principes dans d'autres modèles, principalement les bases de données et les modèles objets.

Mais la perspective n'est pas exactement la même. Dans un contexte objet, une distinction est à faire entre la notion de classe, définissant les capacités d'objets *individuels* et isolables, et le rôle, rattaché à des caractéristiques *partagées* de position ou de responsabilité d'objets dans un système donné. Contrairement à ce que l'on a pu voir dans les approches agents, l'*interprétation* d'un rôle est très peu ambigüe : on veut avant tout conserver les bonnes propriétés de typage dans un système.

> *Le niveau agent correspond à une modélisation organisationnelle au-dessus des objets, où ceux-ci sont agrégés pour former des entités à la dynamique variable [Cardon, 1999].*

Justification

Mais pourquoi faire apparaître ce genre de structuration ? Si l'on part du principe qu'établir une architecture, c'est à la fois concevoir et construire une structure (à partir d'entités et de modèles élémentaires) et répartir les fonctionnalités du système entre les différents composants [Jacobson et Nowack, 1999], on se rend compte que l'on retrouve nos préoccupations initiales : exprimer aspects fonctionnels et structurels. La tâche n'est pas aisée, comme le souligne [Deutsch, 1989] :

> *Interface design and functional factoring consitue the key intellectual content of software and are far more difficult to create or re-create than code.*

Comment faire le lien entre les modèles sociaux et les architectures plus classiques ? Revenons à la base, aux besoins qu'expriment [Bäumer et al., 1996] au sujet des frameworks de taille conséquente :

Structure : l'interface d'un ensemble d'entités de base ainsi que les structures statiques ou dynamiques qui les mettent en relation dans une composition.

Fonction : l'association fonctionnelle entre le problème et la structure, par identification pure ou raffinement de définitions initiales.

Abstraction : l'identification (et le nommage) d'une composition d'éléments ayant une certaine structure et une certaine fonctionalité.

Réutilisation : la capacité à être appliqué plusieurs fois en des contextes différents.

Le problème vient en fait de la liaison à établir entre la conceptualisation (par l'étude du domaine) et l'implémentation (par la construction de l'architecture). Les deux aspects sont la plupart du temps exprimables simultanément d'un point de vue opératoire. Imaginons naïvement que l'on puisse simplement hériter d'une classe "de domaine" pour former une classe "d'architecture". L'héritage aura pour effet que deux instances de différentes sous-classes ne sont pas identiques *même si elles le sont du point de vue conceptuel*. Une solution

consiste à appeler le concept de rôle à la rescousse pour permettre à un objet de s'intégrer dans différentes couches applicatives. Le rôle est alors assimilable à un point de vue (spécifique au client) sur un objet jouant ce rôle.

On trouvera une bonne synthèse des motivations justifiant des modèles à base de rôle dans [Kendall, 1999]. Ce travail a également l'intérêt d'être à mi-chemin entre approches objet et agents. Nous en reprendrons ici les points principaux.

On peut finalement voir un modèle de rôle comme l'identification d'un archétype, une structure récurrente d'objets à décrire comme un ensemble de rôles en interaction. En général, les modèles de rôles cherchent avant tout à capturer la manière d'interagir que vont avoir des objets *concrets*. A ce titre, ils complètent élégamment la vue abstraite et centrée sur l'élément, de la vue en classes. Les rôles vont alors être dynamiquement assignés à des objets dans une application. On peut ramener la notion de rôle à un "service" spécialisé que remplira l'entité (qu'on pourrait ensuite associer à l'idée de comportement pour un agent ou de type, pour un objet). Le rôle permet donc de voir le système d'une façon plus globale : un rôle a des collaborateurs, qui seront les autres rôles avec lesquels il interagit.

En résumé, quels sont les points caractéristiques des modèles de rôles ?

- les modèles de rôle mettent l'action sur l'interaction. Les classes donnent les capacités des objets individuels, mais le rôle s'occupe plus de la position et des responsabilités d'un objet dans une structure globale ou dans le système. Les modèle de rôles sont orthogonaux et complémentaires aux classes
- Les rôles permettent de décrire les systèmes ou sous-systèmes en termes de patterns d'interaction, ce qui offre un nouveau niveau d'abstraction à mi-chemin entre classe et instance.
- Les modèles de rôles peuvent être dynamiques (prise et abandon de rôle, évolution, transfert, ...)
- Le rôle lui même peut être, selon les modèles, traité au même niveau que les classes et objets, avec la possibilté de les instancier, généraliser, spécialiser ou agréger.

A noter que tout cela permet également de penser à la réutilisation en terme de rôles, par exemple pour élaborer de nouvelles définitions à partir de l'existant : par dérivation (avec une entité jouant un rôle dérivé devant pouvoir jouer aussi les rôles de base), par aggrégation (où un rôle peut être l'aggrégation d'autre rôles, masqués ou non), par relations (en posant des dépendances ou des équivalences entre différents rôles).

Rôles et patterns d'analyses

Une autre manière d'approcher la notion de rôle est de la penser comme principe d'architecture. Une étude importante de ce point de vue est celle de M. Fowler, dans [Fowler, 1996]. La notion de rôle est étudiée en tant que *technique d'analyse* pour bâtir une architecture objet (et non en tant que technique d'implémentation).

Le problème initial est de modéliser un groupe d'objets ayant un certain comportement commun. Ils n'ont pas forcément tous exactement le même comportement ou la même fonctionnalité mais l'on arrive à identifier des traits partagés. Au premier coup d'oeil, cela peut passer pour un cas classique d'héritage, mais des complications surviennent rapidement sur des cas réels. On peut par exemple avoir des objets ayant plus d'un seul comportement, ou bien qui acquièrent et perdent dynamiquement ces comportements.

Le concept de rôle est alors avancé par l'auteur comme un moyen de résoudre ces cas ambigüs, et plusieurs techniques de prise en compte et représentation du rôle dans l'analyse sont présentées.

Une première approche simple - et peu satisfaisante - est d'associer le(s) rôle(s) à un seul ou plusieurs types, et de ne pas séparer l'expression de l'objet cible de son ou ses rôles. On peut adopter ainsi les représentations suivantes :

Type de rôle unique. Si la majorité des objets ont le même comportement avec seulement quelques variations, on peut utiliser un seul type pour les gérer tous. L'ensemble des comportements et caractéristiques est fédéré (statiquement) dans un seul type complexe.

Types de rôles distincts. S'ils sont très différents et qu'il n'y a pas de comportement commun identifiable, on peut alors exprimer par des types distincts. L'inconvénient est que tout comportement commun sera séparé, et un même objet jouant plusieurs rôles sera délicat à mettre en oeuvre. L'analyse amène rapidement à trouver des solutions plus adaptables, comme le cas suivant.

Rôles par héritage. La plupart du temps, la manière la plus raisonnable de structurer l'application est de sous-typer les rôles pour décrire les différents comportements. Notons que cela ne veut pas forcément dire que l'on va utiliser un héritage pour l'implémenter. Ce sont bien des patterns *d'analyse*.

Là encore, l'écueil est la prise en compte de la dynamicité ou l'ajout de nouveaux comportements au système. Il faut bien voir que l'entité n'est pas *une sorte de* rôle au sens strict, il faudra donc utiliser des mécanismes internes (drapeau, délégué, objet d'état) pour gérer ou aiguiller les comportements. Il faut également pouvoir explorer ce niveau de typage pour y déterminer les rôles possibles, soit par des méthodes explicites (par exemple `estUnVendeur()` ou génériques (comme une opération `estUn("vendeur")`).

La solution la plus adaptable est donc de réifier le rôle en tant que tel. Là encore, plusieurs techniques sont possibles :

Rôles et délégueur. L'objet devient alors une coquille abritant des liens vers différentes instances (au sens objet) de rôles, qui peuvent alors être définis librement, et éventuellement par héritage. Changer de rôle, rajouter des comportements ne perturbe pas le

type de l'entité en elle-même. Comme précédemment, il faut pouvoir déterminer l'état des rôles tenus et éventuellement les manipuler, que ce soit explicitement ou non.

Relation explicite. Le rôle peut également être vu comme un lien entre plusieurs objets, pré-structurant ou contrôlant la collaboration entre objets.

Il est frappant que dans ces derniers cas (et en particulier dans celui du rôle comme relation), on s'approche de très près d'une expression agent. Même l'exemple utilisé dans le travail original ne déparerait pas dans un exemple multi-agents : des personnes collaborant au sein d'un département et structurées en managers, ingénieurs ou commerciaux.

Le tableau et la figure ci-après résument des patterns d'analyse possibles pour exprimer la notion de rôle selon M. Fowler.

Problème	**Solution**	**Pattern**
Comment représenter les différents rôles d'un objet ?	Combiner toutes les caractéristiques du rôle en un seul type	Type de rôle unique
	Traiter chaque rôle comme un type à part entière	Rôles typés indépendament
	Définir un sous-type pour chaque rôle et factoriser le comportement commun	Sous-types de rôles
	Définir un type central qui servira de point d'accès aux rôles (et donc aux comportements spécifiques)	Rôles réifiés
	Faire de chaque rôle une relation privilégiée avec un objet donné	Rôle réifié comme relation
Comment implémenter la généralisation ?	Utilisation d'un champ interne et sélectionner le comportement approprié dans la classe	Drapeau interne
	Mettre en place les aspects dynamiques dans une classe séparée et invisible, et déléguer les messages au besoin	Délégué masqué
	Créer un délégué caché pour chaque rôle, leur donner un super-type commun avec un comportement par défaut, et rendre visible le lien vers le super-type[5]	Objet d'état

[5]On retrouve d'ailleurs directement ce pattern dans la référence [Gamma et al., 1995]

Comment se référer à un rôle particulier d'un objet ?	Utiliser une représentation explicite du rôle par des méthodes spécifiques aux différents rôles possibles	Méthodes de rôles explicites
	Utiliser un mécanisme générique de typage par rôle, par des méthodes parametrées	Méthodes de rôles génériques

TAB. 3.1: Patterns d'analyse pour la représentation des rôles dans les modèles à objets, d'après [Fowler, 1996]

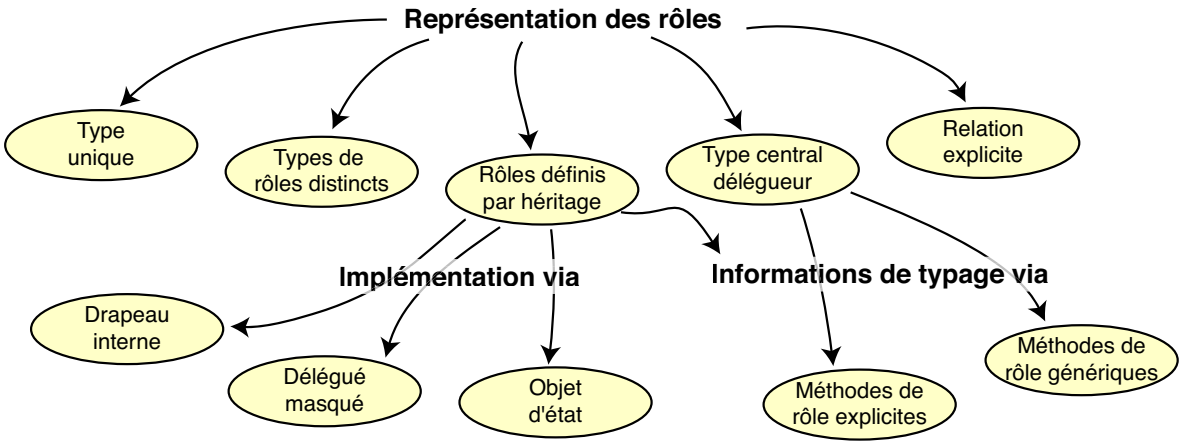


FIG. 3.5 – Relations entre patterns de rôles, d'après [Fowler, 1996]

On pourrait conclure sur l'examen de ce travail en rappelant les limites soulignées par l'auteur lui-même : il n'existe pas de moyen satisfaisant d'exprimer la dynamicité et l'adaptation du comportement d'un objet en se fondant sur ces principes hors la réification du rôle, ce qui revient alors à en faire un concept de premier niveau dans l'activité de design[6].

Rôles et modélisation

Des représentations comme les diagrammes de collaboration ou les cas d'usages d'un langage de modélisation graphique tel qu'UML [OMG, 1999b] ne sont finalement guère loin. Là encore, il s'agit d'abstraire une position dans une situation d'interaction, en l'examinant dans son contexte de collaboration avec d'autres entités, que ce soit dans un protocole figé (dans les séquences ou collaborations) ou dans une première étape de raffinement (avec les cas d'usages). Cet usage informel du rôle n'est d'ailleurs pas sans écueils. Le danger principal de cette manipulation est justement le fossé que se creuse entre l'usage informel du rôle et le besoin d'arriver à un modèle fort écartant précisement cette même notion.

[6]Même si son expression dans une *implémentation* objet par la suite le rend implicite.

On peut sans doute retracer l'origine du rôle dans les modèles UML aux diagrammes entité-relation, mais surtout dans des méthodes antérieures comme [Reenskaug et al., 1996] qui prennent le rôle comme base de modélisation en faisant les remarques suivantes :

- Raisonner en termes de rôles favorise la séparation des problèmes. Même si un objet peut jouer plus d'un rôle, le concepteur peut néanmoins les analyser indépendamment.
- Les rôles mettent l'accent sur la notion de *responsabilité* (par exemple les messages acceptés ou envoyés par un rôle), par opposition aux classes qui sont plus concernées par la *compétence* de l'objet (c'est à dire la réception et le comportement associé à un message).

L'influence de ces principes sur les descriptions d'interactions dans UML sont nettes (que ce soit dans les diagrammes de collaboration ou diagrammes de séquences). On voit d'ailleurs dans certains travaux récents sur UML une remise en cause de l'usage du rôle visant non pas à l'éradiquer des modèles pour faire disparaître cette difficulté de liaison, mais au contraire à lui donner une place plus apparente, avec une sémantique clarifiée. Ainsi, [Steimann, 2000] note que dans la version actuelle d'UML [OMG, 1999b], on voit apparaître trois définitions différentes du concept de rôle, vu comme :

- le point de rattachement d'une association (et donc statique),
- une position dans un diagramme de collaboration (et donc potentiellement dynamique),
- une classe de rattachement dynamique[7].

La première définition se justifie par le souci d'identifier chaque extrémité d'une association sans ambiguïté par un nom (de rôle) et l'utiliser comme tel par la suite. Dans la seconde définition, la contrainte est plus précise. En retournant à la source [OMG, 1999b], voyons comment sont définies ces notions :

> *While a classifier is a complete description of instances, a classifier role is a description of the features required in a particular collaboration, i.e. a classifier role is a projection of, or a view of, a classifier.*

Ainsi que :

> *As the association roles specify a particular usage of an association in a specific collaboration, all constraints expressed by the association ends are not necessarily required to be fulfilled in the specified usage.*

Ceci suggère le rôle comme une "spécification partielle" que l'entité doit respecter.

Dans UML, les diagrammes d'interaction sont situés en fait à deux niveaux : la spécification (les classifieurs) et le niveau des instances. La difficulté est que les interactions ou collaborations prennent place entre objets *concrets* et non entre rôles ou classes. On a forte-

[7] Ce qui est mentionné assez brièvement dans la spécification d'UML, plus en allusion à des modèles préexistants que pour en faire réellement usage.

ment l'impression que le niveau de la spécification n'est nécessaire que par la structuration qu'elle propose.

La spécification de l'interaction en elle-même est ambigüe par sa position. On doit être capable de distinguer plusieurs objets même s'ils ont un rôle identique et, plus important, si un même objet joue plusieurs rôles dans l'interaction.

> *In UML, interaction diagrams are offered at two levels : the specification (or classifier) and the instance level. However, interactions (and collaborations) always take place between actual objects, not their roles or classes, and it seems that the specification level is needed only insofar as interactions require or impose additional structure, structure that, because of its generality, should indeed be expressed on the classifier level.*

L'argument de F. Steimann est que le rôle n'est finalement pas une spécialité des collaborations ou interactions, mais un élément structurel primitif. Les rôles vont alors être des "ponts" entre les classifieurs instanciables et leurs associations et ce, que l'on soit dans le cadre de la définition d'une collaboration, ou bien hors du contexte de toute interaction. Prenons la définition proposée :

> *A role (more precisely, a role type) is a classifier like a class, only that it does not have instances of its own. Instead a role recruits its instances from those classifers that are declared compatible with the role, that is, that comply with the partial specification that makes the role.*

On voit se dégager ici un point important que nous retrouverons décliné dans notre propre travail : la relation complexe entre le rôle, agissant à la fois en tant que structuration d'une instance et partie d'une spécification. Cela induit un jeu de présence au niveau abstrait et opératoire. L'ajustement de la phrase entre "role" et "role type" est très révélateur. Notons également que rien n'est suggéré dans le travail comme conséquences au niveau de l'architecture concrète ou de l'implémentation, malgré l'importance du rôle, remontée ici quasiment au niveau de la classe ou de l'instance.

Un autre travail, [Wegmann et Genilloud, 2000], fait un parallèle intéressant entre les modèles de rôles et les cas d'utilisation. Les auteurs remarquent la similitude d'intention : aboutir à une compréhension du problème en se focalisant sur des collaborations identifiables entre rôles. Le souci est qu'UML définit les acteurs comme un ensemble cohérent de rôles que jouent les participants à un cas d'utilisation. Un acteur a un rôle pour chaque cas d'utilisation dans lequel il est impliqué. Les auteurs s'interrogent alors sur la sémantique exacte d'un acteur : si toutes les entités jouent un ensemble de rôles, qui y a-t'il de si unique à propos des acteurs ?

En fait, l'idée avancée dans ce travail est de clarifier les diagrammes de cas d'utilisation en renforçant la notion de rôle, dans deux buts :

– Pour unifier la notion de participant à une situation et pouvoir mêler acteurs, mais

aussi sous-systèmes, composants, voire classes au niveau implémentation, *via leurs différents rôles explicités et sans a priori sur leur modèle d'action*.
- Pour mieux réutiliser les cas d'utilisation dans différents contextes et pouvoir les combiner plus clairement.

Rôles et implémentation

Il est évidemment tentant de mettre en oeuvre cette notion (jusque là purement abstraite) en tant que mécanisme d'implémentation ou modèle formel opératoire.

Par exemple, [Wieringa et al., 1994] ont donné une définition formelle du rôle dans les modèles objets, via une logique modale. La mise en oeuvre du rôle dans ce modèle est de le définir comme un objet standard, mais avec la propriété supplémentaire qu'il peut être mis en relation avec d'autres objets par un mécanisme (nouveau) de prise de rôle.

La définition exacte est qu'il existe une *opération primitive* dans le modèle, $played_{by}$ qui établit l'association entre une instance du rôle et l'objet qui le joue. Un mécanisme de délégation masqué se charge alors de transmettre au besoin l'évaluation des opérations. Par contre, la notion de contrôle de la prise de rôle n'est définie qu'en filigrane, via l'arité de nombre d'un même rôle jouable par un objet et la possibilité de définir des ensembles de rôles en exclusion mutuelle.

Notons que le problème de la clôture apparaît (comment identifier une personne jouant le rôle d'un acheteur dans différents contextes ?), sans être pourtant résolue.

Une approche plus complète est celle de [Gottlob et al., 1996], qui élabore un modèle conceptuel et implémentatoire[8] de la notion de rôle, là encore dans une préoccupation d'évolution d'objets sur le long terme.
- Une hiérarchie de types de rôles, parallèle à la structure de classes.
- Des primitives d'actions, au niveau du typage en rôle (création d'un rôle, création d'instances de rôles), et sur l'entité (gain ou abandon, vérification de l'existence de rôles).
- Des comportements spécifiques aux rôles.
- Une notion de contexte (rôles "qualifiés" pour pouvoir exprimer le même rôle dans différentes situations).

Les auteurs mentionnent les difficultés de la détermination du rôle *de l'extérieur de l'objet* dans la hiérarchie (comment mettre en place le mécanisme de délégation quand plusieurs rôles définissent le même comportement ?).

La notion de "rôle qualifié" est sans doute la plus proche du groupe tel qu'on le trouve dans des modèles plus orientés agent. Il s'agit de distinguer le contexte de l'interprétation du rôle, pour permettre plusieurs instances simultanées d'un même rôle dans un même objet.

[8] Prototypée en SmallTalk

La notion de rôle dans les modèles à objets a aussi été particulièrement populaire dans le domaine des système de gestion de base de données objets. La motivation étant ici de tirer parti de la dynamique des rôles pour permettre une évolution sur le long terme de schémas de base de données sans risquer la moindre rupture dans le traitement ou l'interprétation. On pourra se référer à [Coulondre, 2000] pour une présentation détaillée des efforts en ce domaine.

Remarques

Ce bref aperçu nous conduit malgré tout à relever des différences majeures entre ces modèles de rôles "objet" et nos préoccupations, et ce dans trois domaines :

- Le rôle est pensé avant tout comme une intégration dans un système de typage de classes, hiérarchies, délégations. Nous avons vu que les besoins de notre problématique se situaient plus au niveau d'une identification fonctionnelle pour l'expression de situations d'interaction ou de coopération.
- Un problème délicat à gérer du point de vue opératoire est ici la mise en place concrète de la liaison vers les opérations associés aux rôles. Il s'agit ici de faire le lien entre l'objet, entité unique et identifiable, et ses rôles, qui définissent attributs et opérations spécifiques. La question essentielle est alors celle de la mécanique *externe à l'entité ou au rôle* de choix d'attribution d'un message à un certain rôle. Dans notre cas, nous nous situons au niveau d'un agent, définissant et interprétant sa position dans la structure sociale. La représentation et l'interprétation de ce lien entre entité et rôle est bel et bien *intrinsèque* au modèle d'agent choisi.
- La question du contexte d'interprétation du rôle est absente ou peu développée. En effet, on se situe dans un modèle global d'application sur lequel on a peu d'ambiguïté sur la nature de l'interaction entre entités : il s'agit d'opérations invoquées sur des instances. Dans notre cas, la situation est beaucoup plus floue puisque là encore, le modèle d'interaction n'est pas figé au départ. D'autre part, le haut niveau d'abstraction des échanges entre agents oblige à prendre en compte à part entière l'ontologie de référence de ces échanges, qui peut très bien être différente dans divers sous-parties d'un système multi-agent de taille conséquente.

3.4 Conclusion

A l'issue de ce parcours des approches à base de concepts "sociaux" dans les modèles informatiques multi-agents ou objets, on peut dégager deux grands axes pour analyser les présupposés de ces modèles. L'un concerne le point de vue humain, du concepteur ou de l'observateur, et l'autre celui de l'effet au niveau du modèle de l'entité informatique.

- La prise en compte ou non de la structure sociale comme préoccupation explicite de conception. Les notions d'organisation et autres concepts afférents sont avant tout *à l'usage du concepteur*, que cela soit ensuite mis en place explicitement ou non au niveau de l'entité.
- La présence d'une vue sociale internalisée et explicite dans l'agent, ou bien dissimulée dans les choix faits sur l'architecture de contrôle ou de représentation des connaissances au niveau des modèles internes.

T. Bouron [Bouron, 1992] rassemble cette analyse en séparant les modèles dans lesquels l'organisation est une structure externe aux agents ou non. Nous préférons séparer clairement le point de vue de l'entité informatique et du concepteur humain, puisque l'utilisation de l'organisation comme structuration de l'action a autant de sens d'un point de vue externe qu'interne au système.

Notre objectif est de trouver un point médian entre les modèles explicites, souvent liés à un modèle d'architecture d'agent (souvent peu génériques et intégrateurs) et les modèles de structures sociales très généraux (plus définis comme cadres conceptuels que comme modèle opératoire). Pour cela, il nous faut faire le point sur les caractéristiques communes que nous avons relevé dans la plupart des modèles organisationnels. Nous y relevons comme points clés :

Une structuration de l'espace social, pour faciliter la coopération, l'interaction entre ses membres. L'organisation est avant tout affaire de support à l'activité collective, facilitant l'action collective des agents dans leurs espaces d'action.

Une dynamique associée a cette structure. L'organisation est capable d'évoluer, de faire migrer certains de ses membres d'un point à l'autre de la structure pour améliorer ses paramètres propres, voire d'envisager une *réorganisation* pour répondre à de nouvelles exigences.

Un point de rencontre obligé entre structure sociale abstraite et entités l'incarnant. On sait maintenant bien qu'une étude de la structure de l'organisation en tant que telle et faisant abstraction de ses agents ne peut rendre compte de la richesse des comportements possibles. L'entité individuelle est contrainte par l'organisation mais garde une liberté de décision dans ce cadre.

Une attribution fonctionnelle à l'ensemble de l'organisation. Ceci peut être aussi simple qu'un désir de survie de l'organisation, mais est plus classiquement un comportement que l'on attend de l'ensemble du système : prospérer, remplir un objectif, accomplir des tâches ...

Une capacité d'adaptation supposée. L'organisation est une entité organique, qui sait réagir à son environnement direct, absorber les chocs, survivre aux aléas. Cela concerne également la capacité à résister aux conflits internes : les situations bloquantes doivent être évitables et gérables rapidement pour ne pas pénaliser l'ensemble.

Une identification manipulable. L'organisation reste une entité désignable en tant que telle et l'on peut imaginer et observer plusieurs organisations distinctes, mais basées sur la même définition abstraite.

Cela nous amène à définir un cadre d'action plus précis, qui nous guidera dans l'élaboration d'un modèle organisationnnel générique. Que peux-t'on effectivement supposer sur une organisation, à part qu'elle servira de base à la coopération ?

Le respect de l'autonomie individuelle. Même si l'analyste - ou le concepteur, dans notre contexte de construction de systèmes computationnels - veut pouvoir envisager l'organisation comme une entité unique et identifiable, il n'en demeure pas moins que l'activité est distribuée au niveau de chaque entité agissante du système. Il n'y a pas de locus unique identifiable dans l'action. Il faut garder à l'esprit la clé de la compréhension de la notion d'organisation qu'est ce couplage entre structuration abstraite et incarnation de l'activité. On pourra après mettre un accent plus ou moins prononcé sur l'entité individuelle ou le contexte social, mais cette dualité demeure.

La capacité de séparation. L'organisation suppose une activité structurée (sinon on ne serait pas en présence d'un système *organisé* !). Cela implique de pouvoir considérer, en tant que regard distinct sur le système, une sous-partie de cette organisation. Cette propriété est essentielle : c'est précisement ce qui nous permet de maîtriser un système arbitrairement complexe par une modularisation et une séparation des problématiques. Dans le cadre d'une organisation, nous voulons un minimum d'indépendance dans l'analyse ou la conception des situations d'interaction, des ontologies manipulés, ou même des objectifs des différents groupes d'acteurs.

La possibilité de communication. Si l'organisation permet de structurer l'espace social, c'est précisément pour fournir le *support* d'une mise en relation entre entités. Le travail collectif implique nécessairement une inter-action entre ses membres, qui n'est pas possible sans une communication, quelle qu'elle soit. Notons que l'on ne se restreint pas forcément ici aux communications d'un haut niveau d'abstraction : nous pouvons tout à fait envisager des mécanismes non symboliques comme le marquage d'un environnement partagé. Le point essentiel est que les entités aient *un processus d'interprétation commun* de l'activité de communication.

Nous allons voir dans le chapitre suivant une proposition de modèle intégrateur d'organisations multi-agents ouvertes et hétérogènes, basé sur les principes que nous venons de dégager.

Chapitre 4

Un modèle organisationnel

Dans ce chapitre, nous allons présenter le modèle centré sur l'organisation que nous avons élaboré. Dans nos préoccupations, nous avions évoqué le souci de gestion de l'hétérogénéité (d'architectures ou de langages), de clôture des interactions, de réutilisation, et de réintégration dans des systèmes plus vastes. Nous sommes bien toujours dans le cadre de systèmes *hétérogènes* et *ouverts*.

Nous avançons donc l'idée d'un modèle organisationnel par rapport à ces besoins, sachant que nous devons prendre en compte :

- La puissance de description : nous posons une contrainte forte de neutralité vis à vis des domaines applicatifs et architectures d'entités (tant au niveau des comportements que des interactions)
- L'opérationalisation : nous avons pour but de décrire des systèmes multi-agents *en situation*, nous devons donc être capables de faire le lien avec les modèles internes de l'agent.

Cet aspect opératoire nous conduit à prendre comme notion clé l'agent lui-même. Du fait de l'hypothèse initiale sur son autonomie d'action et sa proactivité, c'est par lui que s'expriment les situations d'interactions dans les systèmes. Ceci implique que nous n'envisagerons pas d'analyse de systèmes *détachés de tout agent*, par exemple en ne travaillant qu'en fonction des schémas de coordination ou de la structure organisationnelle[1].

Nous allons donc ici nous attacher à présenter le modèle que nous proposons, en voyant d'abord comment nous y sommes conduits en fonction des besoins évoqués, puis en y énonçant les concepts primitifs et leurs relations (statiques ou dynamiques). Nous l'illustrerons rapidement avant d'en examiner quelques conséquences et propriétés. Nous verrons alors quelques expressions possibles d'organisations multi-agents. Nous terminerons en montrant comment la dynamique de ce modèle peut s'exprimer dans le même cadre, au meta-niveau.

[1]Néanmoins, lorsque nous aborderons au chapitre 7 les aspects méthodologiques, nous présenterons des modèles situés à un niveau d'abstraction plus élevé, nous donnant alors cette possibilité d'expression.

4.1 Proposition

Dans ce travail, notre préoccupation principale est de proposer des modèles et outils pour prendre en compte et utiliser à notre avantage l'hétérogénéité des systèmes. Nous voulons donc nous situer à un niveau d'abstraction supérieur à celui exprimant les systèmes et leurs entités, en se positionnant au niveau des structures sociales. Néanmoins, le souci d'opérationalisation nous contraint à manipuler des concepts directement exprimables dans les dits systèmes.

Revenons sur chacun des points que nous avons isolés dans notre problématique pour dégager nos hypothèses de travail :

– Nous nous sommes posé le problème de l'hétérogénéité des langages de communication ou des diverses ontologies devant être manipulées dans un système de grande taille. On peut imaginer pour ce faire de créer des zones, dans lesquelles les agents devront avoir un modèle de communication commun. Des agents "interprètes" pourraient alors servir d'interface entre deux zones linguistiques, et laissant ouverte la possibilité à des agents disposant de mécanismes de communications distincts d'échanger néanmoins de l'information.

– La contrainte sur l'hétérogénéité des architectures est plus délicate. Toute contrainte pesant sur le mécanisme de contrôle des agents aurait pour conséquence de les restreindre à des domaines d'applications particuliers. Pour résoudre ce problème, il est envisageable de n'exiger aucun type de *comportement* spécifique (et donc éviter de choisir, par exemple, entre agents situés, rationnels, ...) hormis le fait d'avoir une possibilité de percevoir et d'agir sur la structure sociale environnante. Comme l'on est plus ou moins implicitement dans ce cadre social dès que l'on parle de systèmes multi-agents, on évitera des contraintes supplémentaires sur l'architecture.

– Le souci de clôture et de modularisation pourrait être rapporté à la question de la gestion de plusieurs modèles de communications simultanés. Si l'on veut disposer de systèmes permettant de contrôler la sécurité (comme par exemple l'aspect privé) des interactions, il faut être en mesure de protéger une communauté d'agents contre des échanges indésirables. On pourrait alors reprendre l'idée de "zones" évoquée plus haut, en y ajoutant la possibilité d'y contrôler l'admission ou l'activité : bref, de pouvoir établir des contraintes sur la structure sociale.

– Pouvoir réutiliser, fédérer et capitaliser des systèmes précédemment modélisés demande à ce que l'on puisse caractériser les entités par une certaine attribution fonctionnelle. Connaître les accointances d'un agent dans un système ne suffit pas pour bâtir une abstraction valable. Pour faire le lien, tout en traitant des entités opératoires,

avec un niveau plus abstrait [2], il va falloir que les agents, ainsi que leur modes d'interactions puissent être identifiés en tant que fonctions.

C'est sur cette base de réflexion que nous allons proposer notre modèle, reposant sur les concepts d'agent, groupe et rôle. Nous allons raisonner au niveau des sociétés d'agents : quelles notions définir pour pouvoir exprimer un modèle d'organisation de systèmes multi-agents, suffisamment générique pour être adaptable, et complété par des modèles spécifiques d'architecture ou d'interaction.

Un enjeu de ce modèle - peut-être même la propriété principale recherchée - est aussi la volonté d'un certain minimalisme. Il nous est apparu important de chercher un compromis entre puissance d'expression et parcimonie dans les notions primitives.

4.1.1 Concepts centraux

On a tendance à qualifier d'organisation des structures sociales stables, ou bien présentant des normes sociales explicites. Nous utiliserons ce terme d'une manière plus générale, pour définir des sociétés d'agents analysables en ces termes, quitte à préciser le domaine d'étude et y raffiner les définitions selon le cas. Notre modèle est basé sur l'association de trois concepts clés : l'*agent*, le *groupe* et le *rôle*, utilisés simultanément pour décrire des organisations concrètes d'agents. La figure 4.1 presente un diagramme UML de ce modèle de base.

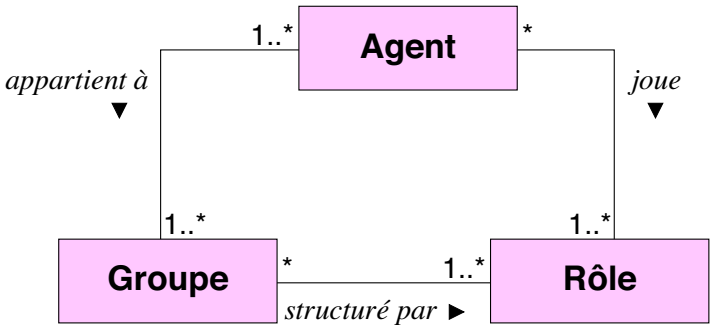


FIG. 4.1 – Les trois concepts centraux

On le résumera par la formule suivante :

> *Un agent peut intervenir dans plusieurs communautés (que nous appellerons 'groupes' dans notre modèle) en parallèle. Il peut jouer dans chacun de ces groupes un ou plusieurs rôles correspondant à ses activités ou interactions. Ces rôles peuvent être portés par un nombre d'agents arbitraire, dépendant de la situation et des normes de l'organisation.*

Voyons maintenant plus en détail ce que nous entendons ici par ces termes.

[2] On pourrait penser ici à la notion de framework avec l'agent en tant que composant

Agent

Dans ce modèle, nous ne posons aucune contrainte ou prérequis sur l'architecture interne de l'agent et nous ne supposons pas de formalisme ou modèle particulier pour en décrire le comportement. On trouvera figure 4.2 le modèle (générique et tout à fait classique) d'agent que nous supposons. Nous pourrions en donner une définition minimale, telle que :

Une entité agissante, autonome, et dont l'action s'exerce dans un contexte social.

Dans le modèle, l'*agent* est simplement décrit comme une entité autonome communicante qui joue des *rôles* au sein de différents *groupes*.

On pourrait reprocher a cette définition son flou, mais nous insistons sur le fait qu'elle est intentionnellement générale pour permettre à chaque concepteur de système de choisir, parmi les modèles classiques, le plus adapté à son application.

Précisons aussi dès maintenant que la nature de l'action sociale de l'agent n'est pas imposé. Ce peut être des mécanismes de coordination reposant sur des actes de langage ou des mécanismes de coopération non symboliques, comme par exemple le marquage d'un environnement. De la même manière, on n'explicite pas *à ce stade* le niveau de représentation de ce contexte social : il peut être internalisé par des connaissances de l'agent sur la société dans laquelle il s'insère, ou bien explicité uniquement au niveau du concepteur.

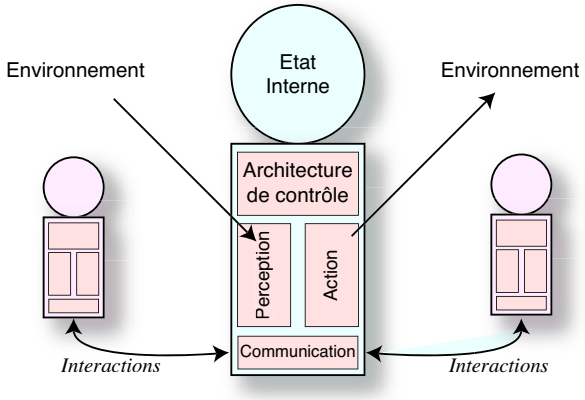


FIG. 4.2 – Modèle d'agent générique

Ce sera par le biais de l'agent que nous ferons le lien avec les politiques d'interaction et de coopération, la dynamique des sociétés, et, à l'opposé, les architectures individuelles des entités.

Notons que même si nous évoquons la notion d'agent dans un contexte opérationnel, cela n'implique pas forcément que la définition de son environnement social soit uniquement faite à ce niveau. Ceci dépend avant tout des choix de représentation et d'architecture interne que fera un concepteur ou un analyste. L'agent est à ce stade une notion "en creux" faite

pour être complétée par un modèle plus classique (d'inspiration cognitive, BDI, éthologique, économique, ...).

Groupe

Nous définissons le *groupe* comme la notion primitive de regroupement d'agents. Chaque agent peut être membre d'un ou plusieurs groupes. Dans sa forme la plus simple, un groupe peut être vu comme un moyen d'identifier par regroupement un ensemble d'agents.

> *Le groupe est avant tout un terme générique pour qualifier une communauté d'agents en relation (par interaction, par partage d'un environnement, par un but ou une ontologie commune, ...).*

D'une manière plus évoluée, le groupe peut être vu comme un SMA usuel. Un point majeur de cette définition est que les différents groupes peuvent se recouper librement, c'est à dire que certains agents peuvent apparaître dans plusieurs groupes sans que l'on doive observer un strict modèle hiérarchique.

Bien que l'expression de la dynamique de ces groupes soit détaillée plus loin, nous pouvons préciser dès à présent que la structuration en groupe d'un ensemble d'agent peut varier avec le temps et que la genèse de l'organisation est vue avant tout comme le résultat de l'action des agents.

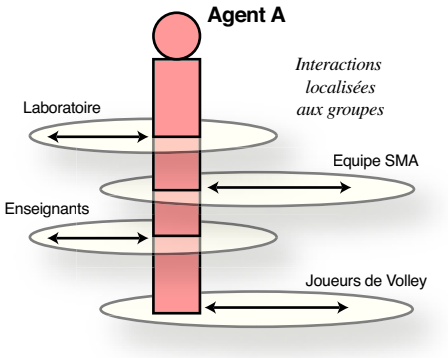


FIG. 4.3 – Le principe du groupe

Prenons l'exemple d'un agent (humain) (figure 4.3) : il peut participer à diverses communautés. Il sera responsable dans une équipe de recherche, dans le même temps simple délégué dans le groupe qui se réactive tous les trimestres pour discuter d'orientations pédagogiques, ou encore disputer dans son temps libre des matchs de volley. Ce que l'on dégagera dans ces situations est la variété des situations d'interactions possibles, toutes très différentes dans leurs manifestations : par exemple, la reconnaissance d'intentions dans le match, la discussion libre et le vote dans une réunion, ou bien la structure assez codifiée

de passivité ou de questions/réponses d'un séminaire. Cela n'empêche pas que ces différentes situations ont toutes leur cohérence, leur vocabulaire propre et leur modèle d'action et d'interaction.

Rôle

Le rôle est la représentation abstraite d'une fonction, d'un point d'interaction ou d'une identification d'un agent au sein d'un groupe particulier. Chaque agent peut avoir plusieurs rôles, un même rôle peut être tenu par plusieurs agents, et les rôles sont locaux aux groupes.

> *On appelle rôle la représentation abstraite d'une fonction du groupe pouvant contraindre le comportement de l'agent, et incarnée dans un ou des comportements spécifiques par l'entité.*

Le rôle va représenter ce que l'on attend comme comportement de l'agent dans l'organisation : travailler en coopération avec d'autres agents, et trouver son positionnement par rapport à l'orgaisation elle-même. Cela peut être une simple tâche à remplir vis à vis de l'application globale, mais également une relation à un statut ou une fonction dans l'organisation. Cette représentation peut se traduire par une stratégie d'implémentation particulière pour les agents mettant en oeuvre potentiellement ces rôles.

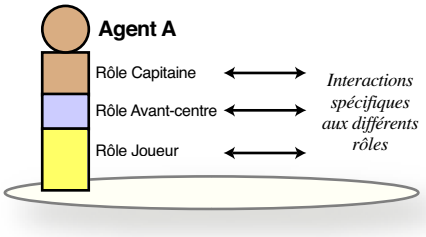


FIG. 4.4 – Le principe du rôle

Pour illustrer cela, prenons l'exemple[3] maintenant classique des matchs de robots footballeurs [Kitano et al., 1998]. On peut associer à un agent (voir figure 4.4) des rôles correspondant à sa position dans l'organisation : être *capitaine* de l'équipe, ce qui évoque plusieurs schémas d'interaction possibles (des indications plutôt impératives sur l'action à mener données aux autres joueurs). Cela n'empêche pas d'avoir d'autres rôles dans cette structure, dont celle de simple *joueur*, qui implique d'autres situations possibles (simplement dialogues ou information de démarquage, par exemple). D'autres dénotent la spécialisation par rapport à une tâche et ce que les autres agents peuvent attendre de lui, comme les capacités et devoirs d'un avant-centre. Ce qui n'empêche nullement, plus tard dans la partie, qu'un autre

[3]Même si nous prenons sans doute un peu d'avance sur ce que l'on trouve dans les équipes actuelles de robots

rôle soit dynamiquement acquis pour faire temporairement le *défenseur* lors d'une offensive dangereuse.

4.1.2 Représentation

Si l'on note
- G l'ensemble des groupes possibles
- R l'ensemble des rôles possibles
- A l'ensemble des agents (en tant qu'entités concrètes)
- C l'ensemble des comportements possibles
- La définition d'un groupe se fait par l'ensemble des rôles possibles que les agents :

$$\begin{array}{rlll} \text{defGroupe}: & G & \rightarrow & 2^R \\ & g & \mapsto & \{r_1, ...r_n\} \end{array}$$

- L'expression de la prise de rôle d'un agent se fait dans l'espace des comportements possibles, si l'on suppose ceux-ci exprimés en fonction des rôles :

$$\begin{array}{rlll} \text{defRole}: & A \times G \times R & \rightarrow & C(=2^{2^R}) \\ & & & processus \quad \text{si } r \in \mathsf{roles}(g) \\ & (a, g, r) & \mapsto & \\ & & & nil \quad \text{sinon} \end{array}$$

4.1.3 Expression des interactions

Jusqu'ici, nous n'avons pas réellement évoqué le problème de la communication entre agents d'un système. Pour rester cohérent avec les hypothèses de ce modèle, nous devons faire intervenir le moins possible un modèle d'architecture spécifique d'agent. Il nous faut donc éviter l'usage d'un langage de communication d'agent ou un support de communication particulier.

Nous ne traiterons donc pas de la communication indirecte, où l'agent échange de l'information via un substrat, sans désigner d'agents concernés par celle-ci. A notre sens, ces techniques de communication via l'environnement, ou via tableau noir, sont plus à mettre en oeuvre au niveau d'un choix d'architecture d'agent ou de système, et viennent donc en complément de ce que l'on décrira au niveau des groupes et rôles.

Nous rattacherons davantage ici la communication directe au modèle : on suppose que les agents interagissent via des séquences de messages. Là encore, nous ne combinerons pas encore *à ce stade* l'interaction avec le déclenchement de comportements particuliers. Voyons comment mettre en rapport les communications directes classiques avec notre modèle :

Communication point à point. L'agent à la source de l'action de communication se fait avec un agent destinataire identifié. On peut raffiner cette situation en deux cas.

– Si l'agent connaît *nominativement*[4] l'agent destinaire la liaison est directe : l'agent émetteur a la connaissance (par une interaction préalable, par un maintien des accointances, ...) de l'individu spécifique
– Si l'agent ne connait que la *fonction* du destinataire quand il veut initier une interaction, on peut alors exprimer cette identification directement par la notion de rôle dans le système. Par exemple, on décrira un schéma d'interaction entre *avant-centre* et *défenseur*, les dialogues entre agents ayant ces rôles s'y conforment alors dans le système final (voir figure 4.5).

Communication par diffusion Il s'agit ici de messages (à comprendre comme des prémisses d'interactions) émis par un agent à un ensemble d'agents du système. Il peut s'agir parfois de diffusion généralisée, où le seul critère pour être destinataire est d'être simplement membre du système, et la compréhension du message n'est pas garantie. L'objectif est souvent d'identifier un sous-ensemble caractérisé d'agents, discriminés par leurs compétences, leur position dans le groupe, comme par exemple un agent envoyant une offre à tous les fournisseurs de services, pour ensuite retenir le meilleur. Là encore, la notion de rôle sera reprise directement pour cette catégorisation. Pour reprendre l'exemple précédent, on pourrait ainsi envoyer un ordre de repli à tous les agents ayant le rôle d'attaquant.

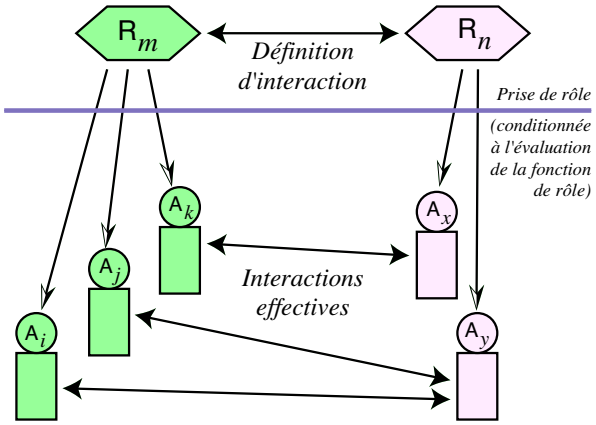


FIG. 4.5 – Relation entre définition de rôles et agents

4.1.4 Dynamique de l'organisation

Nous avons gardé jusqu'ici un point de vue essentiellement statique - et structurel - en envisageant des modèles donnés d'organisation. Néanmoins, notre perspective opératoire centrée sur l'agent nous amène à envisager que cette structure organisationnelle est avant

[4]Quel que soit le mécanisme primitif d'identification des accointances

tout bâtie, perçue et modifiée par les agents eux-mêmes. Cela implique donc de définir les opérations primitives pouvant être accomplies par les agents. On suppose effectivement des agents *sociaux*, et disposant de moyen d'agir et d'identifier leur environnement social, par un biais pouvant dépendre de l'architecture interne retenue.

Opérations de perception de la structure organisationnelle

Le premier style de comportement associé au modèle pour un agent est celui de percevoir son contexte social. On peut donc énumérer comme opérations possibles :

L'inspection des groupes pour qu'un agent puisse savoir ceux dont il fait partie à un moment donné. Il faut noter qu'une inspection globale de l'ensemble des groupes peut très bien ne pas être possible ex nihilo *pour un agent*. Si l'on se place dans la perspective d'une modélisation d'une société d'agents assez complexe, une entité peut tout à fait ne pas percevoir des groupes dont il ne fait pas partie, sauf si un autre agent le renseigne sur l'existence de ces groupes, ou si certains sont supposés connus. Imaginons une grande entreprise : la communauté la caractérisant (ensemble des salariés) peut se représenter de l'extérieur, mais l'existence ou non d'un groupe de travail interne sur un certain thème n'est pas connu a priori.

L'inspection des rôles Dans le modèle, on suppose qu'un agent faisant partie d'un groupe peut connaître les différents rôles tenus par des agents dans le groupe en question. Il s'agit ici de savoir quels rôles va jouer un agent particulier, et quels rôles sont actuellement définis au sein du groupe. Pour reprendre notre exemple précédent, il se peut qu'un salarié connaisse le rôle d'un collègue (par exemple : commercial et supérieur-hiérarchique), ainsi que les différents rôles tenus au sein d'une commission (rapporteur, secrétaire, membre) à laquelle il participe.

Action sur la structure organisationnelle

Assez logiquement, il nous faut également identifier les opérations permettant d'agir du point de vue d'un agent sur l'organisation. On peut dégager les opérations primitives suivantes :

Formation d'un groupe Le groupe n'a de sens que par la présence d'agents. Le groupe est une structure essentiellement dynamique, qui peut être fondée par les agents, et disparaître par retrait de ses membres. Evidemment, en pratique certains groupes vont avoir une durée de vie égale à celle du système (comme l'entreprise dans notre exemple précédent) ou bien être particulièrement fugaces (un groupe n'existant que le temps d'une négociation), selon l'activité des agents.

Acquisition et retrait d'un rôle comme on l'a vu, l'activité au sein d'un groupe est liée à la notion de rôle, qui en caractérise le comportement. Là encore, on fait l'hypothèse que

le rôle est le résultat d'une action explicite de l'agent, pour demander à le jouer, ou l'abandonner.

Il ne s'agit là que d'opérations de base. On peut évidemment imaginer des actions de plus haut niveau composées à partir de celles-ci : par exemple, le transfert d'un rôle d'un agent à un autre (pour confier une responsabilité).

On peut également mentionner le cas où l'on voudrait voir le groupe ou l'organisation être l'initiateur d'un ralliement d'un agent extérieur au groupe. Dans le modèle, cela s'exprime simplement sous forme d'une interaction proposant l'admission entre un agent du groupe et l'agent extérieur, c'est à dire en fonction des opérations primitives du modèle. Rien n'empêche de représenter alors cela comme une opération composée d'un niveau supérieur.

Contrôle et fonction de rôle

Un point délicat demeure malgré tout dans l'expression de ces actions : comment *contrôler* ces systèmes ? Il serait bien évidemment inacceptable qu'un agent puisse demander et obtenir n'importe quel rôle (comme celui de président d'entreprise dans notre exemple précédent), ou bien participer à son goût à n'importe quel groupe. Il nous faut donc un mécanisme pour restreindre les positions possibles accessibles par les agents. Il serait dommage de reporter complètement cet aspect sur le modèle final d'agent : nous allons chercher à exprimer cela au niveau du modèle organisationnel, via la définition des rôles.

Pour exprimer ce contrôle, nous rattachons à la notion de rôle, une *fonction de rôle* exprimant si un agent donné peut ou non obtenir (ou encore garder) le rôle associé. Autrement dit, un agent a peut entrer dans un groupe g pour y jouer un rôle r si une fonction f_r déterminant la contrainte sur le rôle est vraie. Soit R_g l'ensemble des rôles d'un groupe g et r le rôle demandé par l'agent :

$$accepte(a, g, r) \text{ si } \begin{cases} r \in R_g \\ f_r(a) \text{ est vraie} \end{cases}$$

Nous ne définissons pas de mécanisme particulier pour contrôler cet accès, voici néanmoins quelques exemples possibles de fonctions d'admission à un rôle :

Acceptation ou refus automatique. Par exemple, on peut imaginer un groupe *Robocup* avec un rôle de *spectateur* réunissant sans contrainte tous les agents voulant assister à un match de robots footballeurs (sauf s'ils sont sur le terrain)

$$\begin{aligned} f_{spectateur} : \quad A &\rightarrow Bool \\ a &\mapsto \begin{cases} vrai & \text{si } a \notin Robocup_{joueur} \\ faux & \text{sinon} \end{cases} \end{aligned}$$

Admission contrainte par l'état courant du groupe. On peut imaginer de définir un coefficient de similarité entre les membres actuels du groupe et l'agent candidat. Par exemple, ce mécanisme d'admission peut être comparé au principe des "agents de confiance" décrits dans [Lashkari et al., 1994].

On peut aussi imaginer, plus prosaïquement, un groupe refusant d'être trop peuplé en bloquant l'accès aux agents au delà d'un certain nombre (par exemple 42 agents)

$$f_{r42} : \begin{array}{lll} A & \rightarrow Bool & \\ a & \mapsto \left\{ \begin{array}{ll} vrai & \text{si } |g| < 42 \\ faux & \text{sinon} \end{array} \right. \end{array}$$

Admission conditionnée par les compétences. Dans ce mécanisme, un rôle est confié à un agent à partir d'un certain jeu de compétences. En supposant que dans un groupe chaque rôle r_i nécessite un jeu de compétences requises $required(r_i) = \{c_1^i, c_2^i, ..., c_n^i\}$:

$$f_{comp} : \begin{array}{lll} A & \rightarrow Bool & \\ a & \mapsto \left\{ \begin{array}{ll} vrai & \text{si } \forall c \in required(r), c \in competences(a) \\ faux & \text{sinon} \end{array} \right. \end{array}$$

Admission contrainte par l'architecture. L'agent doit exhiber certains prérequis sur sa structure interne pour se voir accepter dans un groupe

Admission soumise à un dialogue préliminaire. Dans ce cas, un agent devra rentrer dans une négociation initiale avec un agent ayant un rôle de gestionnaire du groupe pour déterminer ses droits à l'admission. Ici, la fonction de rôle est en fait internalisée par un agent particulier.

Admission soumise à une position sociale préalable. L'idée est ici de ne laisser un agent accéder à un rôle dans un groupe que s'il est déjà titulaire d'un autre rôle dans une partie différente de l'organisation. Par exemple, on ne laissera participer un agent dans le groupe "conseil de laboratoire" que s'il a déjà le rôle de "permanent" dans le groupe "laboratoire". Cela permet d'exprimer facilement des jeux de dépendances sociales.

4.2 Exemples

4.2.1 Un cas simpliste

Prenons un premier exemple simpliste d'organisation : deux agents jouant au ping-pong (se renvoyant des messages).

– L'activité est structurée par le groupe, ici, chaque agent jouant une partie fera donc partie d'un groupe ping-pong.

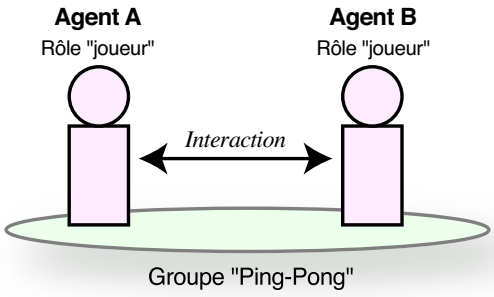


FIG. 4.6 – Un exemple (simpliste) d'organisation

– Les interactions et fonctions de la société sont identifiées par des rôles. On est dans le cas où un seul rôle va être défini, joueur, mais où plusieurs agents tiendront le même rôle.
– Pour jouer au ping-pong, il ne faut être que deux à participer. On peut contrôler la structure via la fonction de rôle. Ici on se contente d'empêcher d'avoir plus de deux joueurs.

$$f_{joueur} : \begin{array}{ll} A & \rightarrow Bool \\ a & \mapsto \begin{cases} vrai & \text{si } |g_{ping-pong}| < 2 \\ faux & \text{sinon} \end{cases} \end{array}$$

On n'impose aucun contrôle supplémentaire

Si cela suffit à définir un modèle pour l'application de ping-pong, on n'a évidement pas terminée la construction du système. Il nous manque encore la définition du protocole d'interaction et des comportements des agents.

4.2.2 L'agence de voyage

Nous allons prendre maintenant un cas un peu plus réaliste même s'il n'est pas d'une originalité éblouissante, vu qu'il s'agit très du célèbre exemple de "l'agence de voyage" qui a servi de cadre d'étude à de très nombreux[5] systèmes à base d'agents [FIPA, 1997] [White, 1996]. Il s'agit de se placer dans un futur évidemment proche, où des systèmes multi-agents auront une place privilégiée pour faciliter des tâches quotidiennes. Des clients sont représentés dans le système informatique par leur agent personnel, fonctionnant sur leur assistant numérique ou leur ordinateur domestique.

L'objectif du système est de faciliter l'organisation d'un voyage, en gérant les préférences de l'utilisateur, la comparaison des tarifs, la planification du trajet et la mise en rapport des

[5]Et que nous ne détaillerons pas ici, car sinon le présent chapitre serait alors plus grand que le reste de cet ouvrage, ce qui est assurément contraire à l'usage

différents intervenants.

Les entités que l'on peut identifier dans ce système sont

- Les clients (exigeants), cherchant à obtenir la meilleure offre pour la destination de voyage de leurs rêves.
- L'agence de voyage, qui va essayer d'établir les préférences et aider un client pour la destination désirée.
- L'agence de voyage en tant que courtier, qui mettra en compétition plusieurs fournisseurs de produits de voyages en fonction de la demande exprimée par un client.
- Les prestataires de voyage, répondant aux appels d'offre par des propositions les plus proches possibles des souhaits exprimés, et prêts à signer un avantageux contrat avec le client initial.

En analysant ce problème au vu du modèle agent-groupe-rôle, nous pouvons déjà identifier deux mondes différents : la relation entre clients et agence de voyage, et l'espace de la négociation entre agence et fournisseurs. Les langages d'interaction, et même les ontologies n'ont à priori aucune raison d'être les mêmes entre ces deux communautés : on va donc représenter le système en trois groupes, l'un pour la partie consommateurs, l'autre reservé aux professionnels du voyage (qu'on imagine observer un standard agent comme FIPA ou KQML [Finin et Fritzson, 1994]). On fera la supposition que la vente finale est faite directement entre fournisseur et client : cette transaction privée aura lieu dans son propre groupe éphémère.

A partir de cela, on représente les différents comportements attendus sous forme de rôles : client et agence pour le groupe consommateurs, courtier et fournisseur pour le groupe professionnel. Le groupe de signature du contrat aura pour sa part que vendeur et acheteur.

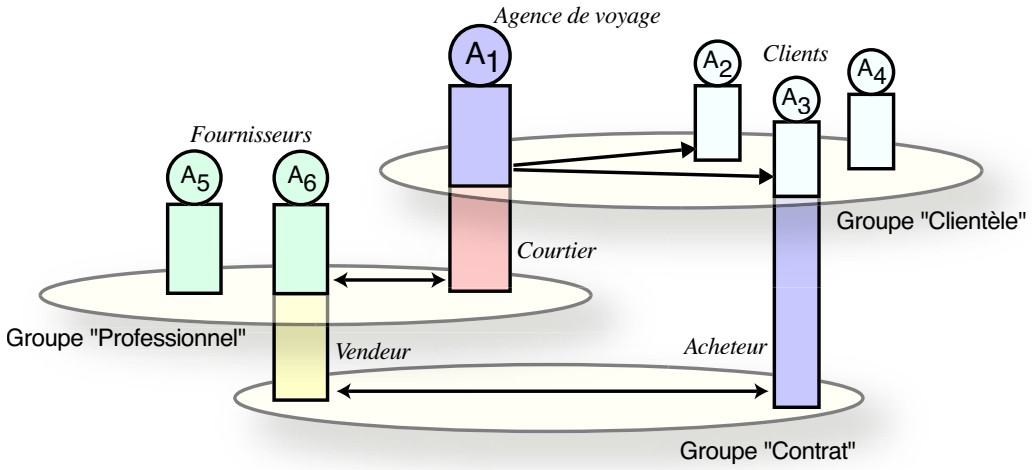


FIG. 4.7 – L'exemple de l'Agence de Voyage modélisée en groupes et rôles

On aboutit donc[6] à un système organisé autour de trois modèles de groupes et de six rôles, mais l'organisation concrète - le système multi-agents - pourra évidemment rassembler bien plus d'entités. Par exemple, s'il est raisonnable de n'avoir qu'un agent ayant le rôle d'une agence de voyage dans le groupe consommateur, on va avoir un nombre non limité d'agents prenant le rôle de client. Précisons également que nous faisons l'hypothèse ici d'une seule agence de voyage dans ce scénario et cette séparation en groupes : dans le cas de plusieurs agences, on aurait autant de groupes de clientèle que d'agences, toutes les agences (avec leur rôle de courtier) et les fournisseurs pouvant continuer à travailler dans un groupe commun.

La possibilité d'appartenance multiple à différents groupes s'avère ici fondamentale : c'est par cela que l'on représente les deux aspects de l'agence de voyage, appartenant à la fois au monde des clients et au monde des fournisseurs, mais correspondant à la même entité concrète. De même, c'est par ce biais que le client et le fournisseur final participent à un groupe éphémère et entament une interaction très précise de paiement, sans risquer d'interception de leurs messages et en minimisant le travail de mise en oeuvre de cette interaction.

Tout comme dans l'exemple précédent, pour aller jusqu'au bout de la réalisation du système, il faudrait maintenant spécifier complètement les schémas d'interactions et définir les comportements et architectures de contrôle des différents agents. Notons que la séparation en groupe et l'identification fonctionnelle par les rôles permet de minimiser les suppositions faites sur l'architecture des autres agents : client et fournisseur, par exemple, pourraient très bien être pour l'un une simple interface WWW contrôlée par l'utilisateur et pour l'autre un système à base de règles.

4.3 Propriétés et conséquences

Après avoir illustré brièvement le modèle sur ces exemples, revenons sur quelques conséquences de ce modèle.

4.3.1 Hétérogénéité

La maitrise de l'hétérogénéité des situations d'interaction est rendue possible par le fait qu'un agent peut avoir plusieurs rôles distincts au sein de plusieurs groupes, chaque politique de communication étant locale à un groupe.

Les notions d'agent, de groupe et de rôle sont des concepts primitifs du modèle et il n'existe pas de mécanisme permettant d'exprimer l'un d'eux en fonction de l'un ou des deux autres.

[6] Le lecteur excusera cette conception de SMA menée tambour battant, le chapitre 7 présentera plus en détail une approche méthodologique possible

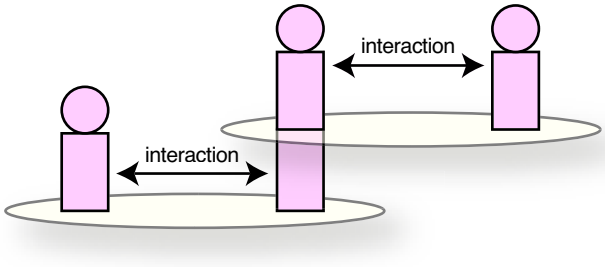


FIG. 4.8 – Agent en multi-groupe

On a vu également que l'intervention d'un agent dans plusieurs groupes permet d'établir une "passerelle" d'information entre deux portions d'un système potentiellement incompatible. On suppose dans ce cas l'agent interface capable d'interagir dans les deux modèles d'interaction relatifs à l'un et l'autre groupe.

Par exemple, les modèles classiques sans structuration du système multi-agents font la plupart du temps la supposition que le mécanisme d'interaction est homogène. Quand des agents sont dans des groupes sans aucune relation, il n'existe plus aucun lien architectural ou conceptuel entre les deux SMA ainsi séparés. Intégrer des applications hétérogènes devient beaucoup plus aisé.

4.3.2 Modularité

Remarque 1 *Un agent agissant dans un groupe n'a pas le moyen de connaître l'activité d'un autre groupe auquel il n'appartient pas. Plus encore, à moins qu'un autre agent ne lui ait transmis une information spécifique, il n'en connaîtra même pas l'existence.*

Une paraphrase de cette remarque pourrait être la définition d'une caractéristique d'encapsulation organisationnelle : l'agent ne voit nativement que les portions de la structure sociale dans lesquelles il agit (figure 4.9). Les autres parties de l'organisation lui sont invisibles, ainsi que leurs modèles propres. Cela implique que dans une organisation, toute modification de la définition d'un groupe auquel un agent n'appartient pas, n'aura aucune conséquence sur lui. Ce groupe peut disparaitre, voir ses mécanismes de communications changer, ses protocoles être redéfinis, il n'en subira aucunement les effets. Une application pratique immédiate de cet état de fait est bien sûr la modularité et les capacités de réutilisation de systèmes conçus en multi-groupes.

Remarque 2 *Si l'on reste au niveau du modèle générique agent-groupe-rôle, les pré-conceptions d'un agent par rapport à un autre se limitent aux connaissances relatives au rôle de l'autre dans un groupe donné.*

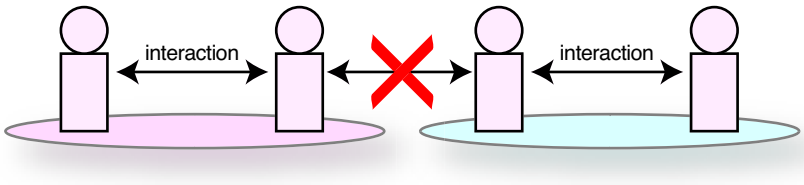


FIG. 4.9 – Le groupe comme encapsulation

Là encore, la conséquence est une relation d'abstraction : les accointances d'un agent se définiront par les identités de ses pairs ainsi que des rôles qui leur auront été associé. L'architecture interne de l'agent n'intervient pas directement dans cette définition. Cela implique qu'un rôle tenu par un certain agent dans l'organisation peut être plus tard tenu par un autre agent d'un type complètement différent sans que l'organisation soit perturbée. La seule contrainte est que les autres agents puissent continuer à exploiter la définition de rôle qu'ils avaient.

Bien entendu, dans la pratique cela pourra se décliner d'une manière un peu différente.

Néanmoins, on notera un effet de bord intéressant de cette définition. Supposons un groupe utilisant un modèle de communication particulièrement riche faisant appel à de multiples représentations de l'autre comme les modèles Belief/Desire/Intention. On peut introduire dans cette organisation un agent ne se conformant pas à cette architecture, à partir du moment où il est capable d'en respecter les contraintes ou les normes (par exemple pour un comportement spécifique). En effet, un agent en dialogue avec un autre ayant le rôle idoine a une vue de son interlocateur quasiment construite par attribution : en tant qu'agent, c'est parce que je peux supposer le respect de contraintes (schémas d'interactions, propriétés) associées au rôle tenu que je peux interagir.

Nous verrons que cette propriété entraîne des conséquences importantes sur la montée en puissance (traduction incertaine de la *scalability* anglo-saxonne) des systèmes. Il n'y a aucune différence visible du point de vue d'une entité demandeuse entre un agent effectuant seul une tâche et un agent l'effectuant en la sous-traitant à un autre groupe dont il fait partie.

4.3.3 Fiabilité

Comme toute communication n'est possible qu'entre agents membres d'un même groupe, on remarquera que le groupe va agir comme mécanisme d'*encapsulation* des interactions inter-agents. Cette propriété va s'avérer fondamentale pour la montée en charge du modèle.

Cette encapsulation interactionnelle a en effet plusieurs propriétés dignes d'intérêt :

Fiabilité La présence d'agents au sein d'un système suppose l'adhésion de tels agents aux schémas d'interactions ou aux ontologies ayant court dans ce système.

On a par exemple le point épineux[7] des agents incapables de comprendre les messages qui leur sont envoyés. Ce mauvais comportement peut avoir de multiple causes : mauvaise implémentation, sémantique incomplète sur le protocole d'interaction, non-accord sur l'ontologie, ou même simple félonie du destinataire.

On peut également noter qu'une approche souvent retenue est l'utilisation d'un performatif particulier, tel que `not-understood` pour que le récepteur d'un message incompréhensible puisse signaler à son envoyeur qu'aucune action attendue ne pourra être prise. Le talon d'Achille de cette technique est que cela suppose la connaissance même *chez l'agent receveur* de cette échappatoire, ce qui n'est absolument pas garanti.

Tout effort pour éviter les situations d'interactions non maîtrisables entre agents hétérogènes ne peut donc que fiabiliser le système global, et la structuration par groupe en est un.

Sécurité La dernière propriété intéressante de cette structuration est la sécurité induite par masquage. Dans deux groupes non connexes, il ne peut y avoir d'interaction directe sans entrée dans un groupe. On peut donc éviter plusieurs risques potentiels : divulgation erronée d'information, déguisement d'un agent en un autre, refus de service par saturation en messages d'un agent, ou espionnage des interactions. Même si la majorité des concepteurs de systèmes multi-agents actuels estiment - à raison - qu'il est trop tôt pour intégrer ces préoccupations dans leurs systèmes, ce sont des points qu'il faudra assurément résoudre avant toute tentative d'exploiter ces architectures dans des cadres plus vastes.

4.3.4 Hiérarchisation

Le groupe n'est pas un type d'agent particulier ayant une capacité de définitions récursives. De même, l'agent n'est pas un groupe particulier fédérant ses activités ou composants internes. De même, des idées de sous-groupes ne sont pas définissables directement dans le modèle puisque nous n'avons pas de fonctions liant directement un groupe à un autre.

Ce n'est pas parce que le modèle n'accepte pas de notion *intrinsèque* d'agent récursif ou de relations entre groupes que de tels modèles y sont impossibles. Nous allons voir que de telles structures sont tout à fait exprimables.

Une manière facile d'exprimer un modèle d'organisation strictement hiérarchique est d'abord de représenter deux niveaux d'activités. On peut identifier ici une structure de *représentant*. La hiérachisation vient de la contrainte posée sur les rôles entre les deux groupes :

> Un agent ne peut être le représentant d'un groupe G_1 dans un groupe G_2 de niveau supérieur si et seulement si il a simultanément le rôle de délégué dans le groupe G_1 (voir figure 4.10).

[7] tant d'un point de vue théorique que pratique, d'ailleurs

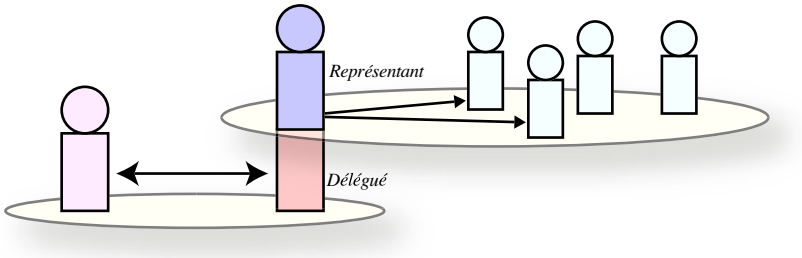


FIG. 4.10 – La structure de représentant

A partir de ce modèle, l'expression d'une organisation strictement hiérarchique est aisée : il suffit de considérer qu'un agent représentant d'un niveau inférieur va être un simple membre dans le groupe supérieur, qui aura lui-même un agent exerçant le rôle de délégué, puis d'itérer cette structure (voir figure 4.11).

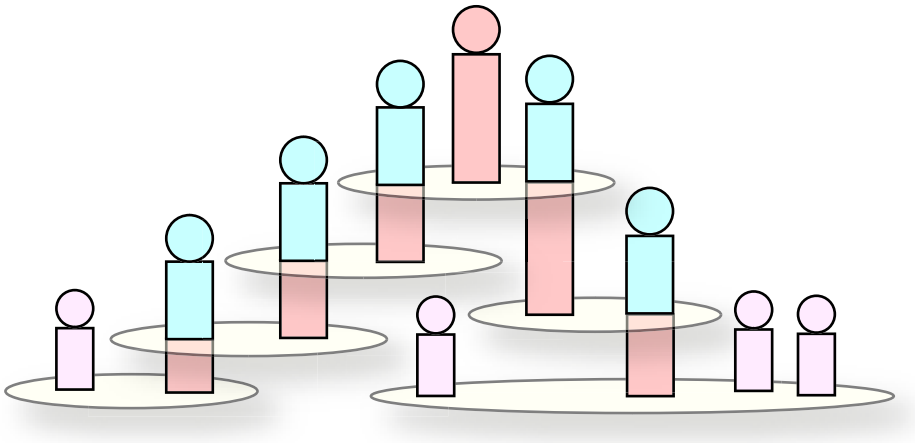


FIG. 4.11 – Une structure hiérarchique

4.4 Aspects réflexifs

Jusqu'ici, quand nous avons mentionné les normes sociales exprimées par le biais des fonctions de rôle, nous n'avons pas explicité la façon dont l'évaluation de ces contraintes se faisait.

Selon les systèmes, ces contraintes pourraient être de simples guides de conception sans réalité implémentatoire, des vérifications rigides faites sur le système final, ou des connaissances sur la structure sociale refletées dans chaque agent. Même si toutes ces stratégies sont possibles, et s'il serait audacieux d'en préférer une par rapport aux autres, il en existe une

expression particulièrement intéressante, car exprimée dans le même modèle.

En effet, si l'on suppose que la mise en place et le respect de contraintes sociales doit être faite par les agents eux-même, on peut alors définir la dynamique de l'organisation au niveau meta.

Le principe est de voir la construction de groupes, le contrôle de rôle, l'entrée ou le retrait d'un agent dans une organisation comme étant le résultat d'interactions avec des agents particuliers, chargés de veiller à la bonne marche de l'organisation multi-agents. Il est alors possible de définir une structure organisationnelle chargée du contrôle de cette même structuration organisationnelle, réflexivement.

Voyons comment peut se structurer un tel meta-niveau :

Les entités considérées sont ici les agents et leur structuration sociale par groupes et rôles.

Les opérations à exprimer au méta-niveau sont les opérations de perception et d'action sur la structure sociale, telles que nous les avons identifiées à la section 4.1.4.

Le meta-niveau est constitué d'agents, structuré en groupe et rôle chargés de contrôler, gérer l'ensemble de l'organisation.

La meta-circularité est exprimable par le biais d'un groupe permettant la coordination des différents agents gestionnaires de groupe.

A partir de ces principes de fonctionnement réflexif, on pourrait dégager plusieurs manières de mettre en oeuvre le meta-niveau :

- Un seul groupe est défini pour structurer le meta-niveau, le groupe des gestionnaires, auquel peuvent s'adresser les candidats à un certain groupe applicatif, et contrôlé par un meta-gestionnaire
- Le contrôle de groupe pour un groupe A est systématiquement représenté dans un meta-groupe-A, qui ne comprendra que les rôles de niveau meta (gestionnaire, membre, candidats).
- Le contrôle d'un groupe A est fait dans ce même groupe A, via des meta-rôles permettant d'exprimer le contrôle, avec en particulier celui de gestionnaire.

Par souci de concision, nous ne retiendrons ici que la première de ces approches pour présenter un bref exemple.

Etre gestionnaire de groupe correspond à une qualification spécialisée, une compétence : l'agent gestionnaire du groupe A sera l'agent responsable de l'évaluation des fonctions de rôles pour le groupe considéré, et donc maître des choix d'entrée d'autres agents sur ce groupe. Un agent candidat à un rôle r dans $G1$ sera soumis à la décision du gestionnaire. Comme on l'a vu, cette candidature peut parfaitement être définie de manière générique par une interaction agent, entre un rôle de *candidat* et un rôle de *gestionnaire* (voir figure 4.12).

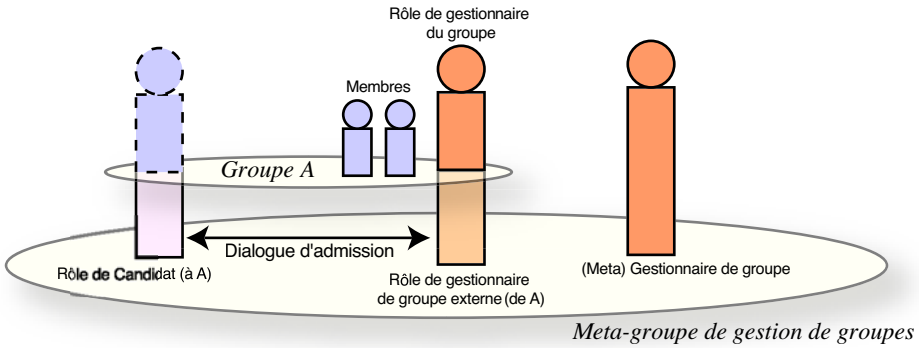


FIG. 4.12 – L'admission à un groupe défini au meta-niveau

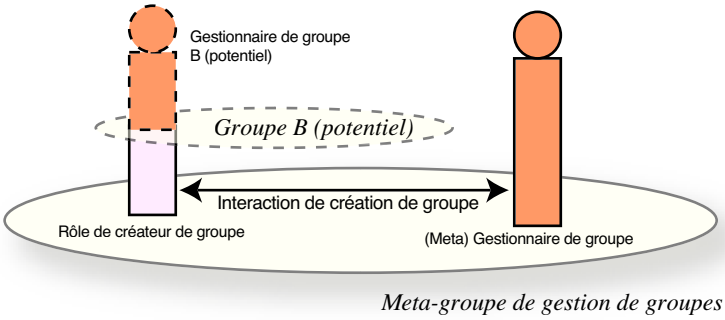


FIG. 4.13 – La création de groupe définie au meta-niveau

L'agent ayant un rôle de "gestionnaire"[8] dans le groupe A aura donc dans ce groupe G des gestionnaires un rôle de "A-gestionnaire". C'est l'acceptation de ce rôle qui conditionne son pouvoir dans le groupe A. Mais comment se passe l'octroi de ce rôle spécifique dans G ? Nullement différement : l'impétrant se devra de demander ce rôle au gestionnaire de G qui le lui accordera ou non selon un jeu de critères généraux ou spécifiques au rôle demandé (voir figure 4.13).

La gestion d'un groupe de gestionnaires de groupe utilise exactement les mêmes mécanismes que tout autre groupe.

Evidemment, comme toute architecture réflexive, se pose la question de l'amorçage du modèle : on doit supposer le gestionnaire de groupe du groupe des gestionnaires de groupes comme préexistant.

Notons que supposer un tel modèle n'implique pas de faire obligatoirement des aller-retours entre meta-niveau et dénoté, surtout si on n'a pas défini de fonction de rôle spéci-

[8] On peut identifier cette compétence de gestionnaire au fait d'être membre d'un groupe qui sera celui fédérant tous les gestionnaires de groupes

fique. Voici par exemple un scénario de prise de rôle possible dans un système qui implémenterait le meta-niveau comme on l'a décrit :

1. Un agent a désire prendre un rôle r dans un groupe g. Il effectue cette demande.
2. Si aucun gestionnaire de groupe n'existe pour le groupe g, la politique de contrôle est inexistante. Aucun saut à un niveau externe n'est donc requis et le rôle est accordé.
3. Si un gestionaire de groupe m existe pour le groupe g, celui-ci reçoit la demande de a. Il en évalue la validité par rapport à la fonction de rôle $f_r(a)$.
4. L'agent a voit sa demande accordée ou refusée et la décision se reflète dans la structure du groupe g.

Pour conclure cet examen du modèle selon un point de vue réflexif, on pourrait ajouter qu'une manière de voir cette modélisation au niveau meta est de comprendre le gestionnaire de groupe comme un dépositaire et garant de la structure d'un groupe (et donc de la définition de ses rôles admis et des fonctions associées). C'est un peu l'oeil du concepteur toujours présent dans le système.

4.5 Conclusion

Nous avons présenté un modèle organisationnel de systèmes multi-agents basé sur les notions primitives d'agent, de groupe et de rôle, sa dynamique et son illustration sur quelques exemples et sur une expression possible au niveau meta. Nous avançons donc l'idée qu'une partie importante d'un système multi-agents peut être décrite déjà au niveau organisationnel avant de l'envisager au niveau individuel par les modèles de connaissance ou de contrôle.

Orienter le concepteur vers un point de vue social a cet avantage de laisser la porte ouverte à un enrichissement progressif du système, par le rajout au besoin de nouvelles structures dans l'organisation. La participation (contrôlée) d'agent à plusieurs groupes simultanés permet d'assurer la cohérence et de maintenir un point de vue opératoire.

On l'a bien vu, ce modèle reste parcellaire et ne prend son sens qu'associé à des modèles d'agents spécifiques. L'agent individuel est avant tout dans l'oeil du concepteur, dès lors que celui-ci est capable de répondre positivement à la question de savoir s'il est de son intérêt ou non de voir un agent dans telle entité de son système.

Nous avons justifié dans cette partie notre approche basée sur l'organisation et présenté les aspects statiques, dynamiques et réflexifs de notre modèle. Nous allons voir au fil des chapitres suivants comment décliner ce trio agent-groupe-rôle selon d'autres points de vue : sémantique formelle, implémentation, méthodologie ou aspects applicatifs.

Troisième partie

Variations sur un modèle

Chapitre 5

Modèle formel opératoire

5.1 Motivation

Nous nous proposons dans cette partie de reprendre l'étude du modèle organisationnel en l'explorant dans ses aspects formels. Ce qui va nous intéresser dans ce chapitre est d'évaluer les conséquences du point de vue organisationnel sur le comportement des agents.

Nous avons jusqu'ici présenté un jeu de concepts de base pour la description de structures sociales dans les systèmes multi-agents, en même temps que nous en donnions une interprétation suffisante pour l'utiliser comme guide d'analyse de systèmes. Néanmoins, nous voudrions également avoir une définition précise du comportement *social* de ces agents d'un point de vue opérationnel, ce qui implique d'être capable de leur associer une sémantique formelle.

Il existe plusieurs efforts pour exprimer une sémantique opérationnelle globale pour les systèmes à agents, tels que CONGOLOG [Shapiro et al., 1998] et AgentSpeak[Rao, 1996], ou des logiques spécialisées [Wooldridge, 1996]. Néanmoins, ces formalismes restent contraints par une spécialisation sur quelques architectures d'agents (par exemple, une grande majorité fait l'hypothèse de systèmes cognitifs communicants). Nous cherchons au contraire à nous focaliser sur les actions primitives du modèle agent-groupe-rôle et les analyser découplées de tout modèle individuel, en nous donnant ensuite le moyen de faire le lien. Pour ce faire, nous nous appuierons sur une algèbre de processus (le π-calcul) et quelques machines abstraites.

Comme les systèmes multi-agents sont des structures relativement complexes, il va être délicat de décrire leur action de façon complète sur l'ensemble d'un comportement. Nous nous restreindrons donc à établir une sémantique opérationnelle des *actions élémentaires* sur les groupes et les rôles que nous avons dégagés précédemment.

Pourquoi une algèbre de processus ? Les descriptions de haut-niveau traditionnelles : sémantique naturelle, sémantique dénotationnelle standard, ne sont pas vraiment adaptées pour décrire des aspects massivement concurrents comme les systèmes que l'on envisage

ici. Nous allons donc plutôt nous tourner vers les algèbres de processus, plus appropriées. Une telle algèbre peut se définir comme un système formel, doté d'une syntaxe, d'une sémantique, de relations d'équivalences, qui tente d'exprimer un modèle de calcul à base de processus concurrents.

Nous allons ici utiliser le π-calcul de Robin Milner [Milner et al., 1992], qui concentre toutes les caractéristiques classiques du calcul concurrent dans un formalisme simple et extensible. La sémantique classique du π-calcul est celle de processus s'échangeant des messages à travers des canaux de communication. Le tout est fondé sur une réécriture de termes où la communication consiste en une substitution de noms. Nous en utiliserons une expression sur une machine abstraite, la CHAM.

Nous commencerons par donner une brève description des caractéristiques notables du π-calcul et de la CHAM. Nous montrerons alors quelques exemples, dont une manière de transcrire des systèmes à agents "classiques" dans ces formalismes. Nous étudierons alors comment notre modèle d'organisation peut s'inscrire dans ce cadre formel, ce qui nous amènera à proposer quelques extensions. Nous finirons en étudiant dans ce même formalisme la dynamique et les aspects méta d'organisations ainsi représentées.

5.2 Un modèle de calcul : le π-calcul et la CHAM

5.2.1 Présentation

Le π-calcul est un modèle de calcul offrant une fondation formelle à l'expression de processus concurrents. En caricaturant un peu, on pourrait le qualifier de "λ-calcul" du parallèlisme.

Pour reprendre la définition originale [Milner et al., 1992], on peut dire qu'il s'agit d'une

> *... manière de décrire et d'analyser les systèmes composés de processus qui interagissent entre eux et dont la configuration et l'environnement sont continuellement en changement.*

Techniquement, le π-calcul est une extension du CCS (Calculus of Communicating Systems) [Milner, 1989] qui prend en compte le passage de noms de canaux en plus de passage de valeurs. La différence avec CCS vient du fait qu'il est possible de transmettre des noms de ports lors d'une communication.

Par exemple, imaginons un processus *Standardiste* S, un *Client* C et un *Technicien* T. Le client aimerait être mis directement en communication avec le technicien pour obtenir un prix (42 F), et qu'un canal de communication a existe de C vers S et un canal b de T. Le π-calcul permettant de transférer des noms de canaux[1], le processus S donnant le canal de communication b à C, les envois faits par T arriveront alors directement sur le port de C.

[1] contrairement au CCS.

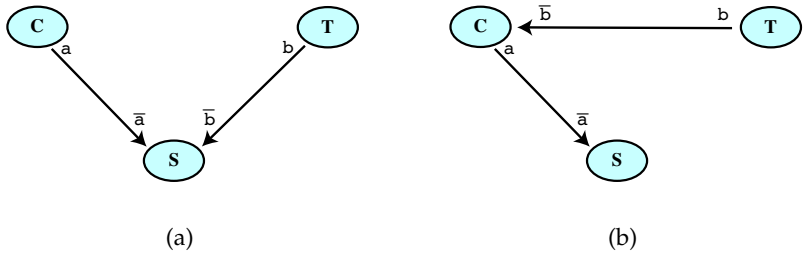


FIG. 5.1 – Transfert d'un canal d'un processus à l'autre en π-calcul

Chaque expression π-calcul dénote un processus minimaliste. Ses seuls comportements possibles sont l'échange de messages sur des canaux de communications. De plus, tous les messages ne sont que références à d'autres canaux. C'est cet aspect minimaliste qui inspire cette comparaison avec le λ-calcul. Il offre un modèle bas-niveau de calcul (parallèle), où l'équivalent de la fonction serait le processus, et où la communication jouerait le rôle de l'application.

Le π-calcul décrit ici est *synchrone* : on admet que les envois de messages sont faits instantanément. Les réceptions sont bloquantes jusqu'à l'arrivée d'un message. On se placera également dans la variante *polyadique* du π-calcul, dans laquelle plusieurs messages peuvent être simultanément émis et reçus sur le même canal. [2]

Comme exemple révélateur de la puissance d'expression de ce modèle, Milner a montré que l'on peut exprimer le λ-calcul en π-calcul [Milner et al., 1992] (l'inverse n'étant pas vrai).

5.2.2 Définitions et syntaxe

En π-calcul, toute expression dénote un processus, à savoir une activité d'exécution indépendante qui fonctionnera en parallèle avec d'autres processus. Un processus peut également définir plusieurs sous-processus si nécessaires. Les communications entre processus ont lieu au travers de ports de communications par envoi de messages.

Expressions primitives

Les règles suivantes donnent la syntaxe des expressions du π-calcul. P et Q sont des méta-variables dénotant des processus ; x, y, z, v et u sont des méta-variables qui dénotent des noms.

[2] A noter que cette variation s'exprime simplement en π-calcul monoadique, qui suppose lui qu'un seul message puisse être émis ou reçu à la fois.

P, Q ::= **0**
| $P \mid Q$
| P ; Q
| x(z).P
| $\overline{\mathtt{x}}\mathtt{y}.\mathtt{P}$
| C ?P
| ν(x)P
| $\mathtt{A}(\mathtt{y_1}, .., \mathtt{y_n})$

1. **0** est le processus nul, c'est à dire la processus qui ne fait rien. Il est souvent omis. Par exemple,
   x(y).$\overline{\mathtt{y}}$(a).**0** est souvent écrit plus simplement x(y).$\overline{\mathtt{y}}$(a)
2. $P \mid Q$ est la composition parallèle de deux processus P et Q en exécution concurrente. Ces deux processus évoluent indépendamment et peuvent éventuellement communiquer via un canal.
3. $P; Q$ signifie que ce processus va se comporter comme P ou comme Q. Les processus P et Q sont alors généralement précédés d'une condition.
4. $x(z).P$ est le préfixe d'entrée (ou *préfixe positif*). Cela peut être vu comme le processus P qui reçoit un message sur un canal de communication x en entrée ; z est le paramètre qui va désigner le message envoyé sur le canal[3]. Quand le processus a reçu le message y dans z, le préfixe est supprimé et le processus se comporte comme $P[y/z]$.
5. $\overline{\mathtt{x}}y.P$ est le préfixe de sortie (ou *préfixe négatif*) : x est le port de sortie sur lequel le nom y est envoyé. Une fois que la transmission est faite, $\overline{\mathtt{x}}y.P$ se réduit en P.
6. $C?P$ est un processus gardé. Si la condition C est vérifiée, alors $C?P$ est réduit en P, sinon il se transforme en processus nul **0**. La condition est supposée primitive. Dans le π-calcul brut, la condition est supposée être une égalité sur deux noms (par exemple $[x = y]P$. Nous généraliserons ici à toute opération dite primitive.
7. $\nu(x).P$ définit un nouveau nom x distinct de tous les noms présents dans P. Le nom x est privé à P et ne peut être utilisé comme canal de communication avec d'autres processus.
8. $A(y_1, .., y_n)$ est un processus définissable par une équation de la forme $A(x_1, ..., x_n) = P$ où les noms $x_1, ..., x_n$ sont les seuls noms libres dans P. Le processus $A(y_1, .., y_n)$ se comporte comme P dans lequel chaque x_i aurait été substitué par y_i, et ce pour tout i.[4]

Les règles que l'on a données ici étaient en fait celle du π-calcul monoadique. Pour l'étendre au π-calcul polyadique, il suffit de considérer que les émissions ou réceptions sur un canal se font sur des n-uplet $y_1 y_2 ... y_n$.

[3] Ceci est déjà une interprétation : le π-calcul brut ne définit que des noms et pas les variables.

[4] Ceci est en fait l'expression du codage d'une fonction à nombre fini de paramètres avec paramètres formels et paramètres d'appels.

Relations d'équivalence

Certaines expressions sont "équivalentes" au sens où elles peuvent être substituées l'une à l'autre sans que rien ne change au calcul. Cette équivalence est notée par une relation de congruence structurelle $\equiv$.

Par exemple, une de ces relations formalise l'intuition que l'on peut avoir de l'ordre des sous-expressions est sans conséquence par rapport à l'opérateur ' | ' (qui est à la fois commutatif et associatif).

$$\begin{aligned} P \mid Q &\equiv Q \mid P \\ (P \mid Q) \mid R &\equiv P \mid (Q \mid R) \\ !P &\equiv !P \mid P \end{aligned}$$

La sémantique opérationnelle du π-calcul est donnée sous la forme d'un ensemble de règles de réductions comme dans le λ-calcul [Barendregt, 1984].

La règle principale décrit comment les processus peuvent communiquer par envoi de messages via des canaux de communication :

$$\overline{x}y.P \mid x(z).Q \rightarrow P \mid [y/z]Q$$

D'autres règles caractérisent les propriétés de parallèlisation processus, à savoir le fait que les réductions ne sont pas modifiées lorsque les processus sont placés en parallèle. [Pierce, 1996] :

$$\begin{aligned} P \mid R &\rightarrow Q \mid R \text{ if } P \rightarrow Q \\ \nu(x)P &\rightarrow \nu(x)Q \text{ if } P \rightarrow Q \\ P &\rightarrow Q \text{ if } P \equiv P' \text{ and } Q \equiv Q' \text{ and } P' \rightarrow Q' \end{aligned}$$

Dans la suite de notre propos, nous utiliserons comme style d'écriture des envois de messages par continuation, bien connu dans le monde des langages d'acteurs [Agha, 1986], bien qu'aussi utilisé parfois pour la description de structures multi-agents [Ferber et Carle, 1991].

Nous supposerons ici que la continuation locale est passée en dernier argument des messages. Par exemple, voici une version de la classique factorielle écrite en π-calcul dans un style utilisant les continuations :

```
Fact(n,c) =def (n = 0)?(c̄(1));
               (n ≠ 0)?(ν(c1)(Fact(n-1,c1)|
                              c1(r).c̄(n*r)))
```

5.2.3 La CHAM : Chemical Abstract Machine

La Chemical Abstract Machine (ou CHAM en abrégé) a été proposée par G. Berry et G. Boudol [Berry et Boudol, 1992] en 1992. La CHAM est une machine abstraite basée sur une métaphore "chimique". On peut faire remonter cette proposition au langage Γ forgé [Banâtre et Le Métayer, 1990]. Cette machine a été introduite pour fournir un cadre de travail formel permettant la définition de sémantiques opérationnelles de langages parallèles et une description aisé de mécanismes de calcul dans le domaine du parallèlisme. Depuis, G. Boudol a montré qu'un bon nombre de constructions théoriques comme le π-calcul pouvaient être décrites dans la CHAM [Boudol, 1992].

Une configuration de la CHAM peut être décrite par une *solution* qui prend la forme d'un multiset d'éléments (appelés molécules), et notées : $S = \{|m_1, ..., m_k|\}$.

Une CHAM est spécifiée par la définition des molécules, construites en utilisant quelque syntaxe abstraite et un jeu *règles de réaction* qui donnent les transformations licites d'une molécule en une autre au sein d'une solution.

Une règle de réaction est de la forme :

$$\{|m_1, ..., m_k|\} \rightarrow \{|m'_1, ..., m'_n|\}$$
$$\text{si condition } (m_1, ..., m_k)$$

Cela signifique que si les molécules du côté gauche de l'expression sont présentes dans une solution et si une condition sur ces molécules est vérifiée, alors elles sont remplacées par les molécules spécifiées au côté droit.

Le fonctionnement de la CHAM obéit à quelques lois générales. La première loi, appelée *loi chimique* exprime formellement qu'une réaction peut se produire librement dans toute solution :

$$\frac{S \rightarrow S'}{S \uplus S" \rightarrow S' \uplus S"}$$

La CHAM a été utilisée pour décrire plusieurs modèles de calcul concurrent comme CCS [Milner, 1984] (voir [Berry et Boudol, 1992]) ou le π-calcul asynchrone (voir [Boudol, 1993]).

La première manière, un héritage de GAMMA a été proposé par Banâtre et Le Metayer. Dans ce style d'écriture, on utilise des jeux de règles pour exprimer des algorithmes. Par exemple, notre factorielle peut être décrite avec simplement deux règles :

```
fact(n) & n ≠ 0 → fact(n − 1), cfact(n)
cfact(n), rfact(n-1,r) → rfact(n,r*n)
```

Pour trouver le résultat au calcul d'une factorielle 4, il suffit de démarrer la machine avec une solution de départ contenant :

```
{|fact(4), rfact(0,1)|}
```

Voici les différentes étapes connues par notre solution :

```
{|fact(4), rfact(0,1) |}
{|fact(3), cfact(4), rfact(0,1) |}
{|fact(2), cfact(3), cfact(4), rfact(0,1) |}
{|fact(1), cfact(2), cfact(3), cfact(4), rfact(0,1) |}
{|fact(0), cfact(1), cfact(2), cfact(3), cfact(4), rfact(0,1)|}
{|fact(0), rfact(1,1), cfact(2), cfact(3), cfact(4)|}
{|fact(0), rfact(2,2), cfact(3), cfact(4)|}
{|fact(0), rfact(3,6), cfact(4)|}
{|fact(0), rfact(4,24)||}
```

Le résultat final est donné par le terme `rfact(4,24)` restant dans la solution sans aucune transformation supplémentaire.

Un second style d'écriture existe pour la CHAM, plus souvent utilisé pour décrire de nouveaux formalismes de calculs. Au lieu de définir le processus de calcul par un jeu de règles, on peut définir un langage de "programmation" (ou tout au moins de réécriture) qui va être interprété par la machine abstraite. Ce sont ces règles qui vont définir la sémantique opérationnelle du langage concerné. C'est de cette manière que G. Boudol a exprimé différents modèles de calcul parallèle, et c'est ce style d'écriture que nous allons utiliser.

5.3 Notre extension au π-calcul et à la CHAM

5.3.1 Le langage PICOL

Le modèle de calcul choisi est donc le π-calcul auquel vont s'adjoindre quelques extensions syntaxiques (voir l'annexe A) et deux nouveaux opérateurs. Le premier, new va créer une nouvelle solution, et l'opérateur in permet de démarrer un processus dans une solution donnée. En voici la sémantique :

$$\begin{aligned}
\{|in\ \mathtt{g}.P, Q|\}_{\mathtt{a}}, \{|R|\}_{\mathtt{g}} &\rightarrow \{|Q|\}_{\mathtt{a}}, \{|P, R|\}_{\mathtt{g}} \\
M = \{|new\ g.P, Q|\}_{\mathtt{a}} &\rightarrow M' = \{|P, Q|\}_{\mathtt{a}}, \{|\ |\}_{\mathtt{g}} \\
&\quad \text{if } g \notin gn(M) \\
&\rightarrow M' = \{|P, Q|\}_{\mathtt{a}} \text{ if } g \in gn(M)
\end{aligned}$$

5.3.2 La MAAM : Multi-Agent Abstract Machine

Nous définissons une MAAM (Multi-Agent Abstract Machine) comme une sorte particulière de CHAM. Au lieu d'avoir une seule solution dans laquelle se font les réécritures, on définit une MAAM comme un ensemble de solutions étiquetées $\{S_0, .., S_r\}$.

On note alors une solution de nom a :

$$\{|m_1, .., m_k|\}_{\mathtt{a}}$$

On définit également l'état M d'une MAAM comme l'union de l'état de toutes ses solutions. La solution S_0 est appelée la *solution originelle* (ou $Orig$ par souci de brièveté) et supposée accessible par tous les agents. Nous verrons qu'elle joue un rôle assez particulier. Notons également $gn(M)$ l'ensemble de tous les noms de solutions pour un état M donné d'une MAAM. Ces noms de solutions sont supposés uniques.

5.4 Représentation du modèle organisationnel

5.4.1 Expressions de base

Dans un système multi-agent simple, on peut voir un agent comme un ou plusieurs processus déigné par un nom unique. L'important est d'avoir au sein du système un seul canal d'entrée par agent. Plus précisement, le comportement d'un agent va être représenté par un jeu d'équations de la forme

$$A_j(id, v_{1j}, .., v_{kj}) = P_j$$

Avec P_j un processus (au sens du π-calcul) et id une constante qui représente le nom de l'agent. Les v_{ij} seront les paramètres de l'agent, c'est à dire ses variables et attributs internes.

5.4.2 Quelques exemples

Commençons par un exemple simple : un agent demande à un autre de réaliser pour lui une certaine tâche. On supposera que nos deux entités ont été définies par deux structures d'agent `A` et `B` et qu'ils seront désignés par leur nom (au sens de la MAAM que nous avons décrite précédemment).

Notre scénario sera le suivant : après demande de l'agent de type `A`, on suppose que l'agent de type `B` va accepter ou non la tâche t selon une condition `P(t)` dont on posera l'application comme primitive. Après réponse, l'agent de type `A` exécutera alors une primitive $\mathtt{Q}_1$ si la réponse était positive et $\mathtt{Q}_2$ dans le cas contraire.

Voici une description dans la MAAM de ces deux agents :

$$\mathtt{A(x,t,y)} =_{def} \nu\mathtt{(c)}(\overline{\mathtt{y}}\mathtt{(request,t,c).}\\ \mathtt{c(m).((m = agree)?Q_1;}\\ \mathtt{(m = refuse)?Q_2))}$$

$$\mathtt{B(x)} =_{def} \mathtt{x(m,z,k).(m = request)?}\\ \mathtt{(P(z)?}\overline{\mathtt{k}}\mathtt{(agree);}\overline{\mathtt{k}}\mathtt{(refuse))}$$

Examinons maintenant ce que donne l'évaluation de cet exemple dans le modèle. On va considérer comme situation initiale deux agents a et b définis respectivement comme étant de modèle A et B. On supposera que le test P appliqué à la tâche t retournera vrai.

(1) {|A(a,t,b), B(b)|}
(2) {|ν(c)($\overline{\text{b}}$(request,t,c).c(m).. ,
b(m,t,k).(m = request)..|}
(3) {|c(m)((m = accept)?Q_1; (m = refuse)?Q_2)),
(request = request)?
(P(t)?$\overline{\text{c}}$(accept);$\overline{\text{c}}$(refuse)) |}
(4) {|c(m).((m = accept)?Q_1; (m = refuse)?Q_2)),
(P(t)?$\overline{\text{c}}$(accept);$\overline{\text{c}}$(refuse))|}
(5) {|c(m).((m = accept)?Q_1; (m = refuse)?Q_2)),
$\overline{\text{c}}$(accept)|}
(6) {|((accept = accept)?Q_1;
(accept = refuse)?Q_2)|}
(7) {|Q_1|}

Le nom c est ici utilisé un peu comme le "customer" dans la terminologie des langages d'acteurs [Agha et Hewitt, 1987], qui recevra la réponse de l'agent b et désignera donc la continuation du comportement défini par a.

Grâce à l'opérateur de restriction ν(c), le nom c est garanti unique, et de cette manière la réponse ne peut être confondue avec la réponse d'un hypothétique agent x de modèle A car les messages accept et refuse ne sont pas directement gérés par X mais bien par un processus dont le canal d'entrée c est unique.

Remarque 3 *Dans cet exemple, les agents disparaissent une fois que leur action se termine. Dans un scénario plus réaliste, le comportement des agents serait décrit par des équations récursives pour permettre un maintien de leur activité.*

5.4.3 Expression du modèle agent-groupe-rôle

Nous allons donner dans cette section une sémantique opérationnelle du modèle agent-groupe-rôle basée sur le π-calcul et en utilisant une description de type CHAM étendue. Ce modèle utilise les définitions classiques du π-calcul et y rajoute deux opérateurs.

En soi, ce modèle de calcul ne peut pas traduire toute la dynamique de l'organisation telle qu'on l'a exprimée dans le modèle organisationnel. Si nous sommes en mesure d'expliciter la structuration des groupes vus comme un ensemble de solutions, et les rôles comme des aspects architecturaux de l'agent, il nous manque encore l'expression des mécanismes d'entrée et sortie de groupe ou de demande et abandon de rôle. Nous verrons dans la section 5.5 que l'opérationalisation de cette dynamique peut s'exprimer dans le même modèle.

Nous allons maintenant faire le lien entre l'idée d'agent et de groupe que nous venons de formaliser avec le concept de rôle.

5.4.4 Agent et contexte social

En fait, on aboutit à une sorte d'incohérence, ou tout au moins de cercle vicieux. D'une part, le modèle agent-groupe-rôle ne nous autorise pas à expliciter l'architecture interne des agents. Mais d'autre part, si nous ne pouvons pas briser cet aspect unitaire et opaque de l'agent, nous ne pourrons en donner qu'une expression purement *descriptive* dans ce formalisme, ce qui ne serait guère satisfaisant.

La solution consiste en fait à se reposer sur le modèle lui-même : dans la vision agent-groupe-rôle, l'agent est vu comme une entité active agissant dans un certain nombre de groupes selon des rôles qui en structurent les interactions.

Notre hypothèse sera donc de supposer que l'agent établit ses actions en fonction de ses interactions. En revenant au modèle, cela veut dire que l'on envisage les comportements comme étant structurés par les contextes des différents rôles que l'agent va tenir. Nous pouvons alors définir le comportement d'un agent comme l'ensemble des comportements qu'il va associer aux à ses rôles.

Chaque comportement de rôle est donc représenté comme un processus, c'est à dire comme un ensemble d'équations de processus. De ce fait, le comportement d'un agent va être représenté comme un n-tuple $\langle A^0, .., A^n \rangle$ où chaque A^i sera représenté par un jeu d'équations de processus de forme $A^i_j(id, v^i_{1j}, .., v^i_{kj}) = P^i_j$ avec P^i_j dénotant un processus et id une constante désignant l'identité de l'agent, c'est à dire un nom unique qui fédère l'ensemble de ses comportements de rôles.

Un de ces comportement de rôle, A^0, va avoir un sens particulier : c'est le comportement principal de l'agent qui va assurer la cohésion de tous les rôles joués par l'agent. C'est là que la connaissance globale de l'agent sera définie, à savoir toute information ne se réfèrant pas à un des rôles de l'agent. Ce comportement sera appelé le *comportement propre* de l'agent.

Il y a quelques restrictions sur la manière de décrire le comportement d'un agent avec ce modèle. A l'exception du comportement A^0, la contrainte suivante s'applique sur tout comportement interne d'un agent :

Contrainte 1 *Toute équation A^i_j définissant le comportement de rôle i d'un agent ne peut inclure de désignation d'un autre comportement, à l'exception du comportement de rôle i lui-même ou du comportement global A^0.*

Donc, pour tout comportement de rôle tel que $A^i_j(id, v^i_{1j}, .., v^i_{nj}) =_{def} P^i_j$ et pour toute définition A^r_k présente dans P^i_j, $r = i$ où $r = 0$.

En fait, cette contrainte signifie qu'un rôle ne peut invoquer directement un autre rôle. Tous les rôles sont opaques les uns par rapport aux autres au sein d'un même agent, et tout

partage d'information entre deux comportements de rôle transitent par le comportement propre A^0.

Une conséquence de cette modélisation est qu'il faut pouvoir cloisonner les différents comportements de rôle d'un agent. Cette séparation s'exprime naturellement par le placement dans différentes solutions de la MAAM. Il reste aussi à empêcher un agent x de communiquer directement avec le comportement propre d'un agent y, puisque toute interaction s'exprime par rapport aux deux rôles qui la séparent.

La manière la plus générique de résoudre ce dernier point est d'associer un groupe privé (au sens du modèle agent-groupe-rôle), qui sera représentée dans la MAAM par une autre solution étiquetée, et dans laquelle un agent placera son comportement propre à la création. La construction d'un agent ayant le comportement propre $A(a, v_1, .., v_n)$ se fait alors comme suit :

$$\texttt{NewAgent(A)} =_{def} \\ \quad in\ \texttt{Orig}.\nu(\texttt{a}).(new\ \texttt{a}.in\ \texttt{a.A(a,}\texttt{v}_1\texttt{,..,}\texttt{v}_\texttt{n}\texttt{))}$$

Une simple opération de restriction ne serait pas suffisante dans ce cadre, car il convient d'éviter toute réecriture potentielle lors de l'arrivée future d'information via un rôle quelconque qui ne correspondrait pas au comportement du rôle choisi.

Les communications entre le comportement propre et les divers rôles d'un agent sont fait par la primitive in qui permet le transfert d'un processus entre deux solutions. De ce fait, le comportement de rôle d'un agent `a` dans un groupe `g` envoyant une requête `req` avec les arguments $\texttt{a}_1, .., \texttt{a}_\texttt{n}$ avec pour continuation `c` peut être décrit par l'expression suivante :

$$\texttt{SendSelf(a,g,req(}\texttt{e}_1\texttt{,..,}\texttt{e}_\texttt{n}\texttt{),c)} =_{def} \\ \nu\texttt{(c1)}(in\ \texttt{a.}\overline{\texttt{a}}\texttt{(req(}\texttt{e}_1\texttt{,..,}\texttt{e}_\texttt{n}\texttt{),c1).c1(r).}in\ \texttt{g.}\overline{\texttt{c}}\texttt{(r))}$$

Le message `req` est alors encapsulé dans un processus qui est transféré dans le groupe propre (de nom `a`). Quand un résultat est envoyé à la continuation `c1`, l'information est renvoyée dans le groupe `g` et le résultat finalement transmis à la continuation finale `c`.

5.5 Dynamique et réflexivité

Il nous reste à voir comment exprimer au sein du même modèle le contrôle de la dynamique des organisations. Nous allons donc reprendre les opérations primitives de création d'une structure et de contrôle de l'admission d'un agent et voir comment les exprimer dans ce modèle.

5.5.1 Création de groupe

Comme on vient de le voir, il existe un niveau fondamental qui définit les primitives de calculs manipulants les solutions, c'est à dire les groupes dans une MAAM. Au niveau

supérieur, il existe un agent spécifique, primitif (que l'on appelera le *GroupServer*), placé dans le groupe originel et qui sera le seul et unique capable de manipuler cette primitive de création. Toutes les demandes de création d'un groupe devront être transmises à cet agent qui prendra ou non la décision d'agir par une création.

On peut alors demander au *GroupServer* de créer un nouveau groupe de nom `g` et de gestionnaire de groupe `gm`. S'il accepte cette demande, un groupe va être créé (c'est à dire qu'une solution de nom `g` sera mise en place), et le gestionnaire de groupe `gm` exécuté dans le nouveau groupe `g`.

Voici l'expression qui définit le comportement du *GroupServer*. Les noms des groupes sont représentés dans un dictionnaire (cf. annexe A).

```
GroupServer(gnames) =def
   groupServer(msg)
     match(msg)(
       createGroup(g,A^gm,c) →
            ν(c1)(gnames(hasKey(g,c1)).
            c1(r)(r = true)?
               (new g|c(OK)|gnames(atput(g,true)).
                  (GroupServer(gnames)|in g.A^gm)
                 :c(fail)|GroupServer(gnames));
       groupNames(c)→ gnames(keys(c)
                      |GroupServer(gnames);
       isGroup(g,c)→ gnames(at(g,c))
                     |GroupServer(gnames);
```

```
CreateGroup(g,gm,c) =def
   in Orig.groupServer(createGroup(g,gm,c))
```

Le gestionnaire de groupe est l'agent qui gère un groupe, c'est à dire l'agent qui autorise ou refuse l'admission à un autre agent, et connait l'intégralité de l'information organisationnelle du groupe.

5.5.2 Contrôle d'admission

Dans chaque groupe il existe un agent ayant pour rôle la gestion du groupe, à savoir l'autorisation ou le refus d'admission pour un agent. Il connait également tous les membres du groupes ainsi que leur rôle, l'agent effectuant ces tâches aura le rôle de *gestionnaire de groupe*. On peut noter qu'il ne s'agit pas forcément d'un agent isolé ayant pour seul fonction ce contrôle : cette tâche étant définie par un rôle, tout agent peut potentiellement le jouer, à condition qu'il en ait les capacités.

Le gestionnaire de groupe peut être joint dans la solution originelle, c'est à dire la solution dans laquelle tous les comportements globaux sont situés.

Nous allons détailler l'expression de l'entrée dans un groupe pour un agent quelconque.

Pour se joindre à un groupe `g`, un agent `a` demande au `groupServer` le nom de l'agent `gm` gestionnaire du groupe `g`. Ensuite, il demande à `gm` s'il peut entrer dans le groupe avec le rôle `r`. Si la réponse est positive, l'agent `a` peut s'inscrire dans ce groupe et commencer à y agir, en utilisant son comportement de rôle behavior $\mathtt{A}^{\mathtt{r}}_{\mathtt{start}}\mathtt{(a)}$.

```
JoinGroup(a,g,r,A_0^r,c) =def in Orig.ν(c1)(
  groupServer(isGroup(g,c1)).
       c1(b).(b = true)?
          ν(c2)((EnterGroup(a,g,r,A_0^r,c2)).
            c2(r).in a.c(r)) :
          in a.c(nogroup))

EnterGroup(a,g,r,A_0^r,c) =def in Orig.ν(c1)(
  ν(c1) in g.groupManager(check(a,r,..,c1)).c1(r)
    match(r)
      refuse() → c(refused)
      agree(s) → in s.A_0^r | c(succeed))
```

5.5.3 Un exemple

Nous allons décrire dans cette section un exemple simple mais complet de création et d'admission dans un groupe. Imaginons deux agents voulant jouer au ping-pong en se transmettant un message 5 fois. Pour ce faire, l'agent "Ping-Pong" va chercher un groupe `pingpong-group`. S'il existe, il va demander son admission avec le rôle `j` de joueur, et attendre le premier message. Si ce n'est pas le cas, il créera le groupe, y entrera et sera prêt à envoyer le premier message.

Le comportement global d'un tel agent pourrait par exemple être :

```
PingPong^0(a) =def
  ν(c1).JoinGroup(a,pingpong-group,p,PingPong^p_init(a),c1).
    c1(r1).(r1=nogroup)?
      ν(c2).CreateGroup(pingpong-group,PingPong^gm(a),c2).
      c2(r2).(r2=success)?
        JoinGroup(a,pingpong-group,p,PingPong^p_play(a),a)
```

Les deux définitions de processus pour le comportement de rôle `p` seraient pour le groupe `pingpong-group` :

```
PingPong^c_init(a) =def
  ν(b).groupmanager(getAgentsWithRole(p,b)).
    b̄(5,a).PingPong^p_play(a)
```

$\texttt{PingPong}^{p}_{play}\texttt{(a)} =_{def}$
$\texttt{a(m,b).(m} \neq \texttt{0)?(}\overline{\texttt{b}}\texttt{(m-1,a).PingPong}^{p}_{play}\texttt{(a))}$

Pour ne pas compliquer abusivement cet exemple, nous supposerons ici que le rôle est toujours accordé :

$\texttt{PingPong}^{gm}\texttt{(a)} =_{def}$
```
groupManager(msg)
  match(msg)
    check(a,g,r,c) →
      c̄(agree(g))| roles̄(atput(r,a))
    getRoles → roles̄(keys(roles));
    getAgentsWithRole(r)→
      roles̄(at(r,c))
```

5.6 Conclusion

On a vu qu'il est possible de donner une sémantique opérationnelle précise du modèle organisationnel agent-groupe-rôle et de ses opérations primitives à l'aide du π-calcul et d'une légère extension à la CHAM. De plus, le processus même de contrôle d'organisation peut être décrit au niveau meta, si l'on fait l'hypothèse d'agents contrôlant les groupes.

Est-ce à dire que l'on va pouvoir exprimer une sémantique forte des systèmes multi-agents à partir de la sémantique opérationnelle que nous avons définie ici ? Ce serait un peu audacieux, car pour en avoir une expression complète, il est nécessaire d'établir un lien complet avec les définitions de comportements de rôles, par exemple en ayant une définition complète en π-calcul ou tout autre formalisme pouvant s'y ramener.

Néanmoins, certaines pistes sont ouvertes et prometteuses dans ce cadre, en particulier dans les relations à faire avec les modèles formels d'agents componentiels tels que ceux décrits par [Brazier et al., 1997], qui à première vue apparaissent assez complémentaires. Le lecteur se référera à [Ferber et al., 2000] pour un premier travail dans cette voie.

Chapitre 6

Aspects implantatoires : la plate-forme MADKIT

Nous avons présenté dans la partie précédente un mécanisme de calcul et une sémantique de notre modèle social d'agents. Dans cette partie, nous allons reprendre l'étude de l'opérationalisation, mais cette fois-ci d'un point de vue implantatoire.

Pour vérifier l'intérêt pratique de notre proposition il nous fallait en étudier en détail l'utilisation dans une gamme large d'applications. La première étape a donc consisté à concevoir une architecture rendant possible son utilisation dans plusieurs modèles d'agents, et de ce fait implémenter une plate-forme généraliste d'exécution de systèmes multi-agents.

Nous commencerons par faire un tour d'horizon des plates-formes multi-agents et verrons pourquoi ce travail a conduit à l'implémentation d'un nouvel environnement.

Nous examinerons ensuite plus en détail l'architecture de notre plate-forme, MADKIT, tant au niveau de sa construction interne que des modèles qu'elle induit pour les agents qu'elle exécute. Nous évoquerons également certaines conséquences sur les domaines applicatifs possibles de cet outil.

6.1 Les plate-formes multi-agents

Depuis plusieurs années, on a vu se multiplier les plate-formes de développement agents. Sous ce terme illusoirement générique se cachent en fait plusieurs grandes catégories de plate-formes. Même s'il n'est pas question de voir ici en détail l'ensemble de ces plate-formes (ou même de ces catégories), nous allons en parcourir quelques exemples et noter les traits marquants, ce qui nous permettra de mieux mettre en évidence les choix faits sur MADKIT, et leur justification.

6.1.1 Catégorisation

Les plate-formes d'agent mobile Elles se situent souvent dans la filiation du précurseur TELESCRIPT [White, 1996], et présentent parfois une confusion entre "agent mobile" et "code mobile"

Un exemple récent de plate-forme d'agents mobiles est la plate-forme Aglets, développée par IBM Japon. On y retrouve les mécanismes de base de la migration : un identifiant unique, un itinéraire qui décrit les différents noeuds à visiter et les actions à prendre à chaque étape. Cela impose évidemment la mise en place d'une plate-forme Aglet sur les noeuds désirés du réseau, car la migration n'est possible que si l'infrastructure est présente. Un outil de conception permet de spécifier les contraintes de sécurité à appliquer à chaque aglet sur chaque site. La communication entre aglets sur les sites se fait soit par un tableau noir, soit par passage de messages synchrones ou asynchrones.

Il faut bien dire que les fonctionnalités que propose Aglets sont plus à rattacher au domaine des objets mobiles que véritablement des agents. En effet, les aspects d'autonomie d'action ou d'interaction sont réduits à leur plus simple expression.

Les outils de la simulation multi-agents Ce sont souvent des outils construits de façon ad-hoc pour un domaine de simulation, ou même une seule application [Ginot et Le Page, 1998] [Coen, 1994].

SWARM [Burkhart, 1997] est une exception notable de plate-forme à vocation générique. Dans ce cas, l'outil propose un modèle suffisamment générique de simulation (basé sur une décomposition récursive des systèmes) sur lequel on peut bâtir éventuellement des modèles spécifiques. En fait, la force de cette plate-forme réside plus dans l'importance des outils annexes : obtention des données par des sondes que l'on peut positionner sur quasiment tous les points de l'application, traitement et première analyse par des outils statistiques, visualisation en temps réel des paramètres, lancement de jeux d'expériences. Le modèle d'agent est peut-être paradoxalement la partie la plus faible de l'outil.

Les plate-formes orientées modèle Sous ce qualificatif se rangent des plate-formes à vocation généraliste du point de vue applicatif, mais développées par rapport à un modèle très spécifique d'agent dont elles permettent en fait la validation pratique.

AgentBuilder [Reticular Systems, 2000] appartient à cette catégorie, et réunit à la fois un outil de conception et une plate-forme d'exécution. Le modèle des agents AgentBuilder dérive du modèle Agent0 [Shoham, 1991], la communication s'effectue par messages KQML. L'architecture laisse néanmoins le concepteur libre de rajouter de nouveaux actes de langages pour s'adapter à un domaine particulier. L'outil de conception est extrêmement ciblé sur le type de modèle défini et couvre l'intégralité du processus de développement. Le concepteur a la possibilité de définir ses schémas d'interaction, ses structures sociales,

ses mécanismes de tâches et de réactions aux messages, etc. Néanmoins, toute construction est strictement limitée aux modèles définis et des variations sont très diffficiles à mettre en œuvre. L'autre partie de l'outil est un moteur d'exécution, qui est en fait un interprète pour les définitions d'agents générées par l'outil de conception.

On retrouve ici un des traits caractéristiques des systèmes construits sur un modèle d'interaction KQML : on se sert de la spécification KQML comme d'une base pratique pour avoir un consensus suffisant sur la sémantique des différents actes, mais on s'autorise des libertés avec l'usage et l'extension des messages. Il s'agit bien de faciliter la vie du concepteur de système plutôt que d'essayer de faire intéropérer des agents d'applications différentes, sans parler d'intégration entre plate-formes.

Les extensions d'architectures classiques Certaines infrastructures ont aussi été désignées comme des technologies de choix pour l'implémentation d'applications multi-agents (tels que CORBA [OMG, 1992] ou Jini [Arnold et al., 1999]). Il est vrai que dans certains cas, pratiquement rien ne manque, hormis une sémantique appropriée. Voyager est un exemple d'infrastructure "mixte" : c'est à la base un bus objet (ORB) écrit en Java, mais étendu par quelques fonctionnalités intéressantes de mobilité d'agent. Voyager permet au concepteur d'écrire des agents, en sachant que la mobilité est fournie au niveau de l'infrastructure, avec quelques autres services. Par exemple, Voyager peut mettre en place automatiquement des adresses permettant de faire suivre les messages au fur et à mesure du déplacement de l'agent. La communication peut être synchrone (et c'est une simple invocation de méthode) ou asynchrone, via des boîtes aux lettres éventuellement distribuées.

Par contre, tout comme Aglets, Voyager repose essentiellement sur les mécanismes du langage d'implémentation (Java) en particulier au niveau sérialisation et gestion fine de la sécurité, mais n'utilise pas vraiment un modèle de comportement riche ou tout au moins un niveau de description authentiquement agent.

6.1.2 La standardisation FIPA

Une approche très différente, par rapport aux grandes catégories de plate-formes que l'on a parcourues, est proposée par un consortium universitaire et industriel, la FIPA (*Foundation for Intelligent Physical Agents*).

Il s'agit d'un effort d'intégration et d'intéropérabilité entre applications multi-agents passant par l'écriture de spécifications, pour *normaliser* rapidement les technologies agents. Comme ce souci de généricité et d'intégration est aussi le nôtre, nous allons explorer plus en détail cette proposition.

Un ensemble de spécifications

On peut dégager deux grands ensembles de spécifications “normatives” dans FIPA : celles ayant traits aux plate-formes qui vont héberger les agents, et celles décrivant le langage d’interaction ACL (Agent Communication Language).

Le but est bien de mettre en place un “middleware” agent qui fournira des services génériques d’accueil, d’identification, et de communication entre agents FIPA. Cet ensemble forme la plate-forme de référence de FIPA (voir figure 6.1). Dans FIPA, ces services de base peuvent être implémentés eux-mêmes sous la forme d’agents conformes à FIPA. Ces services sont :

- L’AMS (Agent Management System) : c’est un peu le cœur d’une plate-forme FIPA. Il enregistre les agents actifs, gère leur identité et garde trace de leur état.
- Le DF (Directory Facilitator) est un service d’annuaire permettant d’identifier les services utilisateurs sur une plate-forme.
- L’ACC (Agent Communication Channel) est un agent particulier chargé de faire le lien entre le mécanisme de communication natif (et potentiellement non-FIPA) de la plate-forme et les autres agents et plate-formes FIPA, éventuellement distantes.

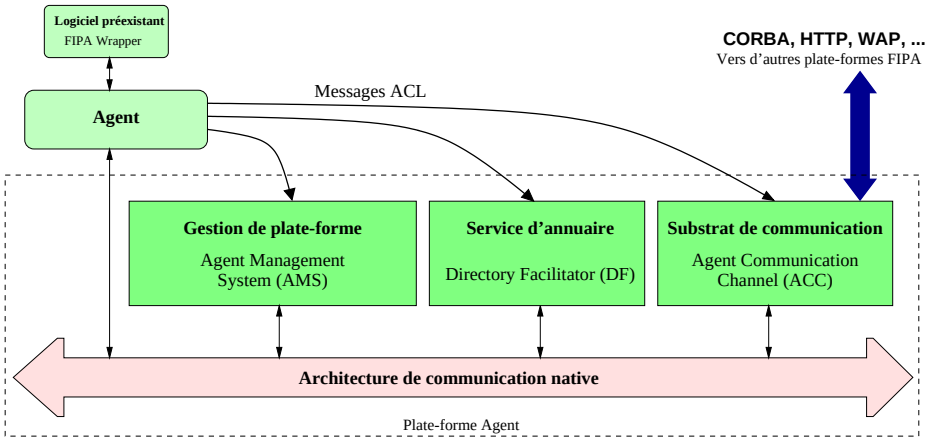


FIG. 6.1 – Architecture de référence FIPA

Il faut noter qu’à l’heure actuelle, un débat a lieu au sein de FIPA sur le sujet de “l’agentification” de ces trois services. Initialement - dans les spécifications FIPA’97[Steiner, 1997] -, ces trois services étaient des agents à part entière, au sens FIPA. Ils communiquaient en FIPA-ACL et étaient des intermédiaires obligés. Par exemple, pour envoyer un message d’un agent FIPA sur une plate-forme à un autre sur une plate-forme distante, il faut déjà faire suivre le message à l’ACC local. Celui-ci transmettrait alors son message à son homologue.

Ce mécanisme a été critiqué pour son inefficacité et sa lourdeur : une interaction ACL n’étant pas quelque chose d’anodin du point de vue traitement, on double quasiment toutes les interactions par des interactions contraignantes pour des tâches purement administra-

tives. Voici par exemple le message nécessaire pour envoyer un message d'un agent FIPA `agent-a` à un autre `agent-b` si l'on observe scrupuleusement la norme :

```
(request
    :sender agent-a
    :receiver acc
    :content (action acc
                (forward
                    (request
                        :sender   agent-a
                        :receiver agent-b
                        :content
                        ......
        )))
    :protocol fipa-request
    :language sl0)
```

L'approche actuelle est donc de considérer l'ACC comme un service non-nécessairement agent, qui peut être décrit en termes d'objet à l'usage de l'AMS ou le DF local et qui puisse éviter le déploiement d'une architecture de traitement lourde pour ce service de base. On y perd un mécanisme d'interaction unifié et extrêmement générique - FIPA-ACL - mais on facilite l'intégration potentielle avec des infrastructures de communication pré-existantes et on allège un peu les plate-formes. La mise en oeuvre est bien plus directe, mais l'interaction avec ces services se fait toujours via un protocole d'interaction ACL.

On peut quand même s'inquiéter de ces dérives par rapport à l'idée originelle de FIPA, qui avait le mérite de la cohérence du tout-agent.

L'autre élément clé des implémentations FIPA est ACL. ACL a été dès le départ vanté comme étant un "meilleur KQML", car formellement défini, plus facilement validable, et doté de protocoles d'interactions [Sadek, 1991]. Cela a été remis en question par la communauté KQML assez rapidement [Labrou et Finin, 1997], qui a ensuite proposé une sémantique formelle pour KQML. Après les premières expériences, il semblerait que les espoirs mis dans ACL ne sont pas tous vérifiés [Pitt et Mandani, 1999].

Pour terminer sur ce bref aperçu de la communauté FIPA, il faut signaler les changements de mode opératoire et de cible que connaît cet organisme. Dans les premiers temps (de 1996 à 1998), des spécifications étaient établies chaque année, puis éventuellement amendées l'année suivante. Depuis 2000, le processus est plus itératif, et se rapproche du fonctionnement d'autres organismes comme l'IETF [Bradner, 1996] : des spécifications préliminaires sont établies, testées en vraie grandeur, puis seulement adoptées comme standards, ou bien automatiquement considérées obsolètes si rien n'arrive dans un certain délai. Cela ouvre la voie à l'expérimentation et laisse un plus grand champ d'action à la FIPA. On pourra comparer les soixante-dix spécifications en cours actuellement aux sept de 1997 et cinq de 1998.

De façon comparable, l'architecture FIPA a été affinée récemment pour aller dans le sens d'une plus grande modularisation. Nous l'avons vu avec la levée de la contrainte d' "agen-

tification" de l'ACC. De même, CORBA/IIOP n'est plus un pré-requis absolu, en ces temps de téléphones mobiles et protocoles WAP. L'encodage des messages peut être fait en s'appuyant sur d'autres formats que l'ASCII pur, pour ouvrir la voie à XML, Unicode ou à des formats compacts. Tout ceci va dans le sens d'une plus grande variabilité des domaines de déploiement des applications FIPA, même si le modèle de base de l'agent reste inchangé et conditionne en fait fortement le style d'application.

La plate-forme Emorphia FIPA-OS

Pour illustrer comment ces spécifications peuvent se concrétiser, nous prenons l'exemple d'une des implémentations existantes : FIPA-OS, une plate-forme de validation des spécifications FIPA élaborée sous la houlette des laboratoires de Nortel Networks. En particulier, elle a été le premier outil à mettre correctement en œuvre les parties obligatoires des spécifications. Elle a été clairement positionnée comme un moyen d'accroître la visibilité et l'utilisation de FIPA.

Son architecture est un modèle en trois couches : transport, modèle et agent. Elle permet d'accueillir plusieurs modèles d'agents à partir du moment où le modèle de base de FIPA est respecté (6.2). Ces agents utiliseront implicitement les services FIPA classiques.

Des éléments de découplage sont néanmoins présents dans les couches de transport, qui ne sont pas forcément restreintes à CORBA/IIOP et aux mécanismes de persistance d'agents.

Les modèles d'agents de bases sont conçus pour accueillir directement des interactions conformes à ce qu'attend FIPA. C'est en particulier le respect d'ACL et des protocoles d'interaction qui y sont associés, mais également l'utilisation des langages de contenus définis par FIPA : SL et CCL[1].

6.1.3 Analyse et discussion

Le plus souvent, la plate-forme se confond avec son application : recherche d'information pour Infosleuth [Nodine, 1998], équilibrage de charge sur Challenger [Chavez et al., 1997], même si la volonté de généricité est affirmée, il est souvent difficile d'évaluer ce point hors d'un corpus d'applications conséquent. D'autre part, les environnements de conception, développement, et exécution sont définis dans la majorité des cas de façon rigide, alors qu'il serait plus intéressant d'avoir un découplage et une modularisation de ces différents aspects pour permettre de spécialiser les outils en fonction du domaine ou des conditions de déploiement (environnement de test, serveur aveugle, ...)

De plus, ces plate-formes posent souvent un problème d'outillage des applications : on ne dispose que rarement d'outils d'analyse du fonctionnement des agents ou de la plate-

[1] L'originalité de CCL [Willmott et al., 1999], par rapport à des spécifications comme KIF, est son expression sous la forme d'un CSP.

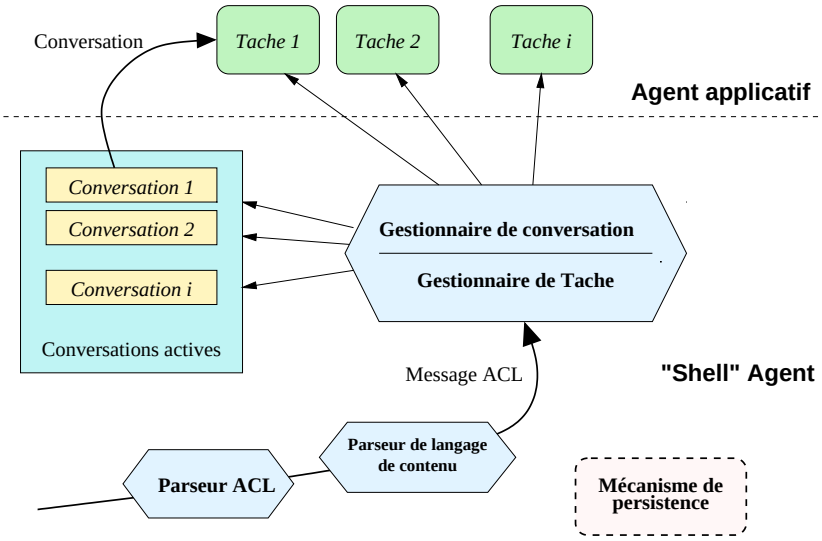


FIG. 6.2 – Architecture d'un agent classique dans FIPA-OS

forme, et lorsque c'est le cas, il n'est pas réellement possible de construire des mécanismes d'observation spécifique sans avoir à intervenir sur le cœur de la plate-forme.

Mais le principal reproche que l'on pourrait formuler à l'égard des plate-formes multi-agents est leur spécialisation : la conception d'agent est centrée sur un modèle particulier, ce qui induit forcément une homogénéité sur les applications qui peuvent y être hébergées. Certaines plate-formes échappent néanmoins à cet écueil, comme DIMA[Guessoum, 1996], Geamas[Marcenac et Courdier, 1999] ou Swarm[Burkhart, 1997] qui reposent volontairement sur des modèles plus génériques.

La contrainte d'un modèle rigidifié d'agent entraîne souvent un écart par rapport aux choix initiaux du concepteur. Cela pourrait être anodin dans le cas d'un domaine établi, mais cela s'avère particulièrement gênant dans le cas des systèmes multi-agents où les architectures, les modèles, ou même les définitions de base ne sont toujours pas véritablement affirmées.

Même dans le cas de FIPA, qui essaie pourtant d'aboutir à une interopérabilité entre systèmes (via des spécifications), on se rend compte que l'architecture des systèmes est en fait contrainte par deux fois. D'une part le modèle de référence d'un agent et de son langage de communication oriente très fortement le concepteur vers un style particulier d'architecture (par exemple, exprimer un système en vue d'une résolution par émergence est très malaisé). D'autre part, l'implémentation particulière de la plate-forme en elle-même va rajouter un modèle de comportement spécifique. Le développeur d'agent n'a d'autre choix que de s'y plier, et d'évaluer soigneusement l'adéquation entre son problème et l'architecture qu'il ne pourra guère adapter à ses besoins.

Pour finir, on notera qu'un des problèmes majeurs du monde des plate-formes agents

n'est peut-être ni technique, ni conceptuel, mais plutôt lié au contexte. L'atomisation du travail sur les plate-formes et la très grande spécialisation des modèles n'encouragent guère l'émergence d'une communauté d'utilisateurs. Comme l'a bien mis en évidence [Gasser, 2000], les faibles possibilités de partage de modèles, d'agents, ou d'expérience entre concepteurs freinent la généralisation des applications multi-agents, une pédagogie de ces systèmes, et l'apparition d'infrastructures communes.

6.2 Motivations et besoins

Nous allons maintenant prendre le point de vue de l'implémenteur et présenter l'architecture de la plate-forme que nous avons réalisée pour valider notre approche, MADKIT.

6.2.1 Contexte

Imaginons que nous ayons à concevoir une plate-forme multi-agents générique. La première question que nous nous poserons sera : *quelles sont donc les fonctionnalités dont j'ai besoin pour faire fonctionner les agents que je veux construire ?* (ce qui est a priori la partie intéressante de la tâche).

On pourrait alors rapidement établir une première liste de fonctionnalités classiques : communication par message, moteur de simulation multi-agents, modèle d'environnement, définition de protocoles d'interaction , gestion du cycle de vie , mobilité d'agent , identifications uniques , observation des systèmes multi-agents en activité , agents à base de règles , exécution concurrente ou distribuée , sécurité et contrôle des systèmes multi-agents, ...

Bref, la liste est longue et décourageante. Mais, réflexion faite, un raffinement de notre question un peu naïve pourrait être : *quelles sont les fonctions **primitives** d'une plate-forme grâce auxquelles les autres fonctionnalités courantes peuvent être ré-implémentées ?*

6.2.2 Principes et choix initiaux

Nous en sommes donc arrivés au cahier des charges suivant :

Base de communication : fournir le minimum de ce que l'on peut attendre d'un système de communication asynchrone. Cela comprend pour tout agent l'envoi et la réception de messages via une boîte aux lettres, la garantie de transmission ordonnée et de délivrance en cas de présence du récepteur sur le système.

Gestion de l'identité et du cycle de vie : permettre d'identifier de façon unique des agents, les instancier (en tant qu'agent et non simplement objet) et les suivre jusqu'à disparition.

Politique d'exécution : définir quelques politiques d'exécution de base (séquencées, parallèles) sur lesquelles bâtir des modèles d'exécution (influence/réaction, comportements concurrents, ...)

Structuration applicative : donner des moyens au concepteur final de faire le lien entre l'organisation de son application et les détails terre-à-terre de catégorisation de ses agents.

Les conséquences pratiques des trois premières contraintes sont très directes et évoquent chez l'implémenteur quelques questions intéressantes : exécution concurrente, implémentation d'un schéma de nommage, mécanisme de routage de messages. Tout ceci est nécessaire, mais cela reste assez terne vis à vis d'autres systèmes. Nous allons donc préciser le dernier point :

Maintien d'une vue organisationnelle : l'ensemble des agents définis sur la plate-forme a accès aux notions de groupes et de rôles. Tout agent peut participer à un schéma organisationnel qu'il peut interroger et avec lequel il peut interagir.

C'est en fait là que réside le choix radical de notre plate-forme : les informations relatives au modèle social ne seront pas mises en place au niveau de l'agent. Au lieu d'avoir des représentations *internes* -modèles de connaissances ou d'accointances ou simples perceptions de l'environnement organisationnel-, on a une information *purement extrinsèque* mais accessible des groupes et des rôles.

Bien sûr, ces informations sont (relativement) sommaires : on pourrait remarquer qu'il ne s'agit que d'un étiquetage d'agents, certes, mais situé à un niveau peu courant puisque les informations sociales sont habituellement considérées comme relevant d'un très haut niveau de description [Castelfranchi, 1995]. Classiquement, elles relèvent de la connaissance générale de l'agent sur son environnement ou bien de choix d'architectures et de construction, apparaissant "en creux" dans les implémentation par les schémas d'interactions ou les mécanismes d'identification ou de communications et perdues dans le système final[2].

Notre réflexion a été la suivante :

- L'architecture individuelle d'un agent est un choix fort fait par le concepteur. C'est elle qui va définir au minimum un modèle d'action et éventuellement des modèles de raisonnement, perception et de symbolisation, il faut laisser ce choix, et ne pas forcément séparer agents "cognitifs" et agents "réactifs".
- L'activité collective est le point singulier de ces modèles. Figer le *modèle de structuration* (et non la structure) d'une société d'agents empêche la réintégration ultérieure d'autres entités et modèles.

[2]On peut noter que l'utilisation de groupes d'agents pour la structuration de plate-formes a été proposée dans d'autres outils, comme [Baumann et Radouniklis, 1997], bien que ces mécanismes soient spécifiques au domaine (migration d'agent), et reposent sur un modèle plus contraignant (sans la possibilité de groupes et rôles simultanés pour un agent donné)

– La plate-forme doit s'effacer devant ses usages. Vu la richesse des domaines applicatifs des modèles agents, il est souhaitable de définir des possibilités de modularisation de l'infrastructure plutôt qu'une plate-forme monolithique.

Ces principes d'architecture pour la plate-forme vont au-delà du simple modèle agent-groupe-rôle. Notons finalement qu'il reste tout à fait envisageable (cela a d'ailleurs été réalisé dans au moins un système à notre connaissance) d'implémenter une structuration de type agent-groupe-rôle dans une plate-forme autre que MADKIT.

6.2.3 Filiations

MADKIT n'a bien évidemment pas surgi ex nihilo. Il nous paraît donc intéressant de mentionner diverses influences ou filiations sur notre système (lesquelles ne sont d'ailleurs pas toutes dans le "monde agent").

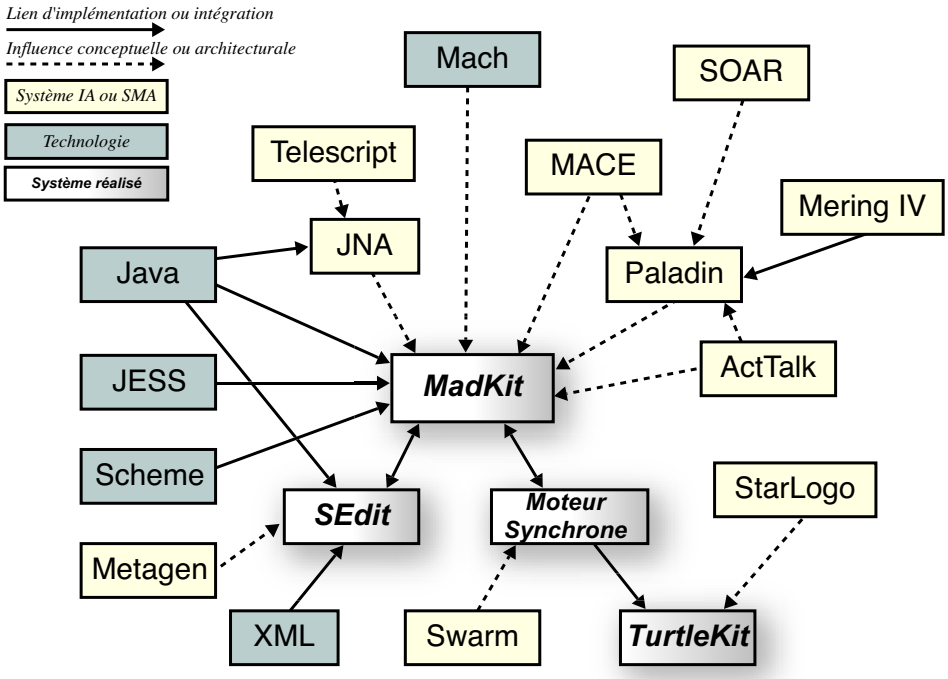


FIG. 6.3 – L'arbre de famille de MADKIT

De MACE [Gasser et al., 1987], on retrouve bien évidemment l'idée de rôle, présente autant comme élément conceptuel du modèle de description que de stratégie d'implémentation. MACE a cette particularité de définir des modèles de connaissances de haut niveau axés directement sur la représentation de l'organisation. Les accointances d'un agent sont désignées explicitement au niveau organisationnel en termes de nom, de classe ou d'adresse,

mais également en termes de rôle, capacité, but ou plan.

JNA (Java Net Agents) [Merlat, 1999] a elle-même un héritage assez net des concepts de TeleScript[White, 1996], avec les notions de lieux d'exécution et de "tickets" pour contrôler les caractéristiques d'un agent. C'est aussi face à un certain manque de souplesse de JNA et TeleScript sur une application qu'est venue l'idée de découpler le plus possible le noyau d'une plate-forme des agents et de l'interface. On peut trouver trace des agents de contrôle (l'agent "concierge") dans l'idée de services systématiquement agentifiés de MADKIT.

De PALADIN [Ferber, 1989] et de certaines de ses influences : on trouve ici l'accent sur la flexibilité de la plate-forme, l'idée d'une plate-forme d'implémentation par et pour des agents. MADKIT se démarque de PALADIN par le choix fait de n'aborder l'agent que comme un certain niveau d'abstraction (au dessus d'un modèle objet ou fonctionnel) et non comme une intégration complète dans le modèle d'exécution lui-même[3]

Java et Scheme. Le choix de Java est venu très rapidement. Parmi les objectifs initiaux de MADKIT, la généricité et la souplesse étaient en tête. Bien sûr, la portabilité nous permet de nous abstraire des machines physiques. Néanmoins, d'autres détails ont fait pencher la balance, comme l'intégration des processus de sérialisation.

La présence de Scheme se justifie par l'interactivité et la souplesse de développement : il est nettement plus confortable de pouvoir prototyper un agent sur le moment que de repasser par un cycle de compilation/empaquetage/chargement.

Les systèmes d'exploitation à micro-noyaux Le nom même de "micro-noyau" agent que nous retrouverons plus tard est un clin d'œil transparent à ces architectures, qui furent une inspiration pour la structuration de la plate-forme. En appliquant leur principe fondateur [Rashid et al., 1989] *'incorporating in micro-kernels a small number of key facilities that allow the efficient deployment of system services."* aux architectures agents, on aura une idée de la transposition faite ici.

De MetaGen. La recherche sur les environnements de meta modélisation et meta programmation est également une référence, avec en particulier le système MetaGen[Revault, 1996] dont des traits se retrouvent dans le mécanisme d'extension de MADKIT et surtout l'éditeur multi-modèles SEdit (que nous présenterons brièvement chapitre 8).

[3] ce qui a aussi peut-être permis que MADKIT soit au final implémenté, au contraire de PALADIN

6.3 Architecture

6.3.1 Principes

La structure organisationnelle est donc implémentée au cœur de la plate-forme MADKIT. Notons tout d'abord que cela sert autant à fournir un modèle organisationnel aux systèmes multi-agents exécutés que pour le fonctionnement interne du système.

En plus de ce modèle d'organisation, MADKIT est basé sur trois principes :
– Architecture à micro-noyau,
– Agentification systématique des services,
– Découplage fonctionnel entre noyau, agents et application d'accueil.

Concrètement, MADKIT est un ensemble de packages Java qui implémentent le noyau agent, diverses bibliothèques de base de messages, d'agents et de sondes.

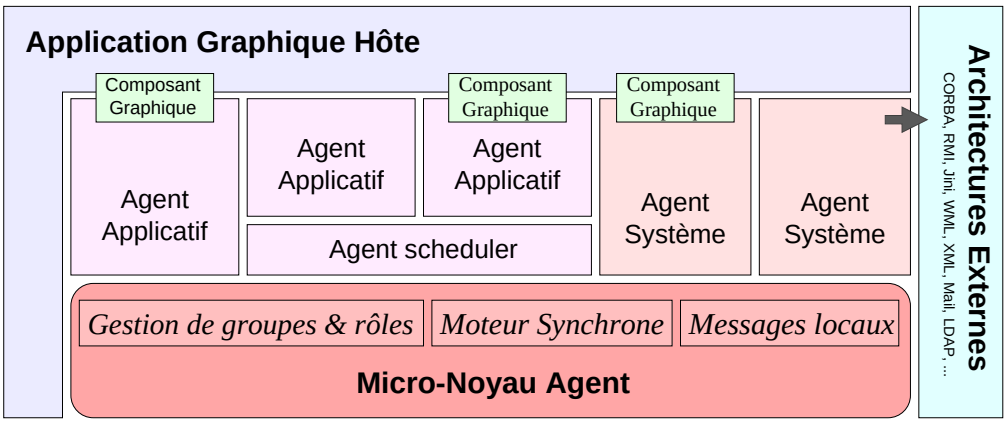


FIG. 6.4 – Structure générale

Le principe essentiel de MADKIT est d'utiliser partout où cela est possible la plate-forme pour son propre fonctionnement. Tous les services hors ceux assurés par le micro-noyau sont implémentés par des agents à part entière. Ceci vient à la fois d'un souci d'unicité de modèle, mais également de mise à l'épreuve des systèmes multi-agents comme modèle de programmation.

Cela implique que MADKIT n'est pas une plate-forme agent dans le sens classique, centrée autour d'un modèle particulier d'agent ou d'interaction. La taille réduite et les fonctionnalités du noyau de base, la neutralité des types agents, associée à ce principe de services agentifiés et de découplage par rapport aux applications "hôtes" permet en fait d'obtenir toute une gamme d'environnements d'exécutions aux finalités parfois complètement opposées.

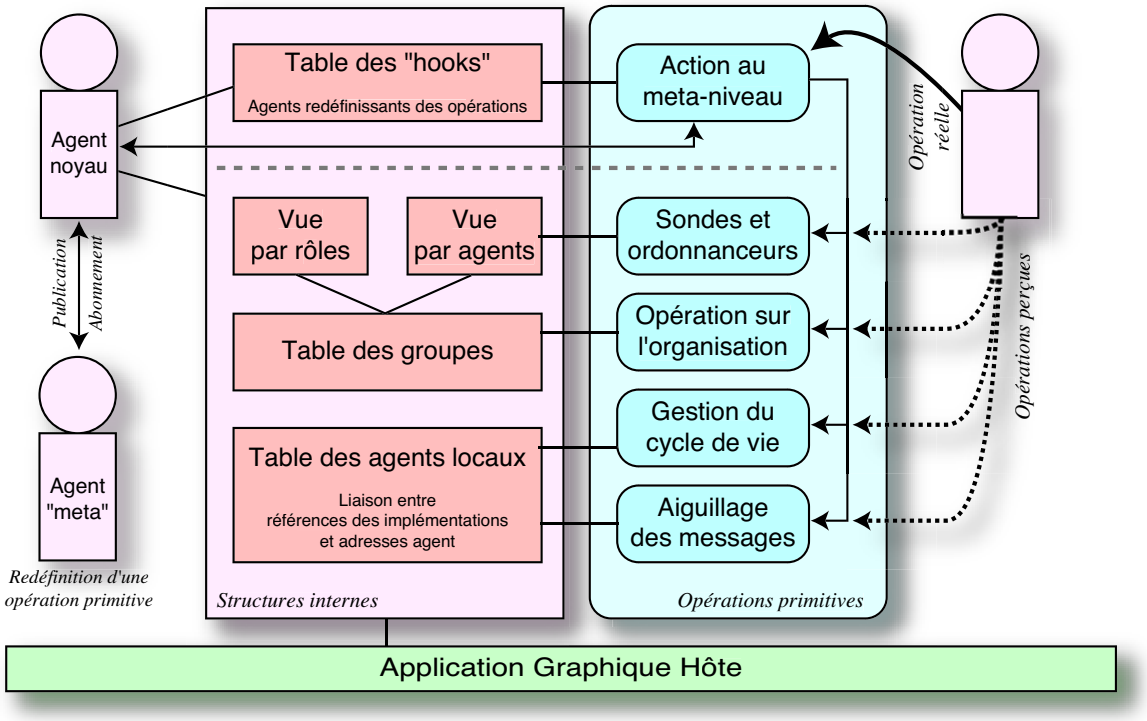


FIG. 6.5 – Architecture fonctionnelle

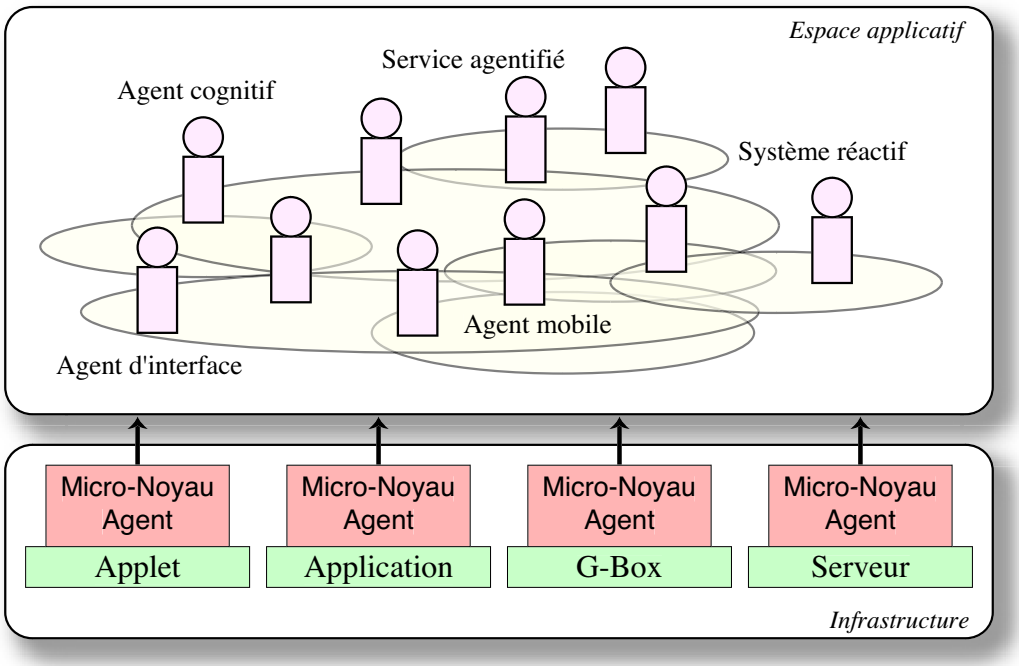


FIG. 6.6 – Schéma général de fonctionnement

6.3.2 Le micro-noyau agent

Le "micro-noyau" de MADKIT est un environnement d'exécution d'agents de taille réduite (moins de 40 Ko).

Ce module ne prend en charge que les fonctions suivantes :

Gestion des groupes et rôles locaux. Comme l'interopérabilité et les mécanismes d'extension de MADKIT se basent sur le modèle agent-groupe-rôle, il est essentiel que cette information organisationnelle soit gérée au plus bas niveau afin que tous les agents, quels que soient leurs modèles individuels, y aient accès. Le noyau a la responsabilité de maintenir une information correcte sur les membres des groupes et les rôles tenus. Il vérifie également si les requêtes faites sur le système de groupes et rôles sont acceptables (c'est à dire en évaluant ou délégant les fonctions d'acceptation de rôles).

Gestion du cycle de vie des agents. Le noyau gère également le lancement (et éventuellement l'arret) des agents et maintient les tables de références sur les objets d'implémentation. Il est également le gestionnaire des informations administratives sur les agents (possesseur, modalités de créations, ...) et donne un identifiant garanti unique à chaque agent. Cet identifiant, l'`AgentAddress`, est forgé à partir de l'adresse du noyau MADKIT (qui tient donc lieu d'espace de nommage) et l'identification de l'agent sur le noyau ; il peut être rapproché de la notion de GUID de FIPA. La forme de cet identifiant peut également être redéfinie pour faciliter l'intégration avec d'autres plateformes agent.

Passage de message local. Le noyau a la responsabilité de l'aiguillage et de la distribution de messages entre agents uniquement **locaux** (s'exécutant sur le noyau). Le passage de message au plus bas niveau revient à de simples échanges de références pour pouvoir facilement implémenter différentes sémantiques de passage de message à un niveau supérieur.

Observation et exécution. Deux politiques d'exécution de base sont définies à ce niveau : l'exécution concurrente classique et un framework d'ordonnancement synchrone. L'instrumentation par un jeu de sondes d'un système multi-agents en exécution se fait également à ce niveau.

6.3.3 Structure et fonctions d'un agent

La classe de base d'un agent MADKIT (`AbstractAgent`), définit quelques fonctionnalités de base pouvant être nécessaires dans les modèles classiques.

Fonctionnalités

Les fonctions associées à tout agent sont :

Cycle de vie. L'agent dispose de quatre états (création, activation, exécution, et destruction), et a la possibilité de démarrer d'autres agents sur le noyau local (et de les désactiver par la suite). Par contre, aucun mécanisme concret d'exécution n'est défini à ce niveau.

Communication. La communication est implémentée sous forme de passage de message asynchrone[4], soit d'agent à agent identifiés par leur `AgentAddress` ou leur groupe et rôle, soit sous la forme d'une diffusion à tous les teneurs d'un rôle dans un groupe donné.

Organisation. Tout agent dispose de primitives permettant d'observer son organisation locale (connaître les groupes et rôles courants) et d'y agir (prise de rôle, entrée et retraits de groupes).

Outils. La classe de base des agents permet également de manipuler une éventuelle interface graphique associée à l'agent, les flots d'entrée/sortie, etc.

Messages

Les messages sont définis par héritage à partir d'une classe de base `Message` qui ne définit que la notion d'émetteur et destinataire. Une bibliothèque de messages de base (voir figure 6.7) est néanmoins fournie et permet l'envoi de chaînes, d'objets sérialisés, de documents XML, ou bien de messages conformes aux spécifications KQML et FIPA-ACL. Ils restent extensibles pour s'adapter à tout protocole d'interaction.

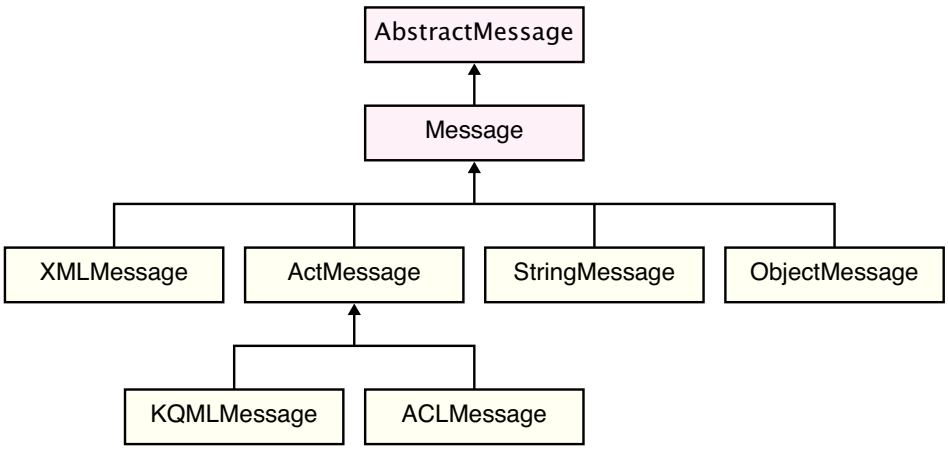


FIG. 6.7 – Hiérarchie des messages standards

La sémantique du passage de message de MADKIT a un trait un peu particulier : toute manipulation de message par le noyau n'entraîne aucune recopie. Cela signifie qu'un message construit et envoyé via `sendMessage` par un agent ne sera pas copié au moment de

[4]Si l'on se place dans le cas d'agents dont on contrôle les politiques d'ordonnancement, comme nous le verrons plus loin, on peut utiliser le mécanisme sous-jacent de manière synchrone

sa délivrance dans la boîte au lettre du destinataire, et que celui-ci ne recevra donc que la recopie de la référence.

Pourquoi ce choix, par rapport à un modèle où tout message transitant par le noyau aurait entraîné une copie du message (ce qui aurait garanti le cloisonnement des données) ? Principalement par souci de généricité des modèles de communications : on peut reconstruire un modèle par recopie à partir de celui que nous proposons, mais l'inverse n'aurait pas été faisable. Le second avantage, fort prosaïque est celui de l'efficacité. Le chapitre 8 donne quelques évaluations de la performance du système d'envoi de messages, locaux ou distants.

Politiques d'exécution

Agents concurrents Pour faciliter les implémentations de modèles plus "cognitifs" d'agents, la classe de base `Agent` est définie pour associer un processus à chaque agent. Quelques fonctions de base sont rajoutées, comme la mise en attente d'un message.

Agents synchrones Néanmoins, pour certains types de systèmes, en particulier les systèmes multi-agents réactifs, il est nécessaire de gérer un grand nombre d'agents (jusqu'à des centaines de milliers) de granularité assez fine, et de contrôler leur ordonnancement. Cela est quasi-impossible si l'on se repose sur un mécanisme de processus en raison de de la surcharge induite. Dans ce cas, on utilise ces agents "synchrones" via des agents externes qui vont définir leur politique d'exécution synchrone. Des agents d'observation peuvent être ajoutés pour avoir une vue globale sur certaines propriétés, à l'instar des plate-formes classiques de simulation comme CORMAS [Bousquet et al., 1998] ou Swarm [Burkhart, 1997].

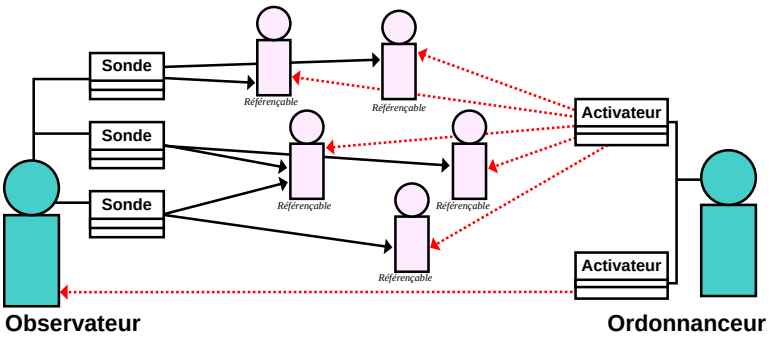
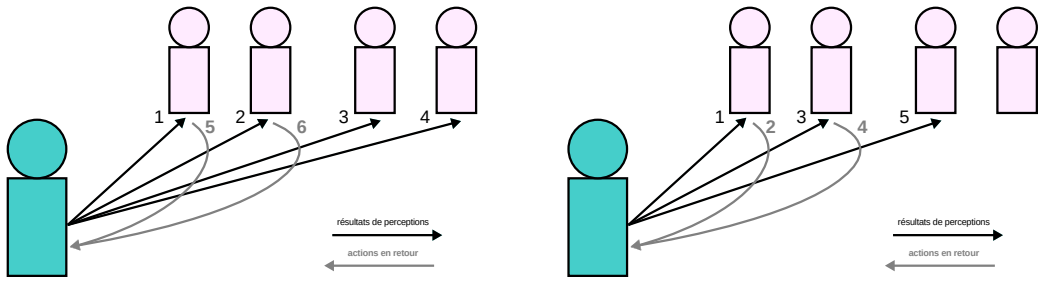


FIG. 6.8 – Fonctionnement des agents synchrones

Ce modèle permet l'utilisation de plusieurs ordonnanceurs simultanés d'un système simulé par le mécanisme de groupe et rôle. Cette architecture est décrite plus en détail dans [Gutknecht et al., 1998]. L'intérêt de cette implémentation est que les agents exécutés dans ces systèmes synchrones ont exactement les *mêmes fonctionnalités* (passage de message, vue

organisationnelle, ...) que les agents en processus et peuvent donc être associés facilement en systèmes hybrides. De plus, la gestion du mécanisme de simulation ou d'observation repose une fois encore sur l'approche en groupe et rôle. Par exemple, on va associer une sonde à une certaine propriété présente dans les tenants d'un rôle donné sur un certain groupe.

[Michel, 2000] montre plus en détail les conséquences de cette architecture pour le contrôle du biais dans les simulations multi-agents.



(a) activités parallèles (b) activités séquentielles

FIG. 6.9 – Deux modes de synchronisation

Pour préciser ce problème, prenons l'exemple classique d'un ensemble d'agents à activer sur un environnement commun.

Une approche possible consiste à mettre en place une synchronisation exécutant tour à tour chacun des agents. Chaque entité va alors percevoir des modifications de son état et entourage (environnement simulé, messages, structure organisationnelle) puis agir sur celui-ci - figure 6.9(a).

Une autre approche courante consiste à séparer cycles de perception et d'action, et de synchroniser d'abord l'ensemble des perceptions et des traitements chez les agents. Ensuite seulement est définie l'action des agents et leur prise en compte par le modèle applicatif - figure 6.9(b).

Cette approche peut paraître plus complexe (puisque l'on subdivise les activités des agents), elle a néanmoins l'avantage d'éviter un favoritisme pour le premier agent activé (premier à percevoir *et* agir dans le premier modèle).

On pourrait continuer en décrivant la structure d'une simulation conduite par événements, ou des mécanismes plus complexes comme [Magnin, 1996] qui évalue en permanence les moments clés d'une simulation d'agents spatialement situés.

6.4 Exemples

Nous donnons ici quelques exemples rapides d'implémentation d'agent MadKit, montrant la gamme possible au niveau des modèles, d'un simple agent réactif à un agent à base de règles.

6.4.1 Politique d'execution

Prenons l'exemple de fourmis se déplaçant au hasard sur le sol. On définira la fourmi comme un agent se déplaçant au hasard, avec l'implémentation (simpliste) suivante[5] :

```
public class Fourmi extends AbstractAgent implements ReferenceableAgent
{
   public double x = 0;
   public double y = 0;

   public void activate()
   {
      joinGroup("fourmiliere");
      requestRole("fourmiliere", "fourmi");
   }

   public void walk()
   {
      x=x+(Math.random()-0.5);
      y=y+(Math.random()-0.5);
   }
}
```

La définition du comportement de la fourmi est ici isolée dans une seule méthode de sa classe (`walk`). Par contre, l'activation de ce comportement n'est pas encore connu.

Nous devons alors fournir un moyen d'exécuter le comportement de telles fourmis : nous allons définir un ordonnanceur pour cela. L'ordonnanceur est un agent particulier qui gère plusieurs objets outils, les activateurs. Chaque activateur définit une politique d'exécution synchrone.

```
public class ActivateurFourmi extends Activator
{
   public ActivateurFourmi(String group, String role)
   {
      super(group, role);
   }

   public void execute()
   {
      for (int i=0; i < agents.length; i++)
```

[5] Une mise en œuvre plus réaliste ferait le lien avec un modèle d'environnement. Puisque nous examinons ici les mécanismes d'ordonnancement, nous passerons sous silence cet aspect.

```
                agents[i].walk();
        }
    }
```

L'ordonnanceur pourrait être simplement :

```
public class OrdonnanceurInsectes extends Scheduler
{
  public void activate()
  {
     foundGroup("fourmiliere");
     requestRole("fourmiliere","scheduler");
     for (int i = 0; i < 100; i++)
          launchAgent(new Fourmi(i));
  }

  public void live()
  {
     ActivateurFourmi a = new ActivateurFourmi("fourmiliere","fourmi");
     addActivator(a);

     while (true)
        a.execute();
  }
}
```

Pourquoi avoir défini cette architecture à deux niveaux, ordonnanceur et activateur pour l'implémentation de la politique d'exécution ?

Il nous paraît essentiel de penser dès cet instant à l'usage final de ces outils : ce que l'on désire construire, ce sont des *simulations* multi-agents, formées d'agents (a priori réactifs), probablement situés dans un environnement commun. Et le premier désir de l'expérimentateur est d'avoir un tant soit peu confiance en ses outils. En particulier il est notable que la définition même d'une politique d'exécution peut avoir un biais inacceptable sur le système que l'on observera. On désire bien se situer dès la conception de ces outils d'infrastructure dans un cadre de construction expérimentale et de validation.

Imaginons par exemple des agents en compétition pour une ressource limitée (par exemple l'accès à un point d'eau). Si les entités sont toujours exécutées dans le même ordre, la première risque d'être systématiquement favorisée pour sa consommation par rapport à la dernière qui ne trouvera plus une goutte. Une simple exécution dans un ordre aléatoire permet de s'en prémunir. Cet exemple simpliste montre la difficulté qu'il peut y avoir à contrôler l'impact de l'outil.

C'est pour cette raison que nous avons choisi là encore de découpler les fonctions de pur ordonnancement, d'interaction, et dáction pour pouvoir plus facilement étudier *isolément* les conséquences des choix d'implémentation, et ce par exemple en remplaçant notre premier activateur itératif de fourmis par autre chose (aléatoire, à deux phases comme dans la figure 6.9, etc...).

6.4.2 Modèles

La définition d'un agent étant plutôt minimaliste, différentes bibliothèques implémentant des architectures d'agents classiques ont été construites en complément de cette architecture.

Par exemple, un modèle d'agent faisant une liaison avec le moteur de règles JESS a été développé. JESS [Friedman-Hill, 1998] est une implémentation d'un sous-ensemble de CLIPS (C Langage Integrated Production System) basé sur l'algorithme Rete. Les propriétés des agents MADKIT (par exemple l'organisation) sont traduites automatiquement sous forme d'assertions dans la base.

Voici par exemple ce que peutt donner une partie d'une implémentation (naïve) d'une compagnie aérienne dans notre exemple de l'agence de voyage, si l'on utilise une définition à base de règles via un modèle d'agent faisant la liaison avec le moteur JESS.

```
(assert (travel (from Montpellier) (to Paris) (price (random))))
(assert (travel (from Paris) (to Montpellier) (price (random))))

(joinGroup "voyagistes")
(requestRole "voyagistes" "fournisseur")

...

(defrule receptionOffre ""
    ?m <- (Message (sender ?s) (action ?a) (content ?c) (id ?i)
        =>
        (assert (message ?s ?a ?i))
        (assert-string (str-cat "(message-content " ?i " " ?cont")"))
        (retract ?m)
)

(defrule reponseProposition ""
    ?voyage <- (message ?client "demandeVoyage" ?i)
    ?contenu <- (message-content ?i ?f ?t)
    (travel (from ?f) (to ?t) (price ?p))
    =>
    (sendMessage ?client proposal (str-cat ?f " " ?t " " ?p))
    (retract ?voyage)
    (retract ?contenu)
)
```

Notons dans cet exemple l'intégration du dialogue par message avec le moteur : les nouveaux messages apparaissent comme de nouveaux faits dans la base (les modifications de l'organisation y seraient de même signifiées). Le travail nécessaire pour faire la liaison avec MADKIT est peu important : définition de quelques types de messages dans l'environnement JESS pour faciliter l'utilisation, liaison des actions de l'agent comme primitives JESS.

Notons encore que l'utilisation d'un de ces modèles n'est nullement exclusif : il est courant d'exécuter simultanément sur la même plate-forme MADKIT des agents Scheme, réac-

tifs, systèmes, BDI ...

6.5 Extension et aspects réflexifs

6.5.1 Rôle du meta-niveau

Extension et surveillance

Le noyau lui-même est en fait encapsulé dans un agent particulier, le `KernelAgent` qui est créé automatiquement à l'amorçage de la plate-forme. Il agit comme représentant du noyau dans le monde agent : toutes les demandes de surveillance ou de contrôle sont alors écrites comme une interaction inter-agent et préservent l'unicité du modèle.

Le noyau est extensible par des points d'interceptions (*hook*) sur ses opérations. Un agent autorisé (par exemple en ayant rempli les conditions d'accès au groupe "système") a la possibilité d'utiliser l'un de ces hooks. La demande se fait par une interaction agent avec le `KernelAgent` mentionné plus haut.

Ces points d'accroche sont un mécanisme de publication/abonnement qui permet soit la surveillance, soit la redéfinition des opérations au meta-niveau. Quasiment toutes les fonctionnalités de base du noyau (lancement d'un agent, accès et modifications à la structure organisationnelle, passage de message) sont concernées. Concrètement, cela permet donc d'écrire facilement des agents d'analyse d'un système multi-agents en fonctionnement (dans le style de [Hubert Proton, 1997]), ou bien des agents étendant les fonctionnalités de base de la plate-forme. Ce fonctionnement est encore partiel car l'intégralité des primitives n'est pas encore exposé au méta-niveau.

Une telle analyse de systèmes a d'ailleurs été implémentée dans la plate-forme sous forme d'un jeu d'agents spécialisés dans l'*observation* d'autres systèmes multi-agents. Les points examinés sont la structure sociale et sa dynamique ainsi que les interactions.

Le nombre réduit de fonctionalités du cœur de la plate-forme favorise la modification du comportement des opérations de base tout en préservant au maximum leur sémantique. Le noyau est extensible par deux types de "hooks" différents, présents sur chaque opération primitive :

Les points de surveillance. Un nombre quelconque d'agents peut s'abonner à une accroche de ce type. Toute invocation de l'opération concernée provoque l'envoi d'un message d'information comprenant le contexte de l'action par le `KernelAgent` vers les abonnés. Par exemple, on peut ainsi faire tracer par un agent les interactions dans un système multi-agents en espionnant l'opération de passage de message.

Les points d'interception. Dans ce cas, un seul agent peut utiliser un point d'interception sur une opération du noyau, selon un pattern de délégation. L'agent reçoit alors le contexte qui aurait dû être celui de l'opération, inactivée : l'agent a alors latitude pour

définir un nouveau comportement. Pour continuer sur l'exemple de l'opération de passage de message, les agents qui permettent à MADKIT de fonctionner en distribué interceptent ces invocations pour pouvoir router les messages non-locaux via un protocole réseau.

L'autre mécanisme d'extension du noyau consiste à invoquer des opérations de base depuis le meta-niveau. Dans ce cas, ce sont des agents habilités qui enverront une demande au `KernelAgent` pour que le noyau effectue une opération. Par exemple, un agent de communication réinjectera par ce biais les messages venus d'une machine distante dans le noyau local.

Amorçage

Pour clarifier ces relations entre noyau, agent, action au méta, nous avons schématisé le démarrage d'une plate-forme MADKIT. Comme souvent, l'apparition du niveau meta demande une petite "tricherie" préliminaire lors de l'amorçage, comme le montre le scénario suivant (voir aussi figure 6.10 pour la séquence des échanges entre parties) :

1. Lancement de l'application hôte.
2. L'application hôte instancie un micro-noyau MADKIT.
3. Le noyau met en place les threads utiles à son fonctionnement propre, puis initialise les tables de groupes et de rôles.
4. Un groupe *local* est artificiellement créé par le noyau.
5. Un agent `KernelAgent` est instancié par le noyau. Le constructeur de cet agent prend en paramètre une référence sur le noyau, ce qui lui donne tout contrôle sur celui-ci.
6. Le `KernelAgent` est démarré par le noyau. Il est inscrit de manière classique dans les tables d'agents actifs et ses méthodes d'initialisation puis d'action sont invoquées.
7. Le `KernelAgent` met en place une table de points d'accroche, qui lui permettront de modifier le fonctionnement du noyau (espionnage, actions sur les tables de groupes, changement de la politique de transmission de message).
8. Le `KernelAgent` crée le groupe *system* et s'y enregistre sous le rôle *kernel* en tant que créateur, il y définit la politique d'acceptation. Tout entrée est validée par lui.
9. Le `KernelAgent` se met en attente de messages.
10. Le noyau finit son démarrage par une mise en sommeil des threads non nécessaires, place une sécurité pour relancer un agent système en cas de mort de celui-ci. A partir de ce moment, d'un point de vue externe, tout se passe comme si c'était le `KernelAgent` qui avait lui-même lancé le noyau, l'amorçage est terminé.

11. L'application hôte qui est la seule composante à part le `KernelAgent` à encore avoir éventuellement une référence directe sur le noyau peut, si désiré, enregistrer une classe implémentant une interface `madkit.kernel.GraphicShell` pour être prévenue par le noyau des démarrages d'agents pour la mise en place d'interfaces graphiques.

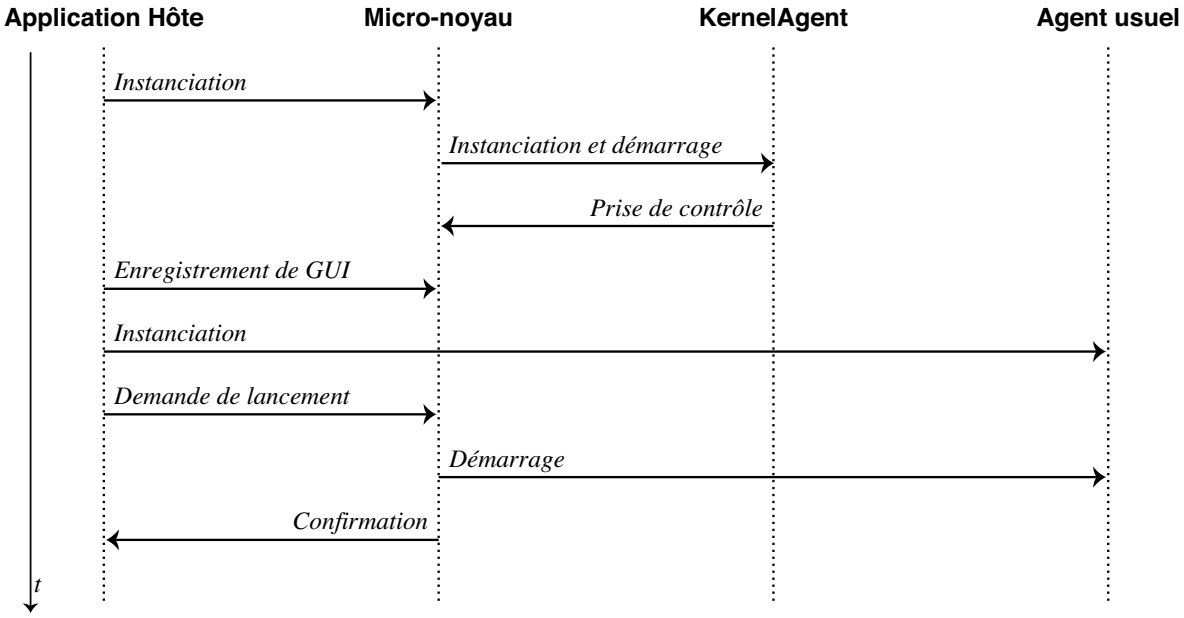


FIG. 6.10 – Amorçage de MADKIT

6.5.2 Agentification des services

A l'opposé de plate-formes plus monolithiques, MADKIT utilise donc des agents pour implémenter au niveau meta des services comme le passage de message distribué, la migration d'agent, la sécurité d'organisations d'agents et divers aspects de gestion du système ; et ce, éventuellement à l'aide des mécanismes d'extension du noyau.

Comme ces services sont mis en œuvre par des agents et décrits par la structure organisationnelle, les interactions entre ces agents systèmes, le noyau et les agents de l'applications sont décrits dans le modèle agent-groupe-rôle. Ceci implique qu'un agent offrant une fonctionnalité donnée peut être remplacé par un autre de façon transparente (et éventuellement durant le fonctionnement d'une application), à partir du moment où les groupes, rôles et interactions sont respectés.

Par exemple, des développeurs extérieurs ont échangé nos agents de synchronisation de l'information organisationnelle par d'autres qui établissaient une interface avec des annuaires distribués. Les agents du système multi-agents applicatif et les autres agents "système" n'ont pas eu à être modifiés et n'ont en fait pas remarqué l'échange.

Baser les services sur des agents décrits en terme de rôles a également l'effet de faciliter une montée en puissance. Un agent tenant plusieurs rôles peut, au fur et à mesure que les besoins de l'application grandissent, déléguer dynamiquement à de nouveaux agents certains de ses rôles dans le but de réduire sa charge.

De plus, la structuration en groupe agit souvent en "masquage" et délégation d'un système multi-agents complexe, un rôle ne correspond pas forcément à une mise en œuvre par un et un seul agent. Par exemple, un agent fournissant un rôle de communication système peut être un simple *représentant* d'un groupe d'agents spécialisés prenant en charge chacun un protocole.

Agents de contrôle

Voici quelques exemples (figure 6.11) d'agents d'observation construits à l'aide du mécanisme d'extension que nous avons décrit dans la section précédente. Sur ce cas, on a pris l'application simpliste du ping pong entre agent et utilisé trois agents de surveillance :

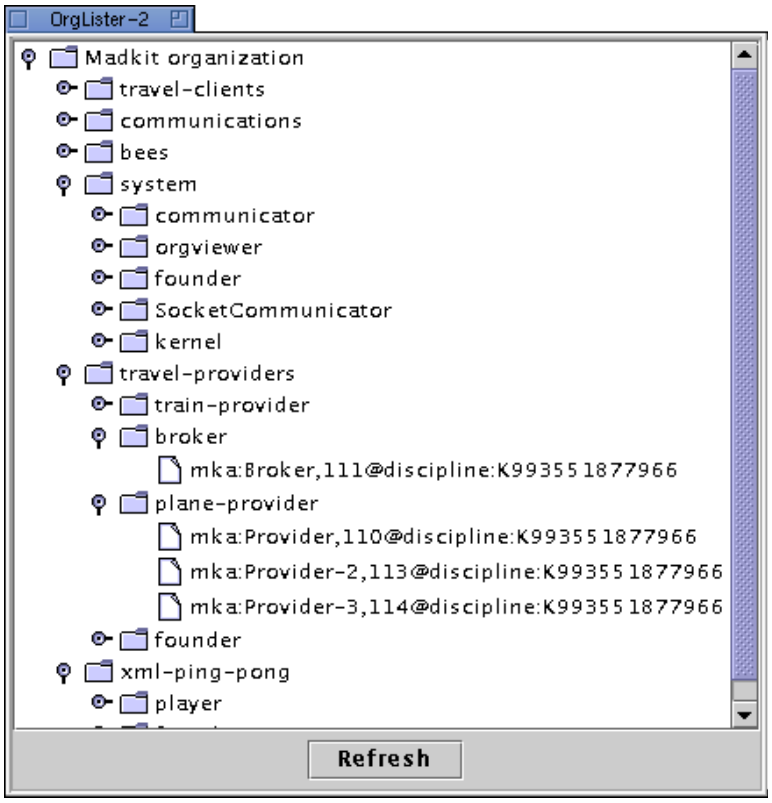
Un observateur d'organisation qui donne accès à l'intégralité des groupes définis dans la société d'agents en cours d'exécution. On peut également vérifier les rôles actuellement tenus (sachant qu'un rôle ne peut exister sans agent pour le jouer) ainsi que les agents correspondants. Un même agent peut donc se retrouver plusieurs fois dans cet arbre. On notera également la présence d'un "groupe" système regroupant entre autres les agents travaillant au meta niveau et l'agent encapsulant le noyau sous le rôle *kernel*. Cet agent d'observation fonctionne en interceptant et synthétisant les différents changements dans la structure sociale.

Un traceur d'organisation Cet agent fonctionne sur les mêmes bases, mais se contente de rendre compte des changements, pour faciliter la mise au point d'organisations complexes, ou pour exporter la dynamique observée à des fins d'analyse.

Un traceur de message L'objectif est le même que l'agent précédent, garder trace d'une activité, mais cette fois-ci uniquement du point de vue des interactions. On notera que les différents mode de communication sont identifiés : envoi direct à une accointance, envoi à un agent identifié par son rôle, diffusion vers un rôle (par exemple ici vers tous les agents de rôle *plane* dans le groupe *plane-provider*, voir figure 6.11).

Fonctionnement distribué

Avoir découplé les services du noyau permet aussi de ne pas alourdir les applications inutilement. Si l'on veut déployer un système multi-agent localement sur un seul noyau, ou bien en distribué sur un ensemble de plate-formes MADKIT, la seule différence réside dans le lancement d'agents spécialisés pour gérer les communications distantes (figure 6.12).



(a) Visualisation de l'organisation

MessageTracer-2

Sender	Receiver	Message Class	Content	Date
Client-2	Broker-4	ACLMessage	(request :conte...	19:33:59 0...
Broker-4	<travel-providers,plane...	ACLMessage	(request-for-bi...	99357684...
Provider-9	Broker-4	ACLMessage	(bid :content 1...	19:34:03 0...
Provider-5	Broker-4	ACLMessage	(bid :content 1...	19:34:03 0...
Broker-4	Provider-9	ACLMessage	(make-contrac...	19:34:10 0...
Broker-4	Client-2	ACLMessage	(make-contrac...	19:34:24 0...
Client-2	Provider-9	ACLMessage	(validate :send...	19:34:24 0...
Provider-9	Client-2	ACLMessage	(accept-contra...	19:34:24 0...

(b) Trace de messages

OrganizationTracer

Agent	Action	Group	Role	Date
PingPong	found_group	ping-pong		12:51:52 0891
PingPong	add_member_role	ping-pong	player	12:51:52 0926
PingPong-2	join_group	ping-pong		12:51:58 0812
PingPong-2	add_member_role	ping-pong	player	12:51:58 0816
PingPong-2	leave_group	ping-pong		12:52:04 0760
PingPong	leave_group	ping-pong		12:52:04 0776

(c) Trace de la dynamique de l'organisation

FIG. 6.11 – Agents d'observation

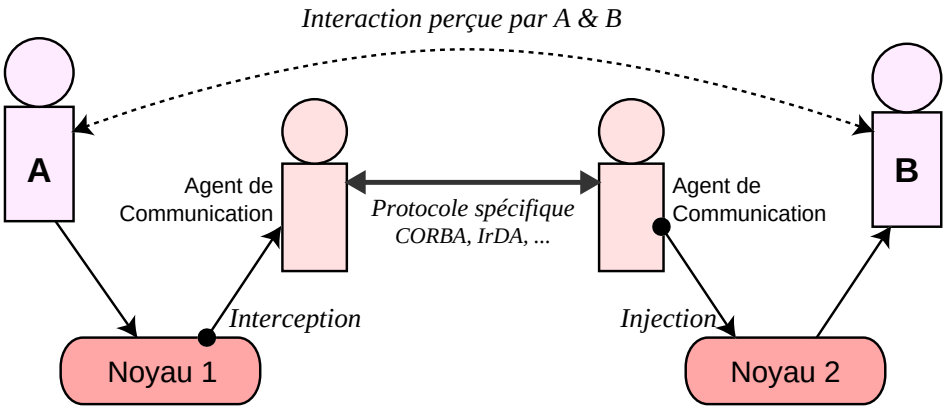


FIG. 6.12 – Agents de communication distribuée

6.5.3 Variations d'usage

Le noyau de la plate-forme ne fournit aucune interface graphique pour l'utilisateur. Il est par contre possible de lui associer une application hôte qui la mettra en œuvre. De plus, chaque agent peut définir un objet graphique pour sa représentation, d'un simple bouton à une application classique encapsulée. L'hôte aura alors la charge, au lancement d'un agent, de décider de la mise en place de l'objet graphique défini par l'agent (dans des fenêtres séparées, combiné avec l'interface d'un autre agent, etc..) et de les gérer au niveau global.

Associé au principe d'agentification des services, cela permet de modulariser complètement la plate-forme, de l'infrastructure d'exécution d'agent à l'outil d'aide au développement. Parcourons donc quelques déclinaisons possibles de la plate-forme.

Un outil de développement

Le plus souvent, l'utilisation de la plate-forme se fait par la G-BOX, un environnement facilitant le test et le développement de systèmes multi-agents. Cet environnement (application graphique hôte et agents spécialisés) permet de contrôler le cycle de vie des agents, charger de nouveaux "packs" d'agents ou d'architectures, de manipuler graphiquement les accointances (via les tables de groupes et de rôle) à fin de prototypage ou de correction d'erreur. L'interface graphique permet de garder trace et d'organiser son espace de travail même avec un nombre d'agent important. la catégoriser en trois zones :

La barre d'agents qui identifie les modèles d'agents (Java ou Scheme), ainsi que leurs regroupements en bibliothèques, et permet à l'utilisateur de lancer directement diverses instances. Des bibliothèques supplémentaires peuvent être rajoutées dynamiquement et apparaissent dans cet espace.

L'inspecteur de propriétés affiche et permet de modifier directement les propriétés de la

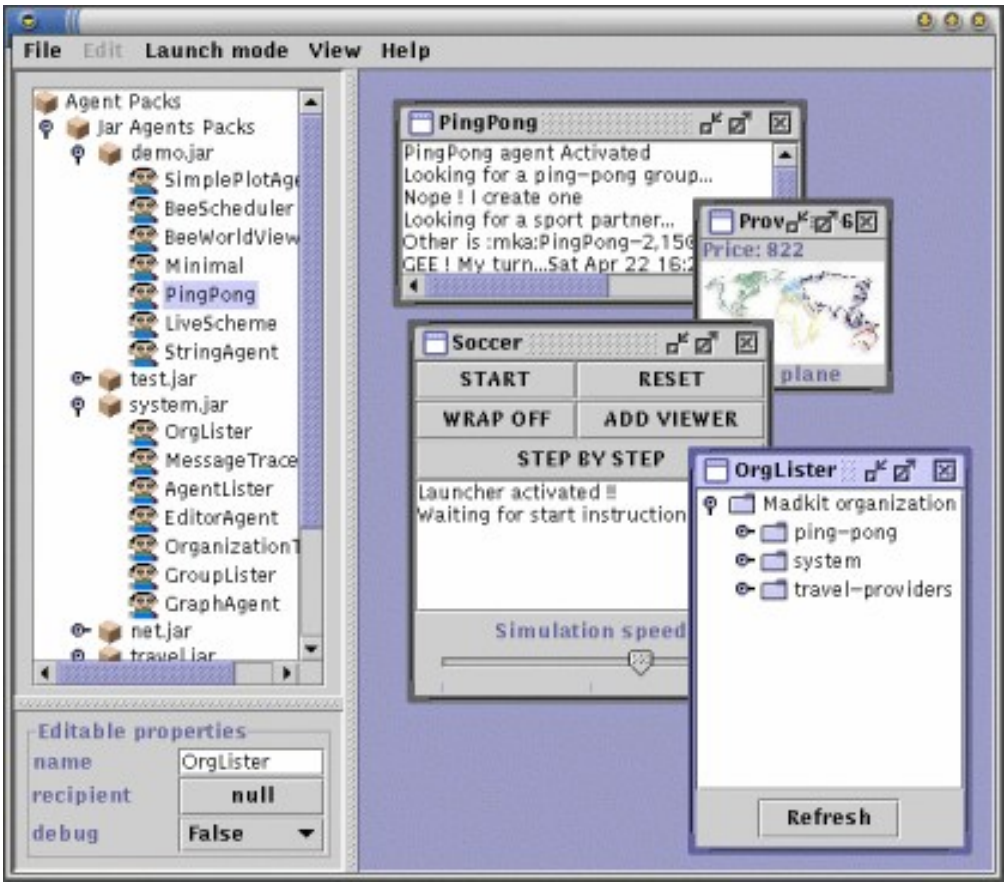


FIG. 6.13 – Déclinaison en environnement de développement

classe principale de l'agent, en découvrant par réflexivité les valeurs manipulables dans le code. Les propriétés peuvent être soit des types primitifs du langage d'implémentation, soit des propriétés "agent" au sens MADKIT : accointances, identification de groupes et rôles, et qui sont alors manipulés en tant que tels (par exemple en présentant la liste des agents tenant des rôles dans un groupes si l'on veut modifier arbitrairement une accointance).

L'espace de travail qui met en forme les interfaces graphiques isolées des différents agents. Une structuration en différents niveaux permet de masquer temporairement à la vue des portions de l'application multi-agents en exécution.

Des scripts de lancement d'applications agents peuvent être également enregistrés, en précisant l'ordre des instanciations et les propriétés à rétablir dans le système multi-agent en démarrage. Notons finalement que certaines parties de la G-BOX sont elle-mêmes conçues sous forme d'agents. En particulier, le module responsable de la mise en forme graphique des interfaces des agents applicatifs est lui-même un agent, communiquant avec le mécanisme d'instanciation.

Une micro plate-forme

Si l'on peut avoir un noyau et un système multi-agent fonctionant dans quelques dizaines de Ko, cela ouvre la voie à l'utilisation d'agents sur des plate-formes restreintes. Par exemple, la catégorie de machines rangées sous le termes d'assistants électroniques ou "PC de poche".

Pour valider cette idée, nous avons développé une version de MADKIT utilisant le même noyau que la plate-forme complète et capable de tourner sur des assistants personnels de type Palm[6]. Un système multi-agents applicatif réduit (gestionnaire de rendez-vous) a été implémenté de façon classique (dans la G-BOX) puis mis en place avec un agent de communication spécifique (utilisant le port infra-rouge). Le prototype complet fonctionne malgré les limitations fortes en puissance de calcul et occupation mémoire (moins de 64 Ko disponible pour l'application). On a vu que notre désir de garder le noyau le plus minimaliste possible a une conséquence pratique assez directe : la taille de ce module est très réduite et permet d'embarquer une (petite) application agent. De plus, le fait de ne garder trace que d'agents abstraits dans le noyau et de ne pas du tout se soucier d'interface graphique ou de réseau, limite au maximum les contraintes d'implémentations (par exemple, le système graphique est complètement spécifique à la plate-forme Palm).

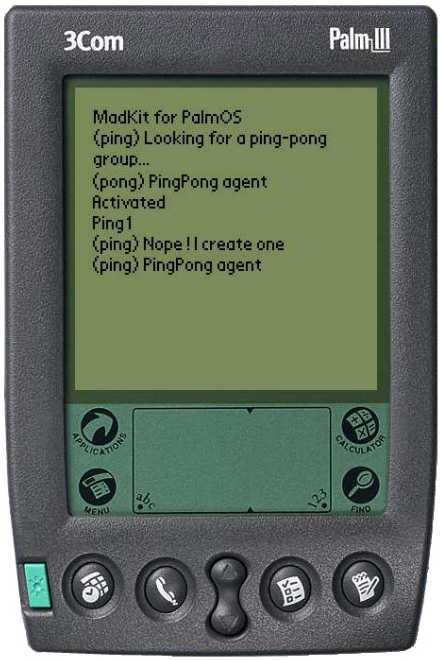


FIG. 6.14 – Déclinaison pour plate-forme embarquée

On voit là aussi l'intérêt d'avoir découplé en agentifié les mécanismes de distribution,

[6]Notre expérimentation a utilisé la machine virtuelle Spotless des SunLabs puis la machine virtuelle KVM de Sun. La plate-forme d'exécution était un ordinateur de poche Palm Pilot doté de 4 Mo de mémoire vive et d'un processeur de type 68000 cadencé à 16 Mhz.

migration et synchronisation de groupe. Il est connu que les applications agent sur des machines à resources limitées et connexions intermittentes forcent à concevoir de façon différente la gestion des messages ou du code mobile. Dans MADKIT, cela correspond à la réécriture d'un communicateur spécialisé, mais qui garde le même schéma d'interaction avec l'agent noyau. La technologie de communication étant généralement assez exotique (liaison par infrarouge, synchronisation sur une base), cela reste abstrait au niveau de l'agent Communicateur. Quand à l'agent applicatif, il n'a pas nécessairement à être repensé.

Un serveur agent

Une autre approche est de dépouiller au maximum la plate-forme, pour ne conserver que l'aspect environnement d'exécution.

Par exemple, pendant plusieurs semaines, un micro-noyau a été testé sur une machine serveur, sans aucune interface, avec simplement divers agents de communication (CORBA, sockets IP et courrier électronique), un agent de synchronisation de groupe et un agent "Directory Facilitator" [FIPA, 1997b]. Cette plate-forme a servi de point de contact connu pour d'autres noyaux et de "routeur" de messages entre plate-formes situés dans un contexte "CORBA" et plate-formes utilisant des communicateurs par sockets.

Dans cet usage, l'amorçage de la plate-forme propose de démarrer uniquement une liste d'agents donnés, qui peut être utilisé pour la distribution finale d'une application agent ou la mise en place d'agents (facilitateurs, annuaires, interprètes) sur un serveur.

Des applications spécifiques

Nous disposons d'un noyau autonome, avec des agents capables de confier la mise en forme de leur interface graphique à un module externe connu du noyau. Cela implique qu'il est assez facile d'intégrer la plate-forme d'exécution au sein d'une application classique : la faible taille et les contraintes réduites du noyau minimisent les implications sur l'application hôte, et les agents individuels peuvent être intégrés au reste de l'application.

En poussant cette logique un peu plus loin, on peut également concevoir des applications complètement agent ayant l'apparence et le comportement d'une application classique. Il suffit d'avoir un agent qui soit le chef d'orchestre de l'interface des autres agents et qui présente un aspect unifié et habituel (menus, fenêtres, documents, ...). C'est par exemple le cas du produit WEX que nous présenterons dans les applications.

Une application qui embarquerait un noyau MADKIT et des agents applicatifs pour prendre en charge les aspects dynamiques ou coopératifs, serait difficilement distinguable d'une application "classique".

Sur ce principe, nous avons également réalisé une variation consistant à empaqueter le noyau agent et le système multi-agent applicatif dans une applet, pour l'exécuter finalement

dans un navigateur distant. La taille limitée du noyau et la possibilité d'ajuster service et application rend cela réaliste.

6.5.4 De la plate-forme comme système multi-agent

Revenons un instant sur notre schéma d'architecture initial. Nous avons présenté l'architecture unifiée qui sert de base à la plate-forme, avec le noyau agentifié pour permettre l'extension des opérations primitives, des agents d'observation et de contrôle de la plate-forme, des mécanismes de communication distribués agentifiés, un environnement graphique lui-même conçu en terme d'agents coopératnts.

Pour conclure sur cette partie examinant les mécanismes réflexifs de MADKIT, on note que finalement, cette plate-forme d'exécution de systèmes multi-agents peut se voir elle-même comme un système multi-agent ayant pour but de fournir le plus de facilités à d'autres systèmes en construction. Si nous reprenons la représentation graphique que nous avons utilisé aux chapitres précédents, nous pourrions la représenter telle que sur la figure 6.15.

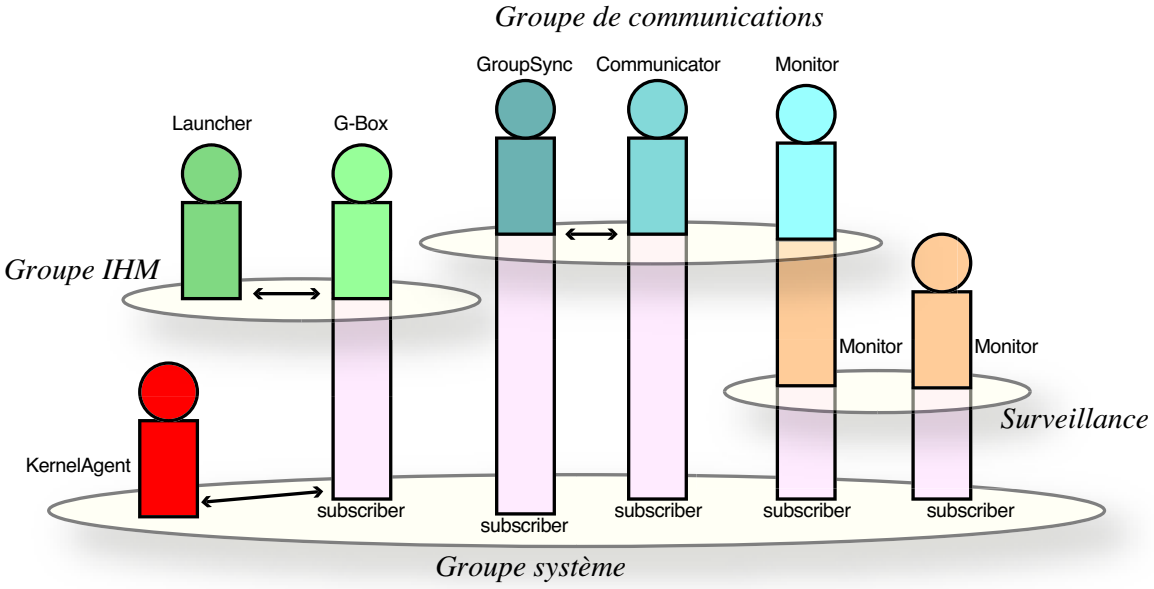


FIG. 6.15 – La plate-forme comme système multi-agent

6.6 Conclusion

Comme on a pu l'observer, ce chapitre a parfois dépassé le cadre du strict modèle "agent-groupe-rôle" qui était le cœur de notre discours jusqu'ici. Nous sommes néanmoins resté

sur le même thème : que faut-il pour implémenter des systèmes multi-agents ouverts et hétérogènes ?

MADKIT est notre contribution sur ce thème, et nous avons voulu montrer l'intérêt d'une structuration organisationnelle des applications multi-agents et comment cela peut être utilisé pour fédérer des modèles distincts et structurer le fonctionnement même de la plate-forme. Le trait distinctif de cet outil par rapport à d'autres plate-formes est d'être voulu dès le départ comme l'étude d'un *middleware* multi-agent, avec ses mécanismes d'extensions, son adaptabilité et sa neutralité vis à vis des architectures internes. Nous verrons plus avant au chapitre 8 quelques applications développées sur cette base et justifiant ce souci de généricité.

Pour finir, remarquons que nous n'avons jamais vraiment envisagé que la validation de cette approche implantoire ne pouvait venir simplement de l'architecture ou la réalisation de cette plate-forme *en soi*. Le critère essentiel pour juger de l'intérêt du modèle Agent-Groupe-Rôle du point de vue de la *mise en œuvre* était avant tout la diversité *applicative* et *architecturale* des systèmes multi-agents déployés sur cet outil.

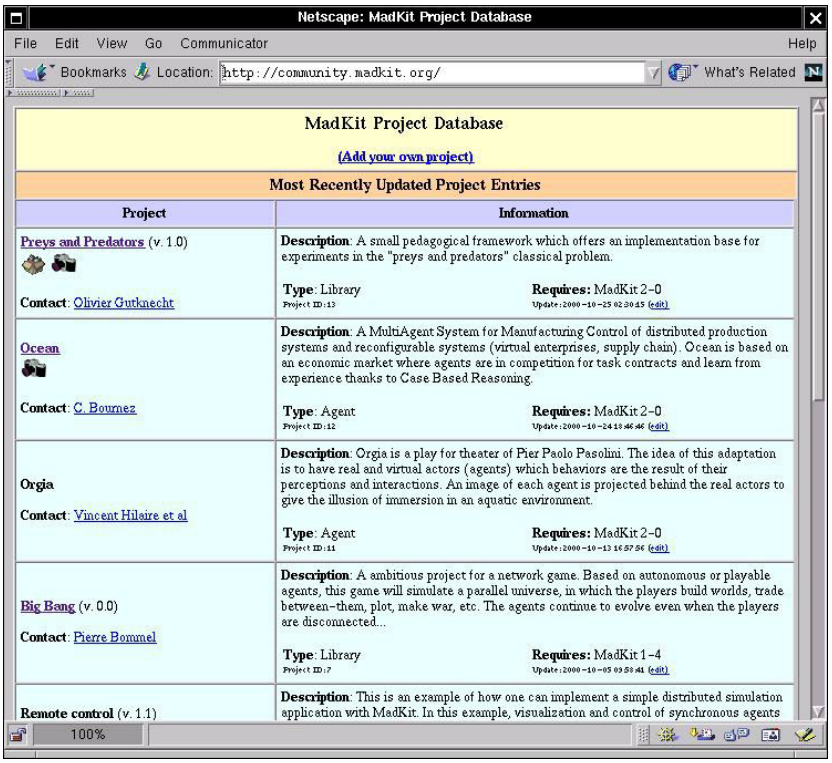


FIG. 6.16 – Base communautaire d'agents et d'applications

Cela imposait de ne pas juger que par nos propres expérimentations et nous a rapidement conduits à ouvrir l'usage de cette plate-forme le plus possible (que ce soit par le mode de

distribution, la facilité de partage de "packs" d'agents ou même le souci de qualité et la facilité de prise en main de l'outil). Nous étions d'abord notre propre cobaye : la réintégration du paradigme agent dans le fonctionnement même de la plate-forme a permis de très vite valider les choix techniques[7].

Un travail annexe au développement proprement dit a donc consisté à favoriser l'utilisation de l'outil et le partage d'agents ou de modèles entre utilisateurs [Gasser, 2000]. L'effort le plus tangible de construction d'une telle communauté a été la réalisation d'une base de données en ligne où tout concepteur peut décrire son application, voire publier et partager des modèles ou des instances d'agents (figure 6.16). La validité de cet échange de modèles vient des propriétés de cloture de la structuration organisationnelle et de leur mise en oeuvre au plus bas niveau de la plate-forme : il n'a pas de risque de conflits directs et la cohabitation de plusieurs "systèmes" est alors possible.

[7] Et montrer qu'au moins le concepteur jugeait l'outil utilisable, ce qui n'est pas toujours un critère vérifié. Sans cela, nous n'aurions jamais osé le proposer à d'autres

Chapitre 7

Approche méthodologique

Revenons à une remarque qui a amorcé notre réflexion au tout début de ce document, quand nous imaginions que les systèmes multi-agents puissent faire partie à terme du catalogue des techniques de construction logicielle classique. Une des contraintes qu'il faudrait assumer pour cela est de penser l'intégration des modèles et techniques agents dans l'optique générale du génie logiciel classique, notamment l'articulation avec les méthodologies de conception et de gestion des cycles de développement.

Cela se confirme d'ailleurs par l'intérêt récent des industriels mettant en place des projets autour de ce type de techniques, et qui insistent sur les aspects de validation et réutilisabilité des systèmes ainsi contruits [Parunak, 1998] [Baumgärtel et al., 1996].

Il faut bien admettre que malgré quelques travaux initiaux, comme [Kendall et al., 1995] ou [Burmeister et Sundermeyer, 1990], souvent élaborés à partir d'analyses a posteriori, l'importance accordée à la réflexion sur les méthodologies de conception dans les agents est un mouvement récent. Dans [Treur, 1998], Jan Treur identifie cinq domaines d'action dans le cadre des méthodes de conception de systèmes à agents, du domaine à l'outil :

- Etude des techniques de conception, par l'élaboration de méthodes et modèles pour obtenir une description complète d'un système à agent. Cela concerne par exemple l'élaboration ou l'adaptation de langages de modélisation ou de spécification.
- Définition et analyse de propriétés. En particulier par les méthodes et techniques permettant de spécifier les propriétés *requises* d'un système et l'interdépendance entre ces propriétés, par exemple via des outils d'analyse formelle et de vérification.
- Guides, méthodes et techniques d'implémentation, pour des prototypes ou des systèmes opérationnels, ainsi que l'émergence d'outils et plate-formes généraux.
- Outils d'aide à la conception : implémentation d'outils de vérification, d'aide au design, de prototypage rapide, de validation.
- Réutilisabilité : définition de méthodes et techniques pour spécifier et maintenir des modèles et modules réutilisables, et ce à plusieurs niveaux : pour les modèles d'organisations, pour les agents individuels et pour les composants internes.

Il y a en particulier un consensus général sur le fait que les modèles et guides du processus méthodologiques classiques (KADS, conception par objets) ne sont pas applicables *directement* aux approches multi-agents, et qu'une réflexion méthodologique sur les modèles agents s'impose.

Nous ne pourrons donc pas échapper à cette préoccupation : avancer la généricité d'une approche organisationnelle, comme nous l'avons fait jusqu'ici se doit d'aller de pair avec l'examen des conséquences de ce modèle d'un point de vue d'analyse ou de conception, et c'est l'objet de ce chapitre. Prévenons tout de suite que notre objectif n'est pas de proposer une méthodologie complète : cela serait par trop audacieux, et de toutes façons notre modèle ne nous le permettrait pas. En effet, nous avons insisté sur le fait que le cadre "Agent - Groupe - Rôle" était volontairement partiel, et ne cherchait pas à entrer dans une architecture spécifique d'agent ou un modèle de communication. Une méthodologie complète serait donc vouée à l'échec. Par contre, il nous paraît plus intéressant d'examiner - toujours par cet axe organisationnel - comment l'abstraction des structures que nous manipulons pourrait faciliter le travail de conception, et voir comment articuler notre approche avec d'autres plus spécialisés. Il s'agit là de proposer un cadre méthodologique organisationnel, apte à compléter d'autres méthodes et modèles.

7.1 Méthodologie multi-agents

Il s'agit avant tout de tirer parti des modèles, méthodes et outils déjà proposés pour faciliter la construction de systèmes multi-agents.

Le rôle d'une méthodologie est de passer d'un cahier des charges à une implémentation, et de pouvoir gérer le cycle de vie global d'une application. Tout ceci couvre des besoins comme la spécification initiale, l'architecture, un cadre d'implémentation, mais encore l'expression des modèles utilisés dans un but de maintenance ou de réutilisation.

Reprenons les points énumérés par [Drogoul, 2000] qui caractérisent selon lui une méthodologie :

- La définition des étapes du projet, des guides de conception pour chacune, et les successions possibles entre elles
- Un langage commun de description ou de modélisation pour faciliter l'expression du problème et la réutilisation éventuelle
- Le choix d'un niveau de description adéquat et les conséquences opérationnelles qui en découlent
- Un ensemble structuré de directives, qui inclut la définition des étapes mentionnées plus haut, des conseils de conception pour chacune des étapes, et des règles de transition entre ces étapes.
- Une façon unifiée et standardisée de documenter le processus de conception. Elle est

utilisée pour partager et transmettre l'expérience acquise durant ce processus, soit entre les concepteurs, soit à destination d'autres concepteurs.
- L'utilisation d'une terminologie homogène, qui possède une signification à chaque étape et qui facilite les transitions entre étapes (souvent une terminologie graphique à base de diagrammes ou d'organigrammes).
- L'utilisation d'abstractions conceptuelles opérationnelles, c'est-à-dire de structures suffisamment abstraites pour permettre un choix suffisant de techniques au moment de l'implémentation, mais assez concrètes pour éviter au concepteur d'utiliser des techniques dépassées ou non pertinentes.

7.1.1 Extensions d'approches classiques

Une première possibilité pour contruire une méthodologie multi-agents consiste simplement à raffiner une approche existante, et suffisamment proche des concepts agents. Dans un panorama des méthodes agents [Iglesias et al., 1998b] sont identifiées deux sources d'inspiration principales :
- Les extensions de méthodologies orientées objets
- Les approches dérivées de l'ingénierie de la connaissance

Extensions d'approches objet

Dans ce premier cas, l'intérêt est déjà la proximité avec le paradigme objet (le long débat sur la différence *exacte* entre objet actif et agent en est une bonne preuve[1]). Un point très positif pour une méthodologie agent venant d'approche objet est la popularité et la généralisation des méthodes orientées objets : l'apprentissage, ou même l'acceptation initiale, ne peut qu'être facilité si l'on se trouve face à une structuration familière. D'autre part, les différentes vues couramment employées pour décrires les objets (statiques, dynamiques, fonctionnelles, ...) se déclinent assez facilement dans le monde agent.

Néanmoins, assez rapidement, la situation se complique à cause de la différence de vue sur la communication : simple transmission de message et invocation pour les objets contre typage et protocoles complexes (voire même adaptatifs dans certains cas) pour les agents. La prise en compte de *l'interprétation* (locale et par l'agent) du message et la primauté de l'autonomie de décision de l'agent compliquent singulièrement l'étude des cas à forte dynamique, ce qui est fréquent dans les systèmes à agents.

Enfin, une question difficile se pose au sujet de la non-prise en compte de particularités que nous jugeons importante pour les systèmes multi-agents : la dimension sociale, le raisonnement ou l'émergence de comportement, qui se modélisent assez mal via ces approches, et qui se retrouvent souvent occultées.

[1] On pourra consulter par exemple [Gasser et Briot, 1998] sur ce sujet

On voit malgré tout plusieurs tentatives : [Burmeister, 1996] qui définit le modèle interne des agents BDI par des extension des cartes CRC et couple cette analyse avec un point de vue organisationnel pour préciser rôles et héritages. D'une manière assez proche, la méthode AOM [Kinny et Georgeff, 1996] propose aussi de diviser la conception à deux niveaux entre niveau structurel définissant l'architecture interne de l'agent BDI et un modèle d'interaction pour préciser communications et relations de contrôle.

On peut aussi citer [Moulin et Brassard, 1996], qui ont l'idée d'identifier tout d'abord des rôles (fonctionnels, organisationnels dans le domaine), pour ensuite les réintégrer dans un modèle OMT, avant de déterminer les responsabilités et les services associés à chaque rôle. L'interaction et la conception des agents individuels viennent alors en dernier.

Méthodes basées sur les connaissances

D'autres méthodes, basées sur des approches dérivées de l'ingéniérie des connaissances, sont proposées pour concevoir des modèles d'agents cognitifs. La définition du comportement de l'agent est alors vue comme un processus d'acquisition de connaissances. Il s'agit le plus souvent d'extensions à KADS.

Certaines méthodologies dans cet esprit introduisent également un aspect organisationnel [Glaser, 1997] pour guider la conception, en particulier les adaptations faites à CommonKADS par [Iglesias et al., 1998a]. On pourrait leur reprocher les contraintes induites - fort logiquement - par le modèle KADS lui-même. Leur orientation priviliégie dans les modèles la communication agent/humain plutôt qu'agent/agent [Sierra et Agustí, 1991].

7.1.2 Méthodologies agent et multi-agents

D'autres méthodes proposent de prendre dès le départ un point de vue centré sur la notion de système multi-agents, ce qui permet d'éviter de passer sous silence les aspects sociaux ou d'émergence de comportement.

Une approche intégrée est celle des "Voyelles" de [Demazeau, 1997] qui incite à penser la conception du système simultanément en terme d'agent, d'environnement, d'interaction et d'organisation. Des modèles spécifiques permettent alors d'exprimer les choix de conception selon ces quatre dimensions et de construire le système final par assemblage de composants venant de ces différents axes. Plus focalisé sur l'organisation [Ferber, 1995] proposait de dégager une analyse à trois niveaux :

Une analyse fonctionnelle, détaillant les fonctions de l'organisation à différents niveaux

Une analyse structurale qui différencie les différentes formes d'organisations possibles

Une analyse des paramètres de concrétisations tels que la redondance ou la spécialisation

Une des approches les plus radicales est sans doute celle de [Muller, 1998], qui prend la question de l'émergence comme base de travail. La tâche du concepteur est alors pour

beaucoup un travail d'identification, pour trouver une représentation de l'espace adaptée à la structure du problème, déterminer les interactions qui induiront les composantes recherchées dans le problème et en tirer l'expression des agents.

Avec CASSIOPÉE [Collinot et Drogoul, 1998], on est également en présence d'une méthodologie purement multi-agents, qui propose de faire un aller-retour entre dimensions individuelle et sociale, pour préciser progressivement comportements, interactions et structuration globale. Le support de cette expression progressive est en fait la notion de rôle, séparé en plusieurs domaines d'analyse :

Les rôles individuels abstraits du domaine en fonction des resources, fonctions, ou dépendances pris en compte par les agents (par exemple : conflits, permission, assistance, etc.). On peut les ramener au niveau de l'agent à des ensembles de comportements associés.

Les rôles d'interaction qui vont structurer les relations d'influence, les dépendances et les mécanismes de communication.

Les rôles d'organisation qui établissent la structure générale du système, sa dynamique et les activités au sens large des agents (par exemple, l'identification d'initiateur ou de participants à une action commune)

7.2 Aspects méthodologiques du modèle organisationnel

Qu'entendons-nous ici par aspects méthodologiques ? Il s'agit d'analyser, *comment* l'organisation multi-agent devra se comporter, pour ensuite pouvoir déterminer *quelle structure* donner à cette organisation. Pour ce faire les abstractions que nous allons retenir sont fort logiquement celles du modèle Agent-Groupe-Rôle. Néanmoins, nous avions jusqu'à maintenant évoqué ce modèle sur un plan principalement *opérationnel* (que ce soit dans des perspectives d'analyse, de formalisation ou d'implantation).

7.2.1 Esprit de la proposition

Maintenant, notre préoccupation est d'obtenir un cadre méthodologique suffisant (et répondant aux critères que nous avons identifiés) pour *exprimer* la conception d'une application en utilisant ce modèle. Cela nécessite de nouvelles abstractions pour exprimer des modèles d'organisations qui, au final, seront concrétisés dans une société d'agents (laquelle sera exprimable en termes d'agents, groupes et rôles).

Nous aurions pu nous fixer sur un modèle particulier d'agent, décider d'un formalisme pour représenter les interactions et avoir ainsi de quoi montrer la conception complète d'un système, avec le point de vue organisationnel comme articulation. Le portrait complet d'une

conception multi-agent de bout en bout aurait pu être dressée, aux dépends de la généricité et de l'articulation avec des modèles externes.

Il nous a paru plus intéressant ici de conserver notre approche : continuer à accepter notre contribution comme volontairement incomplète, montrer en quoi elle pourrait intervenir en complément d'autres méthodes et se focaliser sur des notations et modèles suffisamment souples pour être rattachés à d'authentiques et complètes méthodologies. Notre ambition est donc de proposer :

- des structures abstraites de description d'organisations,
- un langage commun de modélisation pour exprimer ces structures,
- un cadre permettant l'extension et l'intégration avec d'autres modèles d'analyse ou de conception.

Nous restreignons volontairement des aspects du modèle pour qu'il s'intègre le plus aisément possible avec d'autres méthodologies, dont celles orientées vers des modèles spécifiques d'agents [Baumann et al., 1997] [Occello et Demazeau, 1997] [Jung et Fischer, 1998], centrées sur l'organisation ou bien basées sur l'émergence [Muller, 1998]. On n'y rattache donc pas la question des comportements individuels. Ceci dépend en effet d'une analyse intentionnelle et spécifique au modèle d'agent et au domaine applicatif.

Ce modèle s'intéresse avant tout à la structuration et à la dynamique *sociale* des agents. Le rôle est donc directement lié aux positions de l'agent dans l'organisation (par une description explicite) ou comme point d'accroche des mécanismes d'interaction (ce qui le place donc en lisière des questions d'architecture interne).

7.2.2 Vue d'ensemble

Le cadre proposé se compose d'un ensemble de modèles et processus articulés autour des concepts centraux. La figure 7.1 en présente le positionnement.

Nous avons jusqu'ici toujours utilisé le modèle agent-groupe-rôle dans une perspective très neutre, pour analyser ou décrire un système indépendamment de la situation (système implémenté, réflexion sur un modèle de SMA) ou de son origine (structure typique identifiée, modèle d'activité résultant de l'action des agents, organisation émergente, ...)

Une expression dans le modèle peut être l'aboutissement d'un processus de conception classique. A l'inverse, cela peut tout aussi bien être le résultat de la symbolisation de patterns identifiés dans un système réactif, pour en faciliter l'analyse (*bottom-up*). On peut aussi envisager l'analyse organisationnelle d'un système préexistant à des fins de rétro-conception.

La perspective de ce chapitre reste néanmoins celle de systèmes utilisant un mécanisme d'interaction classique entre agents (ce qui suppose l'existence d'une architecture d'agent un tant soit peu symbolique). Ainsi, nous voudrions modéliser l'aspect social des systèmes, souvent décrit de façon très informelle par la plupart des méthodologies agents, afin de mieux exprimer, gérer et réutiliser des portions de design dans des systèmes multi-agents

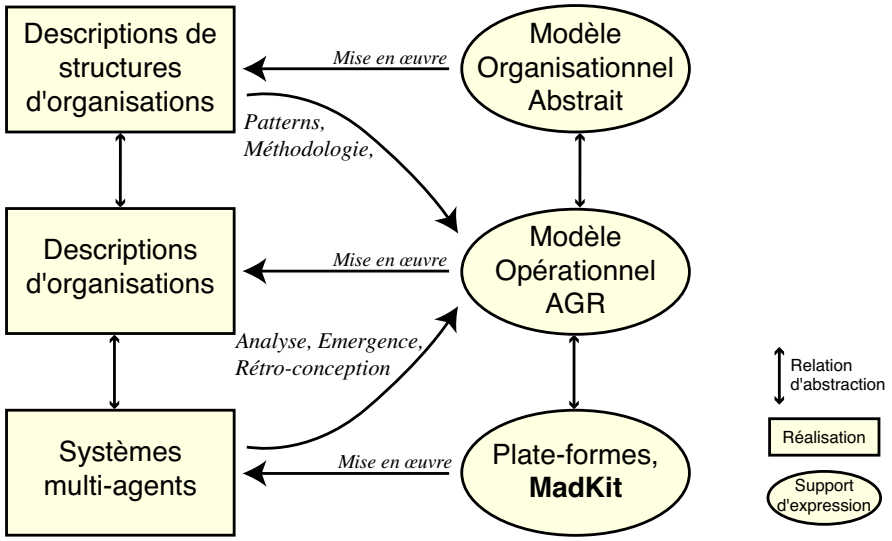


FIG. 7.1 – Vue d'ensemble

hétérogènes ou de taille importante.

La séparation des préoccupations du concepteur revient ici à envisager des modèles sociaux complémentaires, modélisés indépendamment, et que l'on réunira *ensuite* au niveau de l'architecture de contrôle de l'agent et du système global.

Les différents groupes de l'organisation peuvent alors être complétés par des propriétés spécialisées et dépendantes du domaine. Par exemple, le rôle correspond à un certain comportement attendu de l'agent dans son travail dans un groupe. Néanmoins, il peut être dans certains cas réduit à une simple tâche, associé à un statut et un schéma d'interaction, ou avoir des propriétés spécifiques ou des comportements directement rattachés au rôle.

Relations entre modèles

La figure 7.2 indique les relations entre les différents modèles :

1. **Le modèle agent-groupe-rôle** déjà décrit, permet de décrire des organisations multi-agents instanciables (ou instanciées).
2. **Le modèle organisationnel générique** présente les notions de structures de groupes et les définitions de rôles. Ce modèle est donc le modèle abstrait de structures d'organisations exprimables dans le modèle central précédent.
3. **Le modèle organisationnel spécifique** affine le modèle générique en rajoutant des concepts spécifiques au domaine ou en restreignant les définitions existantes. Une manière simple de comprendre ce modèle est de le voir comme défini par héritage du modèle générique.

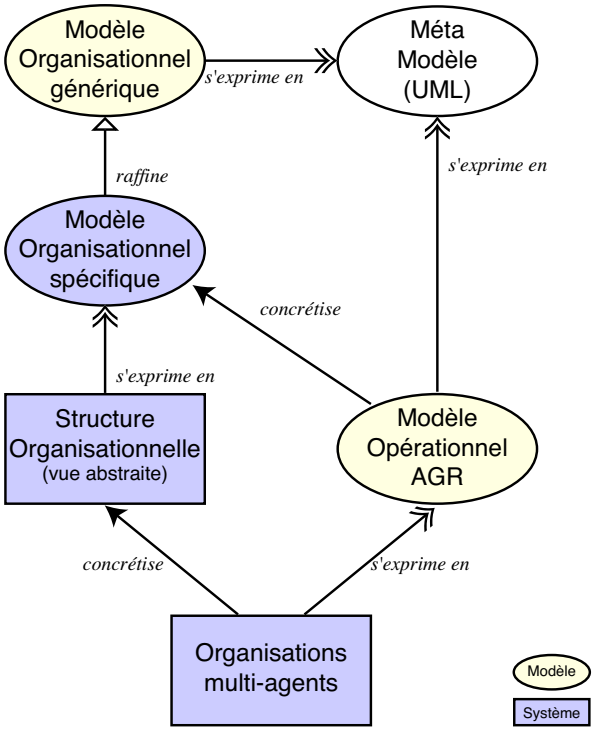


FIG. 7.2 – Relations entre modèles

4. **La structure organisationnelle** est la spécification complète d'une organisation, décrite dans un modèle organisationnel spécifique. C'est la représentation abstraite d'une organisation apparaissant dans le point précédent. Dans notre exemple d'organisation hiérarchique, il s'agit du schéma abstrait de groupes et rôles et des interactions possibles entre rôles.
5. **Les organisations** exprimables dans le modèle agent-groupe-rôle. Il s'agit là de systèmes multi-agents classiques décrits dans leur aspect organisationnel.
6. **Le méta-modèle** est le système de description des modèles et outils précédents (par exemple le meta-modèle UML pour nos notations).

Le modèle organisationnel a pour but de décrire précisément l'ensemble des rôles et des groupes qui vont former l'organisation, tant en terme de fonctionalité, activité ou responsabilité qu'en termes de protocoles d'interactions. Voyons en détail ce que nous entendons par modèle générique et spécifique.

Le modèle organisationnel générique

Le but de ce modèle est de fournir les concepts minimaux permettant de décrire une structure abstraite d'organisation multi-agents. Il doit donc être vu comme une abstraction

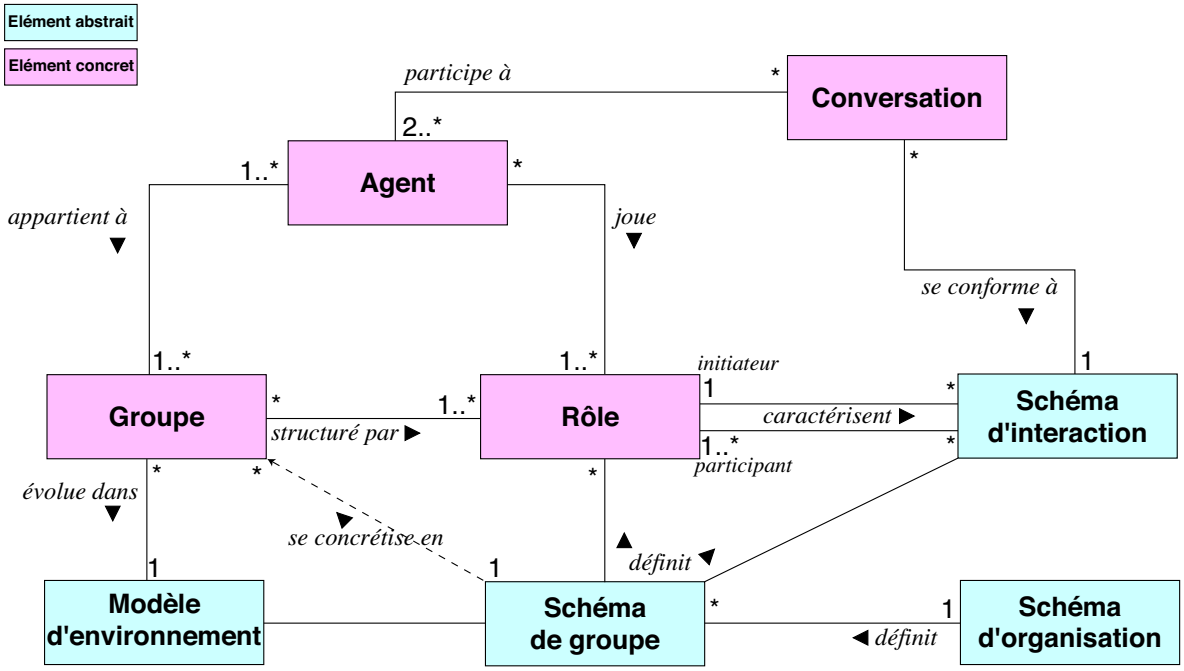


FIG. 7.3 – Le modèle AGR étendu

et un complément (voir figure 7.3) du modèle agent-groupe-rôle initial, avec pour éléments identifiés :

- La structure organisationnelle définie comme un ensemble de structures de groupe participant à la résolution. Elle définit une structure sociale abstraite complète (par exemple, un laboratoire).
- La structure de groupe, qui se définit comme une description abstraite d'un groupe (par exemple : une équipe). Elle identifie la totalité des rôles, l'environnement, et les schémas d'interaction qui sont à isoler au sein d'un groupe.
- Les rôles et leurs propriétés qui seront joués par les différents agents, dans une organisation concrète. A un niveau abstrait, ils peuvent définir des contraintes générales (comme l'arité, la fonction de rôle) et des propriétés et caractéristiques spécialisées.
- Les schémas d'interaction identifiés entre rôles du groupe impliqués (de 1 à n). On y précise le rôle initiateur de l'interaction et les autres rôles pouvant y apparaître. Les conversations qui auront lieu entre agents jouant ces rôles se conformeront à ces interactions. Le modèle d'interaction n'est pas précisé à ce stade.

Les modèles organisationnels spécifiques

Le modèle abstrait d'organisation que nous venons de parcourir est suffisant pour qualifier certains systèmes multi-agents simples. Dans la majorité des cas, il est bien sûr nécessaire de faire appel à des modèles et des concepts additionnels, plus ou moins liés à un domaine

applicatif.

La notion de modèle organisationnel spécifique permet de préciser le modèle générique en raffinant celui-ci ou en rajoutant des concepts. On envisage deux mécanismes d'extension du modèle, l'extension par raffinement et l'extension par addition, pour réexprimer des notions additionnelles à celle de rôle ou de groupe, spécifiques à une activité de design particulière.

7.2.3 Extensions et intégration

Extension par raffinement

Dans l'extension par raffinement, les concepts centraux sont modifiés pour restreindre ou compléter les cas d'utilisation ou pour définir des "prototypes" de définitions.

Par exemple, la notion de rôle peut s'affiner par l'ajout d'attributs complémentaires (compétences, capacités, .. voir figure 7.4), définir des cas particuliers de la fonction de rôle (n'autoriser qu'une certaine multiplicité du rôle au sein des groupes concrets, requérir l'appartenance à un autre groupe ou rôle pour obtenir celui-ci).

On peut également citer les modèles de la méthodologie GAIA [Wooldridge et al., 2000] [Wooldridge et al., 1999] qui peuvent s'exprimer sans modifications en extension du modèle générique. Dans cette approche, la notion de rôle est précisée par des propriétés spécifiques au modèle d'analyse : permissions, compétence, autorité, validité, et correction.

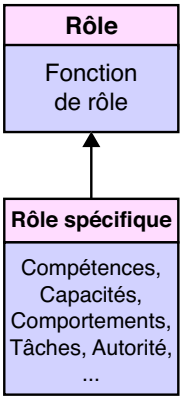


FIG. 7.4 – Spécialisation d'un rôle

L'expression du problème sous la forme identifiée agent-groupe-rôle correspond alors à une *étape* du processus de modélisation, et n'est certes pas le point initial ou final de l'analyse. L'application de contraintes successives dans l'expression - et l'analyse du problème - autorise tout à fait à relâcher un point du modèle en amont.

Extension par addition

Le deuxième mécanisme est l'extension par addition. Dans ce cas, le modèle agent-groupe-rôle n'est pas directement concerné : le modèle spécifique va rajouter des concepts supplémentaires, comme par exemple les tâches, les ressources, ...

Dans ce cadre, le lien avec le modèle central peut se faire soit par association directe (par exemple un *rôle* de foreur est concerné par *l'activité* d'exploitation du minerai), soit par lien direct avec un processus méthodologique spécialisé basé sur les concepts ici rajoutés.

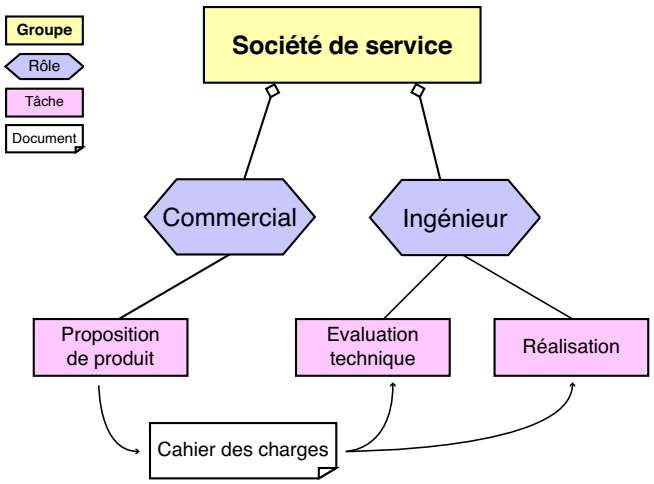


FIG. 7.5 – Extension par addition

La figure 7.5 montre un exemple de ce que peut être un modèle spécialisé de type collecticiel dans lequel des concepts de tâche et de document ont été ajoutés au modéle générique.

Nous allons d'ailleurs voir tout de suite les bases de la notation utilisée dans ce diagramme.

7.3 Notations

Après ce parcours des modèles abstraits d'organisations, nous allons proposer quelques notations pour pouvoir plus facilement représenter, partager et documenter des structures organisationnelles ou des analyses de systèmes.

Nous abordons cette question de notations par deux voies :

- La proposition d'une diagrammatique permettant d'exprimer les aspects dynamiques ou statiques, abstraits ou concrets, des organisations multi-agents exprimées à l'aide d'un modèle agent-groupe-rôle ou de ses abstractions.
- Une représentation sous une forme informatique, déclarative et définie avec précision de ces mêmes modèles, à considérer comme une approche complémentaire de la précédente.

Il nous faut également déterminer ce que nous voulons représenter. De ce que nous avons examiné des modèles jusqu'à ce moment, nous pouvons dégager trois domaines de notations :

- La notation de structures organisationnelles. C'est un regroupement statique d'éléments organisationnels abstraits, avec éventuellement des indications de dépendances ou présence de schémas d'interactions (mais sans aller dans le détail de l'expression de ceux-ci).
- La représentation des mécanismes de contrôle ou de communication des agents. Cette partie ne sera pas traitée ici, car en dehors de notre modèle et dépendante des choix d'architectures faits par le concepteur pour compléter son système. Ce sera au concepteur de respecter les conventions de notation du modèle qu'il aura choisi dans l'expression des schémas d'interaction (automates à états finis , diagrammes de collaboration UML, ...) ou des structures internes des agents (BRIC, assemblages de composants, jeux de règles, modèles objets, ...)
- L'expression d'organisations concrétisées, et en particulier l'expression de leur structure de groupe ou de rôle par rapport à des agents à un instant donné. Il nous faudra également pouvoir représenter l'évolution d'une organisation, toujours du point de vue social.

7.3.1 Modélisation graphique

L'existence des divers niveaux d'abstraction que nous venons de présenter nous amène à proposer un formalisme spécifique.

Avant d'aborder ces représentations, un petit avertissement s'impose. Le lecteur peut remarquer que dans nos notations graphiques, nous utilisons souvent (et majoritairement) des symboles en droite ligne d'UML[OMG, 1999b]. Néanmoins, nous nous en écartons à certains moments clés (en particulier la représentation des rôles). Pourquoi ne pas avoir adopté de bout en bout ce langage de modélisation, éventuellement étendu par des stéréotypes adaptés ? Nous aurions eu effectivement l'avantage d'utiliser une notation admise, et correctement définie par un meta-modèle.

Ce qui nous a fait écarter l'application directe de cette solution est la crainte d'une confusion entre les différents niveaux de modèles. Par exemple, s'il est vrai que le rôle pourrait parfaitement être représenté *au niveau conceptuel* comme une classe le qualifiant, nous craignons qu'une ambiguïté demeure, particulièrement pour la question de la mise en oeuvre finale de ces modèles. S'il est vrai que l'on peut exprimer par exemple la notion de rôle sous forme purement objet[2], le modèle agent-groupe-rôle est à comprendre, ainsi que nous l'avons vu, comme l'expression d'un niveau social situé à un cran d'abstraction *supplémen-*

[2]tout comme cela pourrait être représenté par un jeu de règles de comportement, une logique spécifique ou une expression procédurale

taire. C'est ce qui nous a amené à garder des représentations spécifiques.

Ces représentations sont issues de plusieurs origines : des notations spécifiques à notre modèle, le souci de reprendre quand cela était pertinent des notations préexistantes (en particulier du domaine objet), et des notations proposées dans d'autres travaux sur les méthodologies multi-agents.

En particulier, on peut trouver dans [Parunak et Odell, 2001] une proposition de notation basée sur UML. Les auteurs utilisent une extension de notre modèle Agent - Groupe - Rôle pour proposer une description de structures sociales. Nous reprenons à notre tour certains points de ce travail, en particulier la séparation des rôles par des "lignes de nage"[3].

Structure organisationnelle

La première représentation dont nous devons disposer est celle qui permet d'exprimer le modèle générique de structures organisationnelles. Il se composé d'une succession de définitions de structures de groupes, avec éventuellement l'expression de dépendances entre rôles.

La notation d'une structure de groupe individuelle associe un ensemble de rôles. Chaque rôle peut être précisé par sa fonction de rôle. Les attributs supplémentaires de rôle sont représentés de la même manière qu'un attribut dans UML. Les contraintes sont exprimées sous forme d'une note ou par la la syntaxe `{ ... }`.

Le schéma permet également de représenter les interactions définies au niveau du groupe. Les rôles impliqués sont liés et le rôle initiateur est fléché en direction du schéma d'interaction. Les autres rôles impliqués sont flêchés en sortie, avec la mention éventuelle de la multiplicité des porteurs du rôle impliqués dans l'interaction (par exemple un vote impliquant n agents ayant le rôle de votant mais que deux agents ayant le rôle d'assesseurs). On trouvera sur la figure 7.6 une modélisation de l'organisation abstraite de l'agence de voyage.

Organisations concrétisées

Ce type de diagramme est en fait redondant à celui que nous avons abondamment utilisé dans le chapitre 4 : il s'agit de représenter les appartenances d'agents à des groupes et rôles à un instant donné de l'organisation. Mais si la représentation que nous avons utilisée est assez intuitive et directe par cette vue en perspective utilisée, ce n'est sans doute pas la plus pratique et rapide pour une phase d'analyse, en particulier si les systèmes deviennent un tant soit peu complexes en nombres de groupes impliqués.

Nous proposons donc une nouvelle approche, inspirée de [Parunak et Odell, 2001], qui consiste à porter sur un axe horizontal les groupes, en vertical les agents du système et faire figurer aux intersections les différents rôles portés dans chaque groupe par ceux-ci. L'intérêt

[3] Que l'on retrouvera figure 7.7

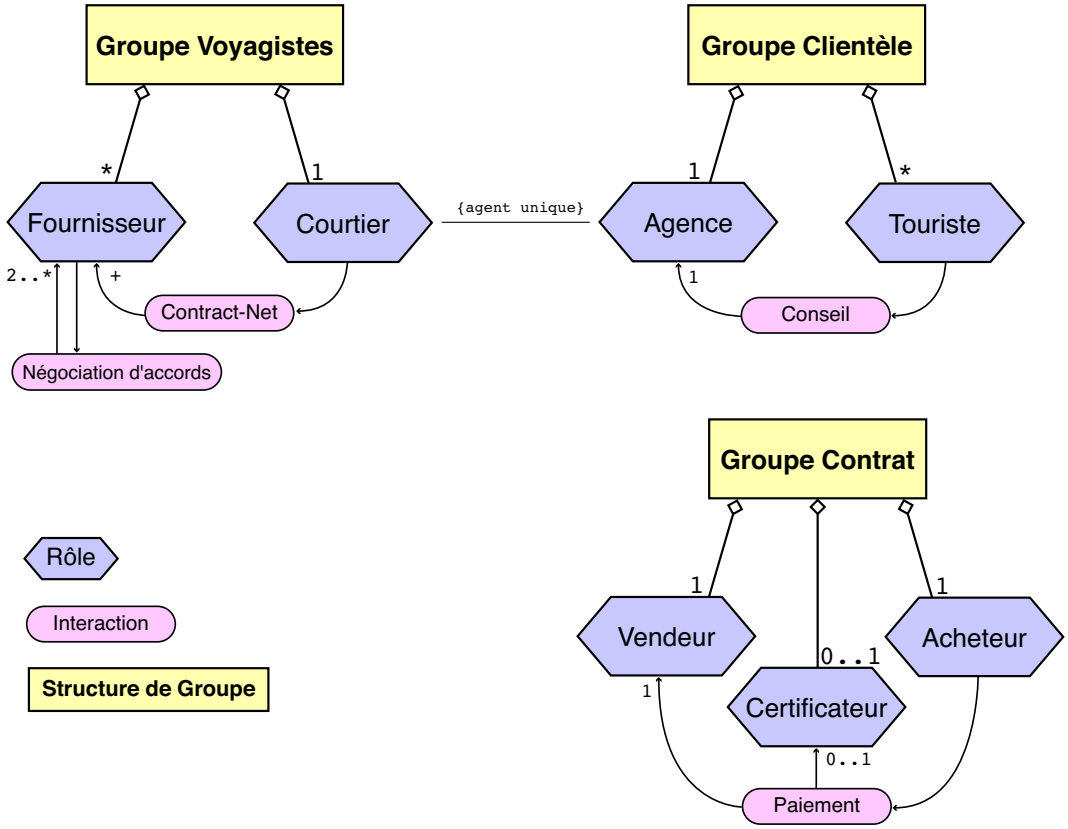


FIG. 7.6 – Représentation de la structure organisationnelle dans l'exemple de l'agence de voyage

est d'avoir un type de schéma capable de représenter des organisations de grandes tailles par simple modification de l'échelle et du nombre de colonnes ou lignes. L'autre avantage est de pouvoir porter en regard des rôles, des indications sur les styles de dépendances ou d'interaction ayant lieu entre agents tenant ces rôles.

Nous illustrons cela par la représentation d'une organisation mettant en oeuvre l'exemple de l'agence de voyage (figure 7.7). On peut comparer ce nouveau schéma avec notre approche initiale (figure 4.7).

Représentation de la dynamique

Nous disposons maintenant de quoi représenter des structures organisationnelles abstraites et plusieurs façons de mettre en évidence l'état *courant* d'une société d'agent. Mais il nous manque encore de quoi montrer *l'évolution* de nos organisations, c'est à dire la dynamique de la structure sociale.

Quels sont les points qu'il nous faut faire apparaître pour tracer l'évolution d'un système ?

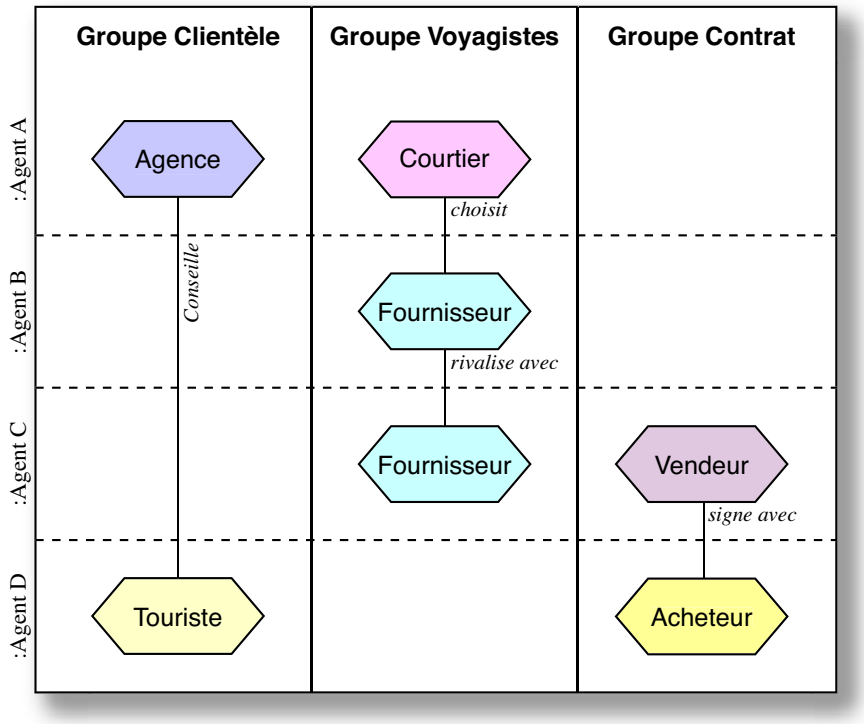


FIG. 7.7 – Représentation de la structure organisationnelle concrétisée dans l'exemple de l'agence de voyage

- Montrer, tout comme dans le schéma précédent, l'unicité des entités : on doit pouvoir suivre les changements de rôle ou de groupe pour un agent donné.
- Mettre en évidence les changements dans la structure, c'est à dire la création ou l'abandon de groupes, ainsi que les agents à l'initiative de ces modifications.
- Montrer les changements de rôles pour un ou plusieurs agents et permettre d'annoter ou de mettre en évidence les contraintes.
- Rester cohérent, et si possible s'intégrer, avec les diagrammes montrant la dynamique de l'interaction.

Le principal point délicat est de faire apparaître les différentes dimensions dans un seul schéma : temporalité, dynamique des groupes, dynamiques des rôles, identification des agents. Cela nous conduit à proposer un diagramme (voir figure 7.8 pour une mise en oeuvre sur l'exemple de l'agence de voyage) qui s'inspire là encore des "lignes de nage" pour distinguer les rôles que nous avons évoqués plus haut, mais avec une réunion par groupe. L'identification des agents individuels se fait par un trait continu portant à la fois les noeuds correspondant aux prises et abandons de rôles, les groupes d'appartenance simultanés, et la temporalité. On garde de plus la possibilité de porter en complément sur le schéma tout ou partie des séquences de messages formant les interactions.

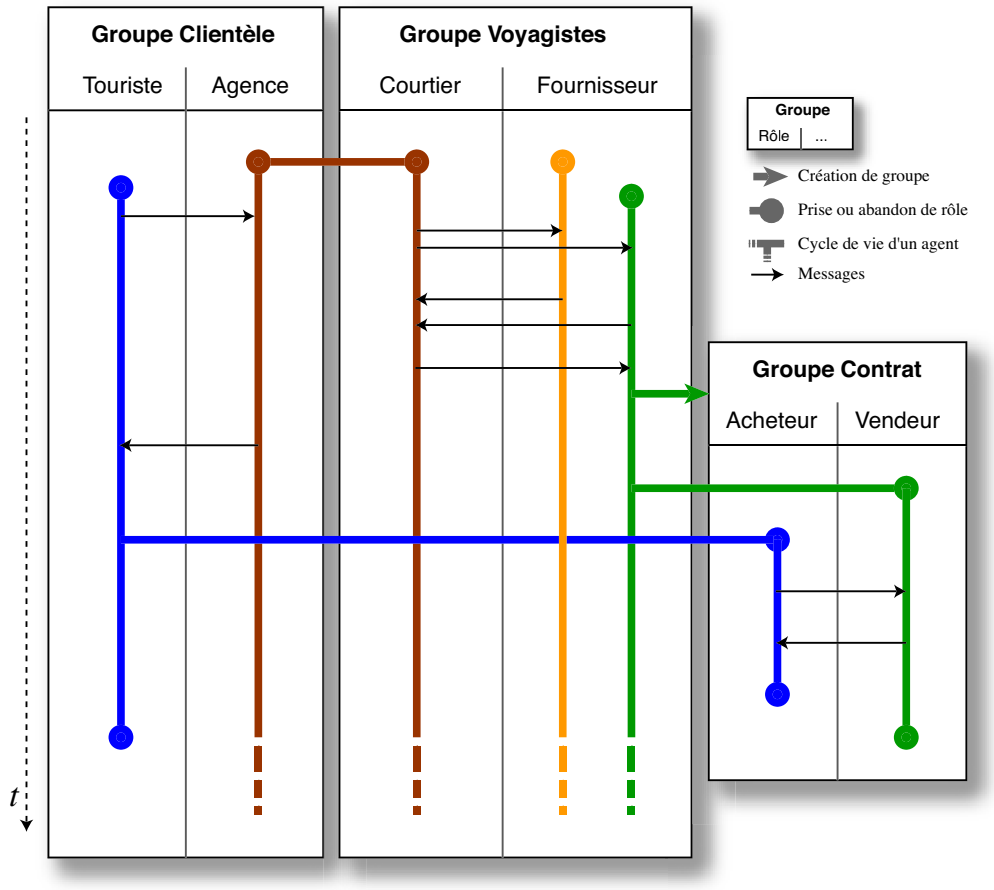


FIG. 7.8 – Représentation de la dynamique d'une organisation dans l'exemple de l'agence de voyage

Notons pour finir que ce diagramme a été conçu pour servir à la fois comme support au concepteur, mais aussi pour être facilement opérationnalisable en tant qu'outil (informatique) de surveillance d'une société d'agent.

7.3.2 Modèle d'échange

Intérêt

Si l'utilisation de notations graphiques comme UML s'avère essentielle au niveau du concepteur -humain-, on ne peut guère adopter le même qualificatif quand il s'agit de les faire comprendre à la machine. Et bien que nos programmes ne puissent (encore) comprendre nos schémas griffonnés, il est essentiel de pouvoir malgré tout fournir des descriptions non ambigües de ces structures manipulées au niveau de la conception : pensons aux outils de conception, aux validateurs, aux mécanismes de vérification de sécurité validant tel ou tel groupe, ...

Nous aurions pu définir notre propre format de données avec une syntaxe ad-hoc. Le nombre limité des concepts manipulés laissait entrevoir une syntaxe simple et implémentable dans la foulée. Nous avons néanmoins choisi une autre voie : utiliser un langage de description externe pour spécifier nos formats de données : XML [Bray et al., 1998], langage de description défini par le Web Consortium pour spécifier des langages de balisage pour données structurées. Bien que relativement compréhensible dans le texte, XML est surtout conçu pour être facilement manipulable par des outils de production, traitement ou représentation de cette information structurée, avec un jeu de marqueurs spécifique à chaque domaine applicatif (comme dans l'exemple suivant pour une bibliothèque).

```
<bibliothèque>
  <livre>
    <titre>Cuivre, Argent, Or:
           un alliage metallique indestructible</titre>
    <auteur>Egbert B. Gebstadter</auteur>
  </livre>
<bibliothèque>
```

L'idée est de définir un modèle de document pour exprimer facilement des structures organisationnelles, telles que nous les avons évoquées de façon informelle à la section précédente. C'est à rapprocher (en beaucoup moins ambitieux et validé bien évidemment) du rôle de XMI [OMG, 2000] dans le monde de la modélisation objet. Plus concrètement, les objectifs en sont :

- Pouvoir développer des ateliers de modélisation graphique de structures organisationnelles se basant sur une description non équivoque.
- Permettre le partage, l'échange et l'évolution de modèles d'organisation multi-agents.

- Favoriser une documentation intégrée de telles structures et la génération automatique d'information pour utilisateurs et concepteurs.
- Pouvoir intégrer des outils de contrôle ou de surveillance de sociétés d'agent, pour effectuer des analyses ultérieures (par exemple via les techniques pratiquées dans l'étude des réseaux sociaux[Degenne et Forsé, 1994]) ou pour boucler sur les outils de conception.
- Autoriser l'extension du langage de description pour ne pas se cantonner à un modèle générique et permettre au concepteur de rajouter des notions spécifiques à son domaine en raffinant la définition de document.

Modèle de description de structures organisationnelles

Nous avons donc défini une structure de document (appelée ici A-ORGML (Abstract Organization Markup Language) pour la représentation de structures organisationnelles multi-agents exprimées dans notre modèle[4]. Le lecteur se reportera en annexe B pour consulter la DTD précise de ces modèles de description.

Voici tout de suite un exemple d'expression : le toujours simpliste cas des agents joueurs de ping-pong. L'idée est de garder une expression la plus dépouillée possible pour des organisations simples.

```
<organisation-def>
  <group-def id="ping-pong">
     <role-def id="joueur">
       <constraint arity="2"/>
     </role-def>
  </group-def>
</organisation-def>
```

On retrouve dans le langage de description nos notions clé introduites dans ce chapitre : définition d'une structure organisationnelle (la balise `organisation-def`), les modèles de groupes et de rôles (`group-def` et `role-def`), ainsi que la possibilité de définir des contraintes au niveau générique, comme l'arité.

Expression de l'agence de voyage Pour une société d'agents de taille plus réaliste, il faut introduire d'autres définitions : pouvoir exprimer les interactions qui vont être définies entre rôles, laisser la possibilité d'écrire des contraintes plus précises (dans un modèle spécifique à l'application ou générique, comme OCL [Warmer et Kleppe, 1999]). Il faut également permettre au concepteur de raffiner ses expressions hors du modèle. On autorise des définitions extérieures pour des comportements associés aux rôles ou pour les interactions (comme des

[4]Notre objectif étant limité à agent-groupe-rôle, nous n'avons bien évidemment pas la prétention d'avoir une description complètement générique de modèles de SMA

jeux de règle, du code Java, des définitions en réseau de Petri). Voici ce que pourrait donner une expression dans A-ORGML du problème de l'agence de voyage :

```
<organisation-def id="SMA-voyage">
  <group-def id="clientele">
     <role-def id="touriste"/>
     <role-def id="agence">
       <constraint type="OCL">
          agent->hasRole("voyagistes","courtier")
       </constraint>
     </role-def>
     <interaction id="conseilVoyage">
        <endpoint role="touriste" type="initiator"/>
        <endpoint role="agence" arity="1"/>
     </interaction>
  </group-def>
  <group-def id="voyagistes">
     <role-def id="courtier"/>
     <role-def id="fournisseur">
        <definition type="java" ref="fournisseur.class"/>
     </role-def>
     <interaction id="fipa-contract-net">
        <endpoint role="courtier"/>
        <endpoint role="fournisseur"/>
     </interaction>
  </group-def>
  <group-def id="contrat">
     <role-def id="acheteur"/>
     <role-def id="vendeur"/>
     <role-def id="certificateur"/>
     <interaction id="paiement">
        <endpoint role="acheteur" type="initiator"/>
        <endpoint role="vendeur"/>
        <endpoint role="certificateur" type="optional"/>
        <definition type="petri" ref="protocole-signature.desc"/>
     </interaction>
  </group-def>
</organisation-def>
```

Remarquons dans cet exemple les interactions exprimées en fonction des rôles définis plus haut, et par le statut de ceux-ci (initiateurs, présents optionnellement, etc ...). Les balises `definition` renvoient à des documents externes pour une description plus précise des interactions dans des formalismes appropriés.

Modèle de description d'organisations

Dans le même esprit, nous avons défini une autre structure de document XML (celle-ci appelée C-ORGML : Concrete Organization Markup Language), complétant la première et permettant de décrire des instances d'organisations *effectives*.

Il d'agit ici de décrire les individus impliqués dans l'organisation concrète, les groupes et les rôles tenus, à un instant donné, cette description pouvant venir d'un outil de surveillance de système, d'exemples décrits à la main dans un outil de conception, de définitions d'organisations à instancier plus tard dans une plate-forme, etc..

Voici par exemple ce que donnerait en C-ORGML un système multi-agent concrétisé de joueurs de ping-pong :

```
<multi-agent-system>
  <individuals>
    <agent id="Nick"/>
    <agent id="Mike"/>
  </individuals>
  <organisation name="Gymnase">
    <group name="Table 4">
      <role name="joueur" handler="Nick"/>
      <role name="joueur" handler="Mike"/>
    </group>
  </organisation>
</multi-agent-system>
```

On trouve dans ce second format deux parties distinctes : la liste des agents participant au système, puis leur implication courante dans l'organisation. Il est possible également d'indiquer, en regard des différents groupes ou organisations identitifiés, sur quelle définition abstraite on se base (et qui sera évidemment décrite dans le format précédent).

Exemple de l'agence de voyage Voici ce que donnerait une concrétisation particulière de l'agence de voyage, en liaison avec la définition que nous avons présenté plus haut :

```
<multi-agent-system>

  <individuals>
    <agent id="touriste-ol" type="touriste-def.lisp">
      <property name="etat-interne" value="file:/agent42.txt"/>
    </agent>
    <agent id="touriste-lou"/>
    <agent id="agence-montmartre"/>
    <agent id="air-france">
      <property name="Agent-UID" value="fipa://afparis/dkw47lp5"/>
    </agent>
    <agent id="versum-airways"/>
    <agent id="sncf"/>
  </individuals>

  <organisation name="Test de SMA No. 12" def-id="SMA-voyage">
    <group name="Dans l'agence" def-id="clientele">
      <role name="touriste" handler="touriste-ol"/>
      <role name="touriste" handler="touriste-lou"/>
```

```
        <role name="agence" handler="agence-montmartre"/>
      </group>
      <group name="place de marché" def-id="voyagistes">
        <role name="courtier" handler="agence-montmartre"/>
        <role name="fournisseur" handler="versum-airways"/>
        <role name="fournisseur" handler="air-france"/>
        <role name="fournisseur" handler="sncf"/>
      </group>
      <group name="signature7" def-id="contrat">
        <role name="acheteur" handler="touriste-lou"/>
        <role name="vendeur" handler="air-france"/>
      </group>
    </organisation>
  </multi-agent-system>
```

7.4 Le processus méthodologique

7.4.1 Découpage

Classiquement, le processus de construction d'une application s'effectue en trois phases : analyse, conception, réalisation. Nous allons détailler dans cette section comment interviennent nos modèles organisationnels dans chacune de ces phases.

Analyse : Ce sont les phases d'analyse fonctionnelle, d'analyse des dépendances, d'identification des communautés et contextes pour préparer la conception des groupes, ainsi que du choix des mécanismes de coordination et d'interaction et éventuellement de modèle d'environnement. D'autres méthodologies d'analyse spécifiques au domaine ou génériques interviennent à cette étape, comme par exemple CASSIOPÉE [Drogoul et Collinot, 1998].

Conception : C'est la construction du modèle organisationnel : groupes, rôles. Puis sont définis des schémas d'interactions : protocoles, messages. Cela comprend également la définition des autres entités manipulées par les modèles spécifiques (actions, tâches, objectifs, ...)

Réalisation : C'est le choix d'architectures d'agents en fonction du résultat des phases précédentes, ainsi que les stratégies d'implémentation retenues pour les protocoles d'interaction et les entités du domaine.

7.4.2 Conception de la structure organisationnelle

Nous pouvons maintenant détailler les étapes du processus méthodologique correspondant à nos modèles.

Identification des groupes

Pour isoler des groupes potentiels à cette étape du processus, il nous faut disposer de critères de formation d'un groupe. Le premier critère, qui est en fait le seul pré-requis indispensable, est l'existence d'un mécanisme de communication commun au sein du groupe potentiel. La définition primitive du groupe étant centrée sur l'interaction entre les membres, une transgression de ce critère est signe d'un mauvais départ dans le design de l'application. Les autres critères d'isolation d'un groupe sont soit inhérents à l'application (structuration d'une activité commune), soit relatifs aux contraintes d'architectures, de paradigmes de communication (rassemblement autour d'une propriété commune), ou d'espace de travail du concepteur (masquage d'un sous-système par les mécanismes de représentant).

Dans des groupes encore non identifiés, ou plus exactement non séparés en agents individuels, le modèle holonique est un bon candidat au dégagement d'une hiérarchie de perception ou d'action entres groupes. Pouvoir exprimer à ce niveau un groupe comme étant un agent individuel afin de faire abstraction d'une partie de l'organisation est une aide conceptuelle indéniable offerte *à l'analyste* [5], étant entendu que le système concret est mis en oeuvre sous forme d'une société d'agents, donc conforme au modèle agent-groupe-rôle, ou à une de ses déclinaisons.

Choix ou conception d'un modèle organisationnel spécifique

La deuxième étape de l'identification des structures de groupes est la séparation des rôles au sein des groupes ainsi isolés. Il est à noter que l'on se retrouve face à un compromis classique : garder un rapport correct entre la complexité individuelle et le nombre des structures de groupes à définir. En considérant des cas extrêmes, on pourrait construire une application de taille respectable en un seul groupe qui contiendrait l'ensemble des rôles et interactions rencontrées, ou à l'opposé en une multitude de groupes, chacun n'ayant que deux rôles et un seul scénario d'interaction.

Spécification de la structure organisationnelle

La spécification de la structure de groupe se fait alors dans un des modèles spécifiques relatifs à l'application (ou directement dans le modèle générique). A cette étape, les contraintes de l'application identifiables au niveau social sont exprimées à travers les fonctions de rôles.

Dans l'exemple de la figure 7.9, une notation en flèche pleine exprime une fonction de rôle particulière (dépendance sur l'appartenance préalable à un autre rôle).

[5] Voir par exemple [Parunak et Odell, 2001] pour une fusion des modèles holoniques et agent-groupe-rôle

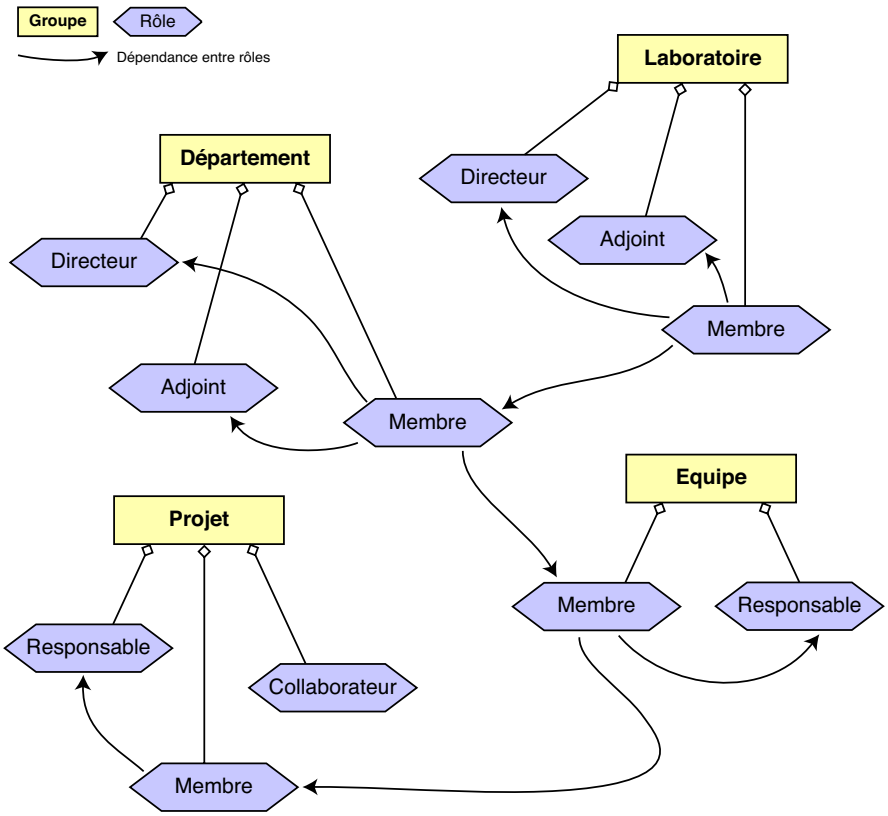


FIG. 7.9 – Exemple de spécialisation d'un modèle organisationnel

Description des schémas d'interactions

Les protocoles d'interactions sont alors précisés entre les rôles identifiés. Ces modèles de protocoles sont exprimés dans le formalisme propre au modèle organisationnel dans lequel est décrite la structure de groupe. Ce formalisme peut être des diagrammes de séquence UML, un réseau de Petri définissant le protocole, des scenarii ACL [FIPA, 1997a] ou KQML [Finin et Fritzson, 1994], ...

Définition des architectures individuelles

Le choix de l'architecture est alors contraint par les rôles possibles pouvant être joués par un type donné d'agent, en particulier pour l'implémentation des différents schémas d'interactions. On cherche ici à restreindre en fonction des rôles identifiées, la gamme des architectures possibles.

C'est à cette dernière étape que se place la frontière avec les approches centrées agent, et que le relais se fait avec la conception de l'architecture interne.

7.4.3 Remarques

Il convient de noter que pour exprimer une contrainte de conception au niveau du modèle générique, nous ne pouvons utiliser **que** les concepts présents à ce niveau, c'est à dire encore une fois les seules notions d'agent "boîte noire", de groupe et de rôle. Il est impossible d'utiliser telle ou telle propriété des architectures choisies pour nos agents puisqu'à ce niveau de description, nous ne connaissons pas cette information (et il y a de forte chance qu'à cet instant de la conception, elle ne soit simplement pas encore arrêtée).

Est-ce à dire que nous sommes arrêtés par une trop grande généricité et incapables d'exprimer quelque contrainte que ce soit sans passer à un modèle spécifique ? La réponse est négative : nous pouvons continuer à manier notre trio de concepts et surtout *toute relation structurelle* entre eux.

La question de l'arité en est une bonne illustration : cette notion simple contraint la relation entre le rôle et les instances d'agents qui vont les jouer lors de la concrétisation de l'organisation. Imaginons les expressions suivantes :

- Les groupes **commerce** ne peuvent avoir qu'un seul *broker* si le nombre d'agents ayant un rôle *client* est inférieur à 10, et trois au-delà.
- Il ne peut y avoir qu'un *pilote* dans le groupe *cockpit* et au plus 100 agents ayant le rôle *passagers* dans le groupe *cabine*

7.5 Conclusion

Nous avons présenté dans ce chapitre plusieurs guides méthodologiques pour la mise en oeuvre du modèle agent-groupe-rôle dans une conception, que cela soit pour exprimer des structures sociales abstraites, dans un contexte de représentation et notations, ou encore pour faire la lien avec d'autres modèles et méthodes.

La modélisation organisationnelle ainsi faite ne disparait pas pour autant dans les systèmes finaux. En effet, ses effets sont présents, que ce soit de façon implicite par les représentations internes de l'organisation ou des schémas d'interaction dans les agents implémentés, ou encore de façon explicite dans le cas où les outils d'implémentation proposent directement ces concepts.

L'enjeu est de dégager des systèmes multi-agents ainsi décrits, des *patterns de conception* organisationnels, c'est à dire des descriptions *éprouvées* d'organisations classiques (réseaux contractuels, organisations hiérarchiques, structure de représentant, ...) pour en faciliter la réutilisation. Evidemment, un catalogue de solutions éprouvées est encore chose lointaine dans le domaine des systèmes multi-agents, mais exprimer de telles structures est un premier pas nécessaire, comme le prouve le travail autour d'extensions agent à UML [OMG, 1999a]. Dans le cadre des applications et prototypes développés durant ce travail, l'évolution rapide tant du modèle sous-jacent que de la plate-forme n'a bien sûr pas permis

d'appliquer *a priori* cette approche, mais de la faire construire itérativement en fonction des expériences et retours.

On pourrait enfin aborder dans le thème de ce chapitre la liaison entre conception et spécification formelle. On peut se référer sur ces aspects à [Hilaire et al., 2000] pour un système proposant des descriptions formelles en Object-Z et un prototypage associé, et reposant sur un modèle organisationnel proche d'agent-groupe-rôle.

Chapitre 8

Evaluation et applications

Ce chapitre présente l'évaluation de ce travail, faite sur plusieurs axes.

D'une part, nous élaboré un jeu de mesures sur l'aspect purement implémentatoire de ce travail, afin d'évaluer la performance et la légèreté du noyau de notre plate-forme. Cela se justifie par le souci d'offrir effectivement un substrat de mise en oeuvre suffisamment souple pour ne pas orienter vers un seul type d'application. Connaître les domaines de performances d'éléments comme la communication ou la charge d'un très grand nombre d'agents était donc crucial.

D'autre part, nous avons voulu valider la généricité architecturale du modèle et de la plate-forme en implantant plusieurs modèles d'agents "classiques", de la mobilité à la simulation en passant par les architectures à base de règles. L'objet n'est évidemment pas d'offrir un kit de programmation évolué au point de lutter avec des plate-formes spécialisées, mais d'avoir des indications sur la faisabilité de telles bibliothèques. Nous verrons que dans certains cas, nous aboutissons à des frameworks agents en fait tout à fait utilisables même au delà de simples exemples.

La dernière évaluation à faire porte sur la généricité applicative de ce modèle et de la plate-forme. Dans ce cas précis, nous ne pouvions guère être juge et partie en implémentant nous-même les applications concernées. Nous présenterons donc quelques applications extérieures (industrielles ou académiques) basées sur notre travail.

Nous évoquerons également une application particulière, SEDIT [Gutknecht et al., 2001], qui est un outil multi-formalismes d'aide à la conception ou au développement d'agents et montrerons en quoi son architecture même est un bon exemple d'utilisation de l'approche agent comme base méthodologique et implémentatoire.

8.1 Mesures primitives

Cette première section regroupe une évaluation des mécanismes de base de MADKIT. Nous avons choisi de mesurer les aspects suivants :

– Performance des mécanismes primitifs de communication.
– Tenue en charge au nombre d'agents en termes d'occupation mémoire.
– Tenue en charge au nombre d'agents en termes de temps d'amorçage.

Nous comparons dans ces mesures les deux politiques primitives d'exécution d'agent (asynchrone ou par ordonnanceur externe), et l'utilisation dans un cadre local ou distribué.

Sauf mention contraire, le noyau MADKIT est configuré pour tourner seul, en mode console, avec uniquement les classes correspondantes aux agents de test. La machine virtuelle Java est réglée sur une valeur maximale de tas de 256 Mo. L'occupation mémoire des instances est obtenue par la différence `totalMemory` - `freeMemory` renvoyée par le runtime Java.

8.1.1 Occupation mémoire

Protocole de test

Le premier facteur est la consommation de ressource mémoire selon le type d'agent. Le protocole de test consiste à démarrer 15000 agents sur un noyau MADKIT. Le code de test est extrèmement simple : un agent de test se charge d'instancier et demander l'activation des agents, réduits à leur plus simple expression, sur l'un et l'autre modèle.

```
for (i=0; i<max; i++)
{
  if (test_reactive)
     launchAgent(new MiniReactive(i), this, false);
  else
     launchAgent(new MiniThreaded(i), this, false);
}
```

```
public class MiniThreaded extends Agent
{
  public void live()
  {
    waitNextMessage();
  }
}
```

```
public class MiniReactive extends AbstractAgent
{
  public void doIt()
  {
  }
}
```

Les classes `MiniThreaded` et `MiniReactive` ne font qu'hériter des classes `Agent` et `AbstractAgent`, respectivement.

Résultats

Première approche Notre première figure (8.1) montre le tracé de la mémoire occupée en Ko par les instances d'objets dans la machine virtuelle en fonction du nombre d'agents en fonctionnement sur la plate-forme.

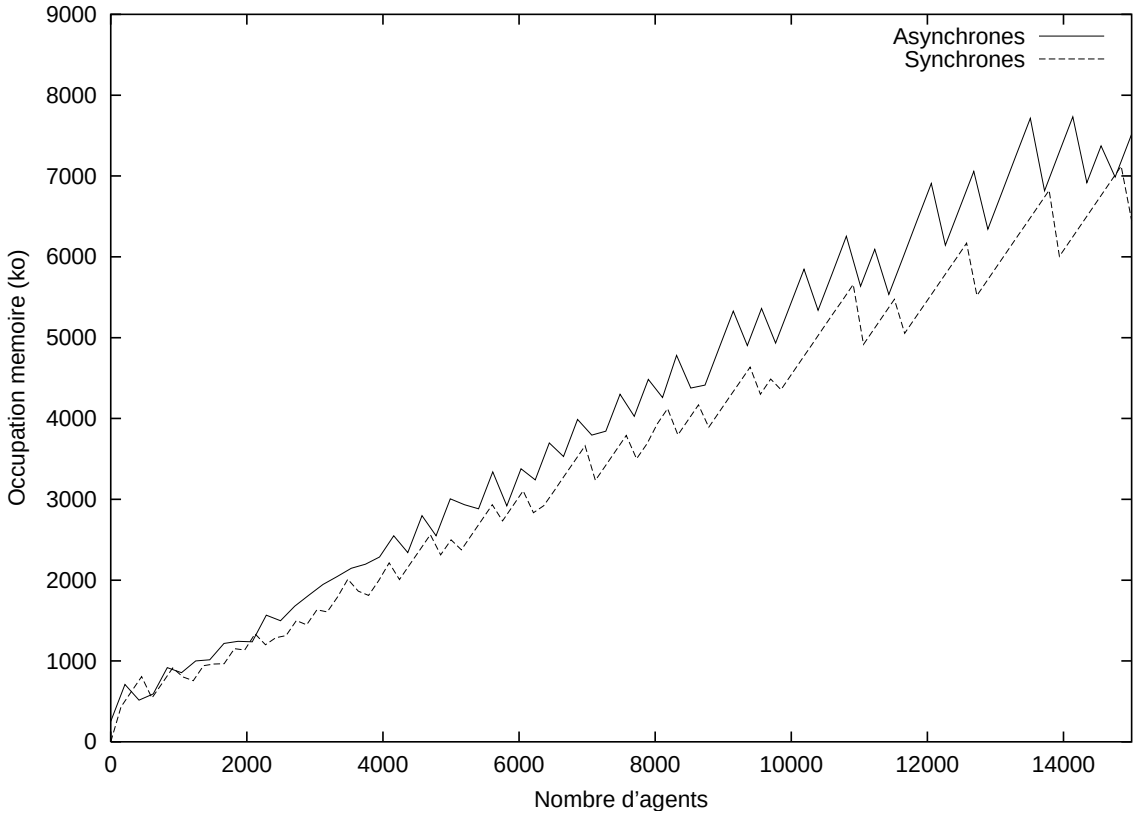


FIG. 8.1 – Consommation mémoire par rapport nombre d'agents

On constate que l'écart n'est pas particulièrement significatif. La consommation supplémentaire de ressource est d'environ 15 % pour un agent asynchrone par rapport à un agent synchrone. L'aspect en dents-de-scie s'explique par le déclenchement du ramasse-miette et par l'augmentation dynamique de la taille du tas qui interviennent l'un et l'autre à intervalles réguliers.

Prise en compte des piles Néanmoins, cette mesure a une faille. D'abord, on ne tient compte ici que des resources mémoires *à l'intérieur de la machine virtuelle*. Or, les mécanismes d'implantation des threads ont leur influence par la présence des piles d'appels interne

(Java) et externe (C) associées à chaque thread, qui rajoute un coût net à chaque processus concurrent. Une fois prises en compte les piles (mesurées en externe au niveau du processus de la machine virtuelle), ce qui revient à une correction d'environ 5 Ko par thread[1], on obtient la figure 8.2.

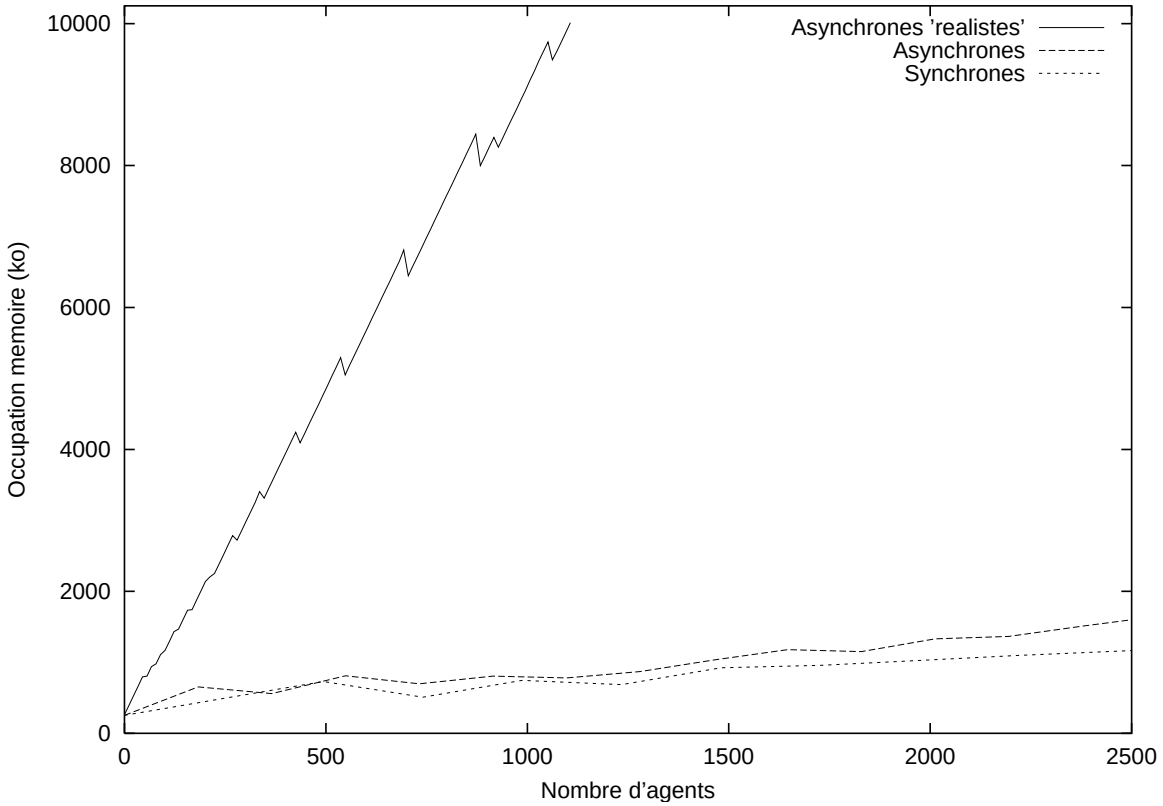


FIG. 8.2 – Correction avec prise en compte des piles

Cas réaliste Bien qu'ayant une vue maintenant plus précise de la situation, force nous est d'avouer que ces mesures sont encore bien éloignées de la vérité. En choisissant d'implanter des agents "vides", nous nous plaçons dans une situation outrageusement optimiste. Aucune information n'étant renvoyée, et aucun message n'étant traité, les structures internes des agents asynchrones (flots et boîtes aux lettres) restent non initialisées. Après avoir corrigé en ce sens nos agents de test pour obtenir une sortie `foo` à l'écran et l'envoi d'un message `bar`, nous avons alors un cadre un peu plus représentatif d'un agent "classique". La figure 8.3 reflète ce changement. La charge occasionnée par les agents asynchrones croît alors d'un facteur 5.

[1] ce qui est sans doute une évaluation optimiste, vu qu'en situation réelle, la machine virtuelle peut réserver jusqu'à 400 Ko pour la pile Java et 128 Ko pour la pile C

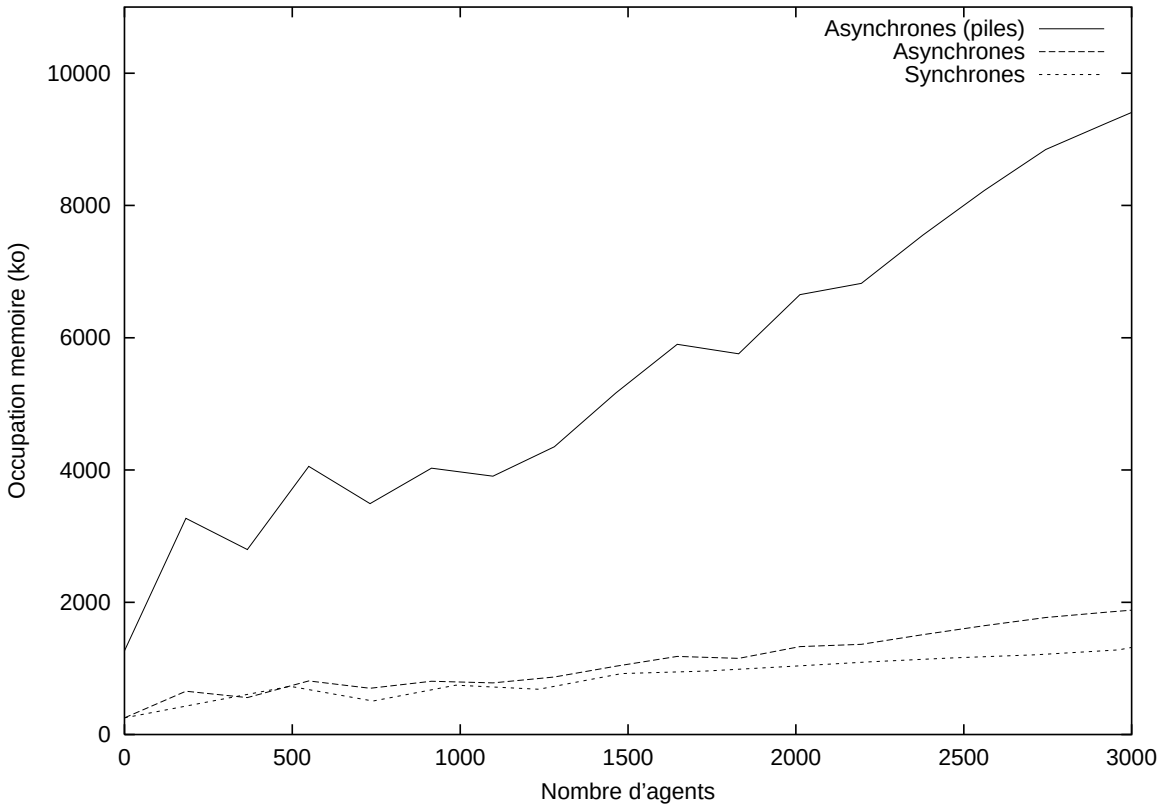


FIG. 8.3 – Correction après mise en place de flots et message

8.1.2 Temps d'amorçage

Présentation

Nous avons ensuite évalué la différence entre démarrages d'agents synchrones et asynchrones. Le test est constitué de la reprise de nos agents minimalistes précédents. Nous mesurons les temps cumulés nécessaires au démarrage *effectif* d'agents lancés par blocs de 500. Pour être certains de mesurer le temps de mise en situation, la mesure est faite dans le code même de l'agent lancé.

Le code utilisé est le même que dans le cas précédent.

Résultats

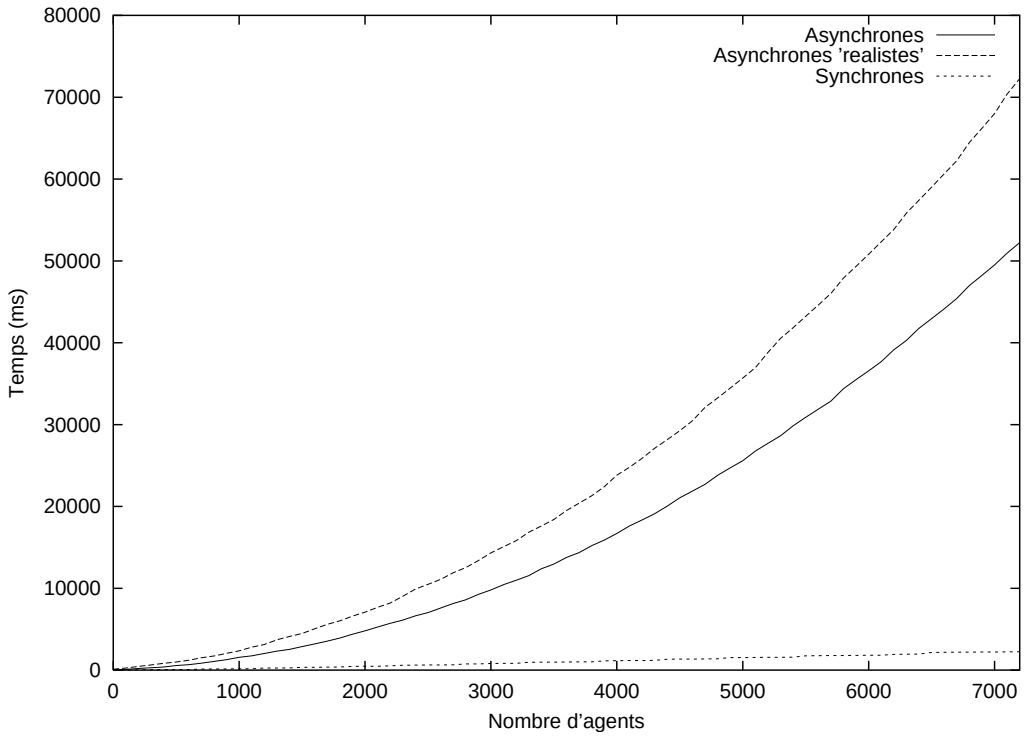


FIG. 8.4 – Temps d'amorçage

La figure 8.4 montre les temps nécessaires pour le lancement de jusqu'à 7000 agents synchrones, asynchrones et "asynchrones réalistes". La figure 8.5 reprend la mesure pour 20000 agents asynchrones et 100000 asynchrones.

Là encore, la différence est flagrante entre agents synchrones et asynchrones, en particulier au niveau de la croissance linéaire dans le cas des agents synchrones, au contraire du cas des asynchrones où tout nouvel agent ne fait que ralentir le fonctionnement de ceux précédemment présents.

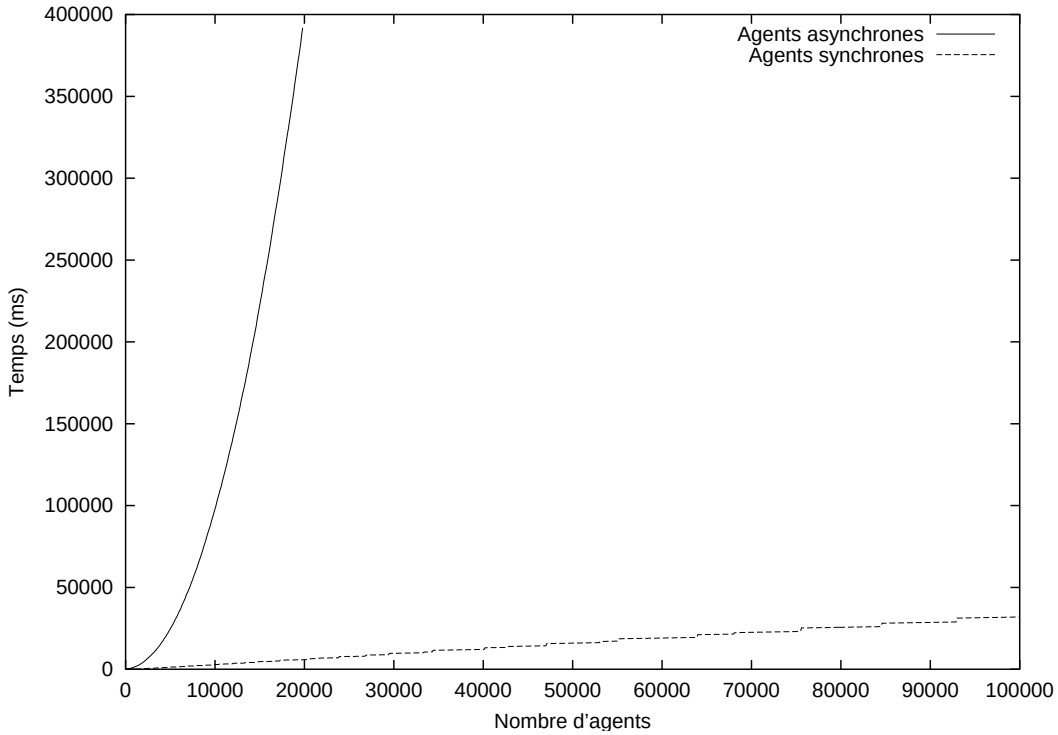


FIG. 8.5 – Temps d'amorçage

Remarques

Autant l'asynchronisme reste confortable au niveau de la conception d'un système multi-agent, autant la pratique peut s'avérer délicate dans le cadre d'applications nécessitant un grand nombre d'entités (par exemple les problèmes de simulation multi-agents).

Les deux solutions que nous y voyons sont d'une part d'avoir des mécanismes de contrôle permettant au concepteur un passage de l'asynchrone au synchrone le plus facile possible, et d'autre part de permettre le passage au distribué dès que possible.

Sur ce point, la transparence pour l'agent du lieu d'exécution et de son environnement social, qui est assurée par les groupes distribués dans MADKIT, est une caractéristique bienvenue.

Il est également à noter que pendant ces tests nous avons touché aux limites des machines virtuelles Java, voire même des machines physiques elle-mêmes[2], principalement par le nombre de threads concurrents présents. Les concepteurs de machines virtuelles Java choisissent la plupart du temps entre deux grandes types de représentations des threads.

– Une première méthode fait une association directe des threads Java sur les threads

[2]Pourtant dans notre cas un robuste bi-processeur Pentium III à 500 Mhz doté d'un respectable 256 Mo de mémoire vive

natifs fournis par l'OS (tels que des Winthreads ou des threads POSIX) : dans ce cas, le système d'exploitation sous-jacent a rapidement trouvé excessif la masse de nos 50000 processus a gérer.
– L'autre approche consiste en un mécanisme d'émulation de processus concurrents *à l'intérieur* du processus de la machine virtuelle elle-même. Dans ce cas, le nombre de processus systèmes reste correct (1 seul), mais ce sont la pile Java et la pile C native associée à chaque thread qui provoquent notre chute.

8.1.3 Performance de la communication

Protocole de test

Dans cette série de tests, nous avons vérifié le bon comportement de système d'envoi de message, par des mesures de temps de propagation d'un agent à l'autre. Le test constitue à envoyer des rafales de 500 messages similaires entre deux agents et de mesurer le temps d'un aller-retour. A chaque groupe d'envoi, le contenu des messages (des tableaux d'octets aléatoires) devient de plus en plus massif.

Nous avons effectué la mesure dans le cas d'un noyau seul, d'un noyau lancé en mode G-Box, et d'un noyau en mode G-Box, avec une synchronisation faite avec une plate-forme distante.

Résultats

Test local La figure 8.6 donne les temps de propagation de messages *locaux* entre agents situés dans la même plate-forme. Le mode de passage sans recopie dans le noyau se vérifie : les temps de propagation restent constants malgré l'augmentation d'une taille de message.

Malgré le temps moyen un peu plus important dans les modes G-Box (explicables par la charge mémoire de la machine virtuelle et l'activité plus importante du ramasse-miette), on remarque que la présence d'un `Communicator` interceptant les messages non locaux n'a pas d'influence sur le temps de propagation local. On note simplement un léger ralentissement sans doute du à l'aiguillage local/distant des messages au sein du noyau.

Nous vérifions effectivement que le temps de propagation en nombre reste linéaire, et que cela n'a pas d'effet à long terme sur la consommation mémoire du noyau, ce qui est essentiel pour la stabilité et la mise en application de la plate-forme.

Test en distribué Nous avons ensuite repris ce test dans le cadre distribué. Les deux agents se trouvent maintenant sur deux moyaux MADKIT distinct, avec les agents de gestion de la communication pour assurer la distribution des messages. La première mesure faite est un test sur deux noyaux, mais sur une seule machine physique, afin d'évaluer uniquement

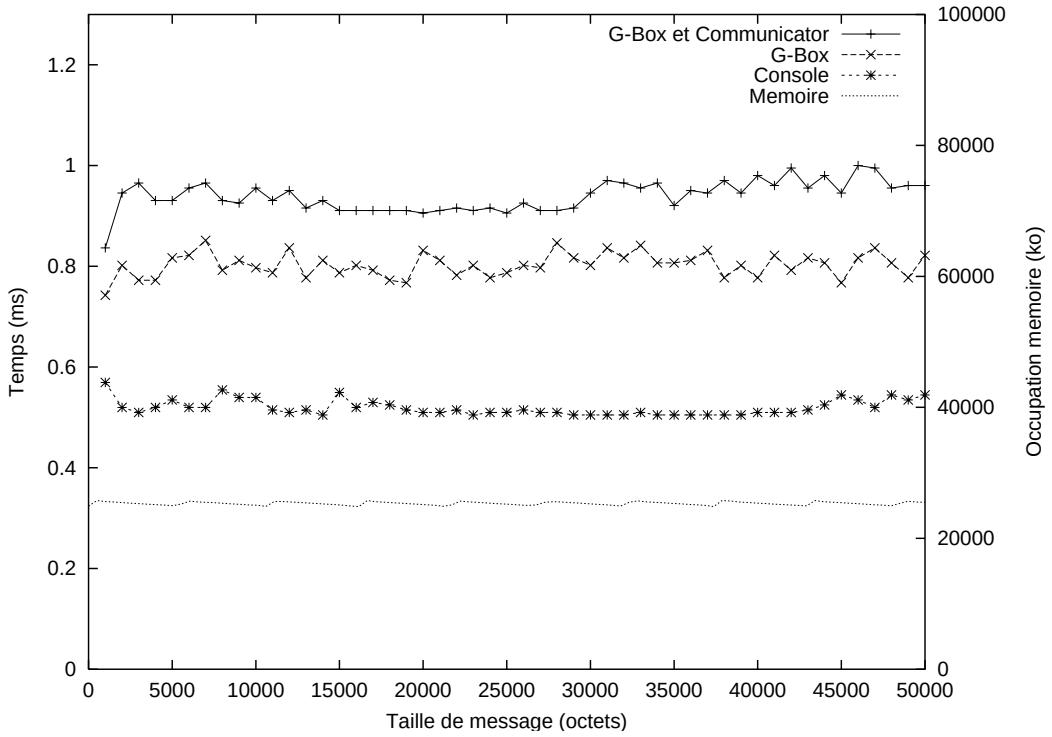


FIG. 8.6 – Temps de transmission de messages

l'impact du passage de MADKIT en distribué. La seconde mesure est faite sur deux machines distinctes raccordées sur le même réseau local.

Dans ces tests, le passage de message local et performant s'arrête donc forcément aux agents de communications qui ont la charge de la sérialisation, de l'envoi et de la désérialisation du message. On constate comme prévu que l'envoi de message passe en $O(n)$ sur la taille du message. Notons aussi que le prix d'un message distribué reste abordable : un message court (que nous avons constaté être la majeure partie des interactions dans des systèmes classiques) ne prend "que" 20 ms.

8.2 Extensions et architectures agents spécialisées

L'objectif de cette section est d'illustrer comment le modèle d'agent "minimaliste" de notre modèle et de la plate-forme MADKIT peut être étendu et s'incarner dans des architectures plus classiques. Nous n'allons évoquer que deux des bibliothèques d'agents que nous avons réalisées : simulation distribuée et agents mobiles.

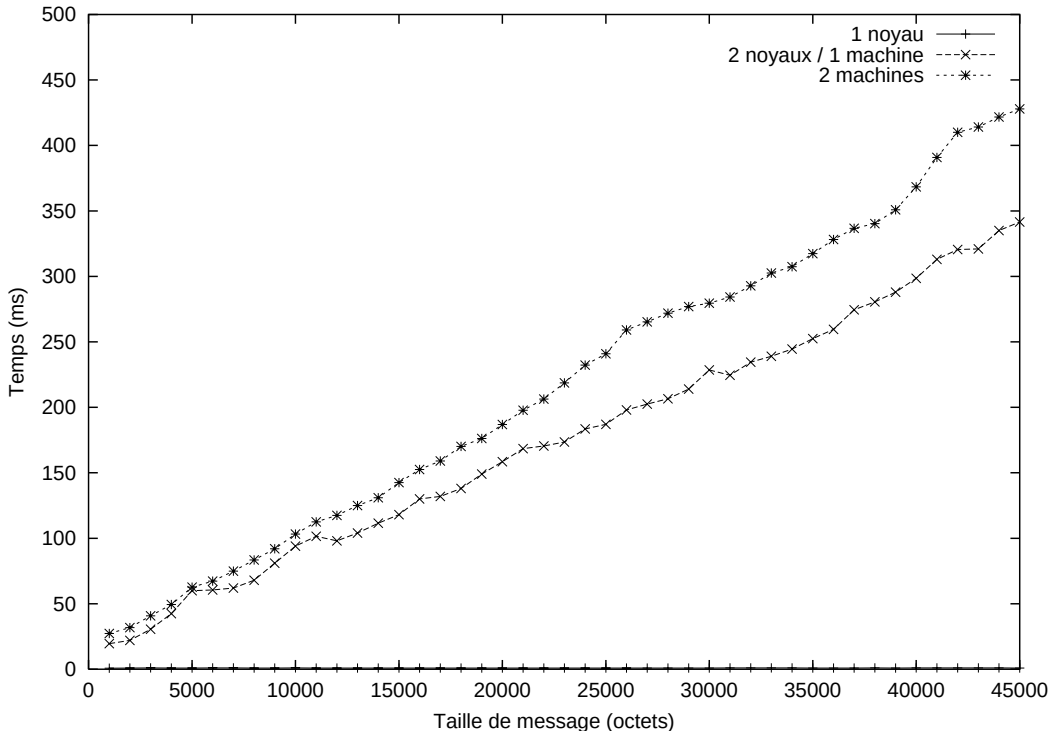


FIG. 8.7 – Temps de transmission de messages

8.2.1 Simulation multi-agents

L'étude que nous présentons ici relève avant tout du problème de la simulation distribuée avec ses contraintes de gestion de la charge et de synchronisation, et de leur traduction dans un modèle de simulation. Nous abordons également les questions de simulation modulaire, et en particulier la gestion de différents domaines, temps ou espaces de simulation. Ce travail a été réalisé tôt dans le développement de l'outil, en collaboration avec Emmanuel Lieurain, et réétudié plus tard avec l'aide de Pierre Bommel et Fabien Michel.

Comme le fait remarquer Van Dyke Parunak dans [Parunak et al., 1998], les systèmes multi-agents se prêtent naturellement bien à une utilisation dans un mode distribué. Néanmoins, l'application de cette hypothèse répandue aux simulations par systèmes multi-agents peut s'avérer décevante : la simple répartition du calcul est loin de garantir automatiquement des temps de simulations plus courts. Une conception inadaptée de la simulation pourra conduire à un nombre important de messages de partage d'informations entre agents simulés ou environnement transitant par le réseau, causant dans certains cas une exécution plus lente qu'une simulation non distribuée.

Dans ce cadre, nous avons proposé [Gutknecht et al., 1998] un modèle de conception de simulations environnementales pouvant faciliter la mise en place de simulations dis-

tribuées. De plus, nous pensons qu'un lien peut être fait entre ce modèle et deux thèmes actuels des simulation par SMA, à savoir la gestion de l'hétérogénéité des échelles spatiales [Servat et al., 1998], et le rapport entre plusieurs bases de temps [Fianyo et al., 98].

Les problèmes auxquels nous nous rattachons sont donc : d'une part la combinaison de sous-modèles de simulation fonctionnant à des échelles de temps ou d'espaces différents (par exemple : répartition du calcul selon zones, interactions de simulation animal/végétal), dont découle le problème de l'expression de synchronisations et des échanges nécessaires à la cohérence entre sous-modèles ; et d'autre part, à l'intégration modulaire de sous-modèles de simulation : comment gérer l'interdépendance entre sous-modèles dans une architecture "ouverte" (dans laquelle on puisse ajouter/retirer ou modifier des sous-modèles).

Nous nous basons sur deux hypothèses :

- L'environnement, vu comme le support des ressources et l'espace d'action des agents, est lui-même défini comme un ou des agent(s) particulier(s) encapsulant un certain modèle de simulation. Les interactions des entités simulées avec le monde seront donc définies comme des échanges d'information entre agents.
- Une simulation par systèmes multi-agents peut être découpée en plusieurs groupes de simulation, cohérents vis-à-vis d'un modèle, d'une base de temps, ou d'une zone spatiale.

Ces hypothèses nous conduisent à l'expression d'une structure organisationnelle pour la simulation multi-agent, basée sur le modèle agent-groupe-rôle.

Une structure organisationnelle de simulation

Nous nous proposons d'appliquer le modèle à la conception d'une simulation multi-agents. Le principe est d'envisager l'environnement (dans son contexte de support de simulation) comme étant lui-même un système multi-agents. Chaque groupe correspond donc à un sous-modèle particulier du modèle complet à simuler.

Les avantages d'une telle décomposition sont les suivants :

- La spécification préalable de groupes de simulation permet d'exprimer plus finement l'ensemble des interactions nécessaires entre ces zones, et d'évaluer précisément les messages nécessaires, les synchronisations et donc l'impact sur un fonctionnement en mode distribué.
- L'hétérogénéité des modèles apparaîssant dans la simulation est rendue explicite, et une conception incrémentale des différents groupes possible.

L'application utilisée pour tester ces approches a été une implémentation du problème "SugarScape" [Epstein et Axtell, 1996] dans la plate-forme MADKIT. Un premier travail a mis en évidence la nécessité d'avoir une évaluation précise des messages distribués et de leur répartition entre agents pour pouvoir déterminer l'intérêt et la méthode de distribution du calcul. Une deuxième phase a conduit à l'intégration d'un moteur pour agents réactif que

nous avons décrit au chapitre 6.

Chaque structure de groupe de simulation comprend :

- Un agent ayant le rôle d'*environnement*, qui reçoit les demandes d'action et perception des agents, et en calcule les effets. Nous nous plaçons dans le contexte d'un modèle à influences / réaction décrit notamment dans [Ferber et Muller, 1996].
- Des agents ayant des rôles d'*entités simulées*. Ces rôles définissent un ensemble d'interactions valides avec l'environnement. A chaque type d'entité simulée correspond un rôle particulier.
- Optionnellement, un agent ayant le rôle de *scheduler* responsable de l'évolution du temps au sein du groupe de simulation. Ce rôle peut éventuellement être tenu par l'agent ayant déjà le rôle d'environnement.

Lors de cette étape de conception de la simulation, on spécifie indépendamment les différentes structures de groupe de simulation impliquées. Ceci correspond à la définition de types (rôles) pour les entités simulées, l'explicitation des interactions possibles entre chacun d'eux et le rôle d'environnement, ainsi que la définition d'architectures d'agents qui jouent ces rôles. L'architecture de l'agent "environnement" est également définie à cette étape.

La simulation par multi-groupes

A partir de l'ensemble des groupes de simulations, il est alors possible de construire une structure de groupe définissant les relations entre les différents agents qui ont un rôle d'environnement (pour gérer les recouvrements entre partitions de l'environnement global, etc.). Le but est de disjoindre le problème simulé (présent dans les groupes de simulation) de la gestion de la simulation. Cette gestion est définie dans une structure de groupe, qui spécifiera les interactions entre plusieurs agents ayant un rôle d'environnement dans leurs groupes de simulation respectifs. Ils agissent en représentants de leur environnement local dans ce groupe global.

On peut dégager deux manières principales d'associer les divers groupes de simulation :

Globale : les agents " environnement " de chaque groupe appartiennent tous à un même groupe où se déroulent les échanges d'informations entre zones de simulation ou les synchronisations. On n'est en présence de deux niveaux de groupes seulement.

Hiérarchique : un agent d'environnement d'un groupe de simulation peut être lui-même simple objet de simulation dans un groupe de niveau plus général, et c'est avec les entités simulées de ce groupe qu'il acquiert les informations nécessaires au fonctionnement de sa propre zone, en fonctionnant selon le schéma entité - scheduler- environnement.

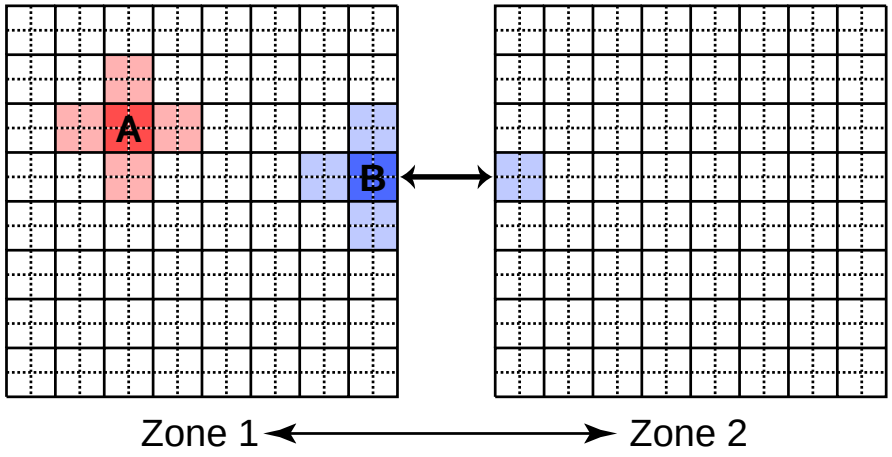


FIG. 8.8 – Division spatiale pour simulation distribuée

Pourquoi diviser en multi-groupes ?

L'intérêt est d'obtenir une clarification des dépendances du modèle initial, en encapsulant chaque sous-modèle dans un groupe de simulation, et de faciliter la mise en application en mode distribué, puisque chaque groupe de simulation instancié est conforme à une spécification commune : les messages nécessaires peuvent être catégorisés et évalués entre messages de contrôle de l'ordonnancement, interactions relatives au modèle d'environnement, et interactions entre entités situées.

La division elle-même peut être faite via plusieurs approches :

Division spatiale c'est la plus naturelle, et celle qui a déclenché cette étude. Il s'agit de diviser la simulation en zones spatiales utilisant le même modèle de calcul, et donc la même structure de groupe de simulation.

Par exemple, dans la figure suivante, l'interaction entre l'agent B et l'agent d'environnement Env_1 pour connaître l'état de son voisinage déclenche un échange d'information entre Env_1 et Env_2 La même action prise par l'agent A n'entraine aucun échange (figure 8.8).

Division en types de modèles L'idée est ici de segmenter la simulation en plusieurs mécanismes de calculs selon la nature des entités simulées. On se situe donc dans le cadre de modèles hétérogènes. C'est dans ce cadre que l'on envisage la division en multi-groupes pour des échelles temporelles ou spatiales différentes. Dans cet exemple, on a deux entités simulées F et H dans un groupe d'environnement. L'agent F s'avére aussi être l'agent ayant le rôle d'environnement dans un autre groupe de simulation correspondant à une zone d'échelle différente, qui comprend lui-même 4 entités simulées, F_1 F_2 F_3 F_4 (figure 8.9).

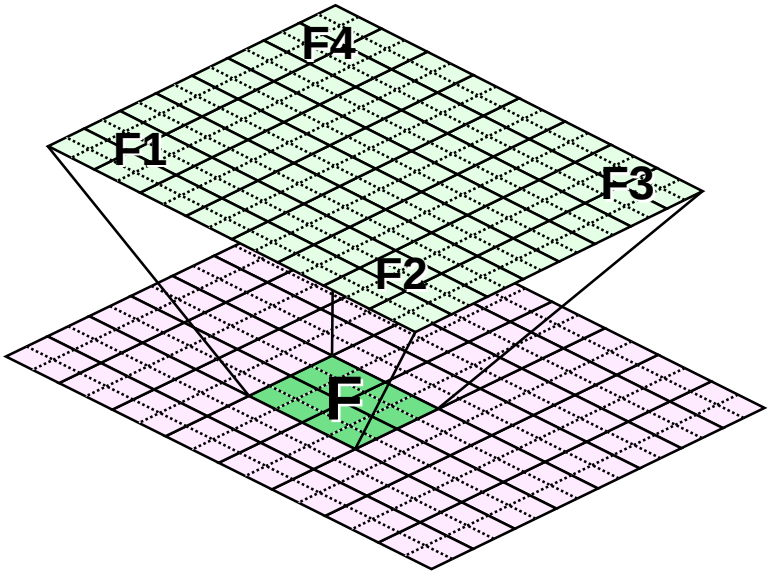


FIG. 8.9 – Division spatiale pour simulation distribuée

La mise en place de simulations multi-agents distribuées n'est généralement pas directe. L'impact des transmissions des informations partagées entre environnements est crucial et peut annuler le gain obtenu par la répartition du calcul, la prise en compte d'une approche non centralisée, telle que notre proposition d'environnement par multi-groupe, apparaît donc nécessaire dans la conception de la simulation. Nous pensons que cette approche pourrait être adaptée aux problèmes de simulation sur plusieurs échelles ou modèles : chaque sous-modèle cohérent par rapport à une échelle de simulation est conçu comme un groupe de simulation indépendant. La liaison entre ces groupes est faite d'une part par les agents ayant les rôles d'environnement dans l'un et l'autre groupe, et qui se communiquent des informations pour maintenir l'unité de la simulation, et d'autre part par les entités simulées, qui peuvent également faire partie de plusieurs groupes simultanément.

Un effet additionnel d'une séparation en multi-groupe est de pouvoir modifier indépendamment un groupe de simulation. En supposant qu'une zone utilise un modèle et qu'une évaluation est nécessaire avec un autre modèle sur cette zone, le groupe entier peut être retiré et remplacé à partir du moment où les interactions au niveau des agents d'environnements restent conformes à la spécification initiale. Le lien de représentant apparaît ici comme un mécanisme d'abstraction du modèle de simulation (par exemple, l'évaluation croisée d'un sous-modèle avec un modèle continu et d'un modèle basé sur les individus).

8.2.2 Agents mobiles

Un autre exemple d'architecture d'agent très spécialisée est celui des agents mobiles [White, 1996]. Dans ceux-ci, on ne s'attache guère à la construction d'un moteur d'action ou de raisonnement évolué ; par contre l'accent est mis sur la capacité de migration des agents, de reprise sur exécution ou de sécurité.

Cette architecure étant assez éloignée du modèle de base de MADKIT, il nous a paru intéressant d'étudier comment une bibliothèque d'agent mobiles pouvait être implémentée à l'aide de nos structures organisationnelles. Nous allons voir que cela en fait un exemple intéressant d'utilisation du niveau meta dans la plate-forme et le modèle.

Nous sommes partis de l'hypothèse suivante : tous les agents d'un système ne vont pas forcément vouloir obtenir des capacités de migration. L'expression de la mobilité agent sous forme de groupe spécialisé permet d'utiliser le même formalisme standard de groupe et rôle, avec la mobilité gérée en fait via deux groupes. Chaque site d'exécution contient un groupe local *mobilité* qui contient tous les agents itinérants (c'est à dire potentiellement mobiles) du site, ainsi qu'un agent outil ayant le rôle particulier de *migrateur*. Pour tenir le rôle d'*itinérant* dans ce groupe, l'agent doit avoir la compétence d'être sérialisable, ce qui définit la fonction d'acceptation du rôle : un agent non sérialisable ne sera pas admis dans le groupe. Le seul agent tenant le rôle de *migrateur* a la charge de faire migrer les agents de ce groupe qui le demandent vers un autre point d'exécution. L'agent migrateur appartient également à un groupe *système* qui rassemble les agents habilités à manipuler le cycle de vie des autres agents. Un autre groupe (distribué), *contrôle de migration*, rassemble tous les agents migrateurs sur chacun des sites d'exécution, et permet l'interaction des migrateurs lors d'une demande de migration.

Un scénario typique de migration d'agent se déroule comme suit :

- L'agent a envoie une requête au gestionnaire de groupe *mobilité* pour demander de le rejoindre avec un rôle d'*itinérant*. Le gestionnaire de groupe évalue la capacité de l'agent à rejoindre le groupe (possibilité de sérialisation, taille du code, ...)
- Quand cet agent désire migrer sur un autre noyau d'exécution, pour quelque raison que ce soit, il envoie un messsage demandant sa propre migration vers l'agent ayant le rôle de *migrateur* dans le groupe *mobilité*. Cette requête peut également être envoyée par un agent de niveau méta de gestion de la charge.
- L'agent migrateur sérialise le candidat, demande à l'agent responsable du noyau de suspendre l'activité du candidat, et envoie a vers son pair sur un autre site s'
- L'agent ayant le rôle *migrateur* sur le site s' désérialise l'agent, confirme sa bonne réception et demande à son propre agent noyau de reprendre l'exécution de l'agent transmis.
- Le migrateur du site original demande au noyau de détruire l'agent suspendu.

Notons que l'on voit ressurgir l'idée de réflexivité organisationnelle dans cette approche

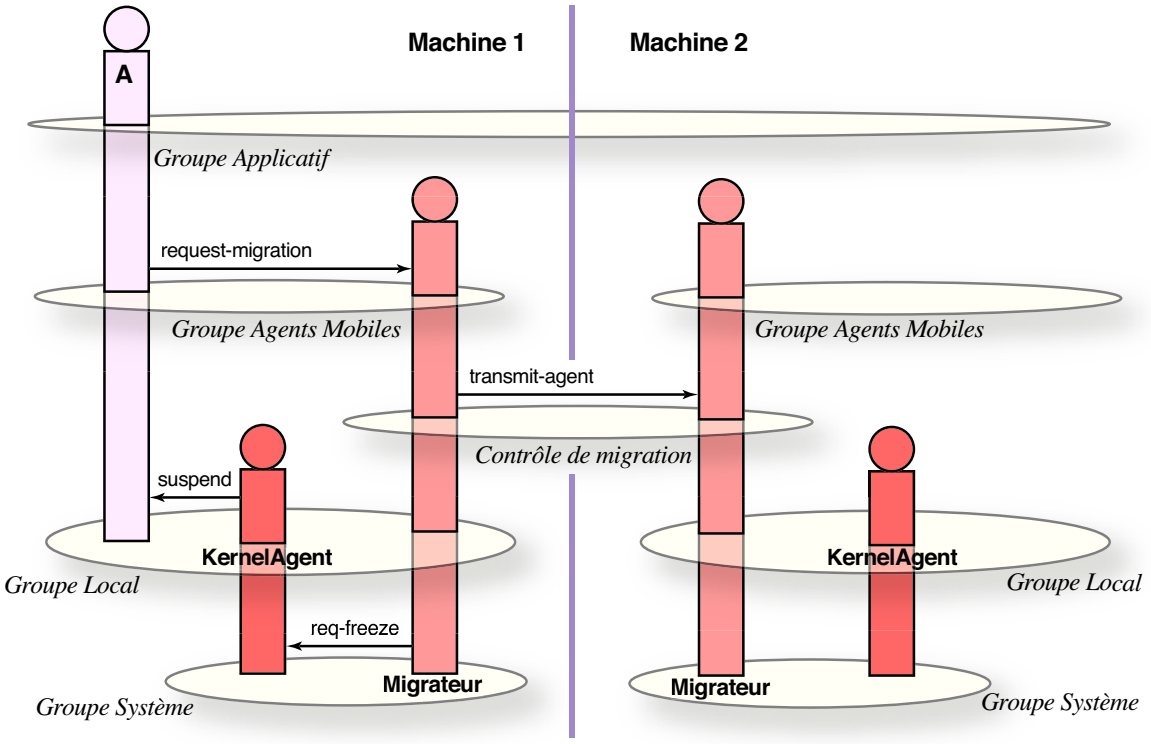


FIG. 8.10 – Première étape de la migration

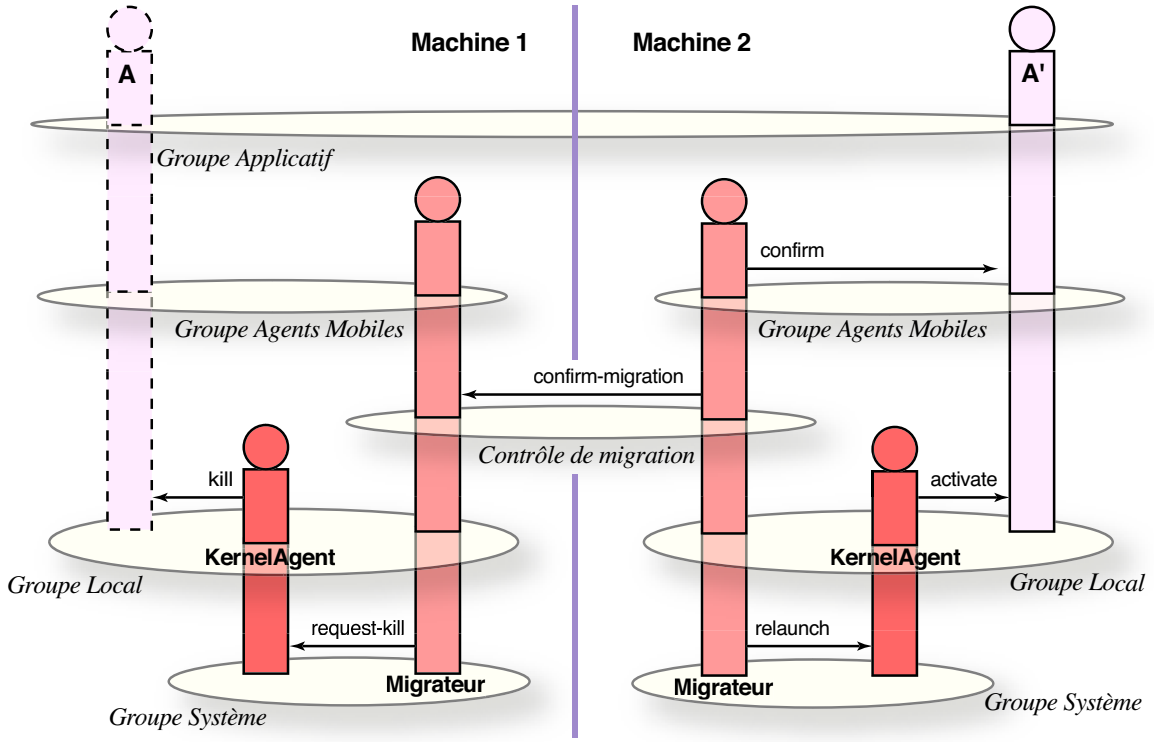


FIG. 8.11 – Deuxième étape de la migration

de deux côtés :

- L'utilisation d'un contrôle "agent" meta des opérations, pour contrôler l'arrêt ou la reprise de l'exécution des migrants.
- La construction organisationnelle du système, qui met bien en évidence la conception d'une couche de gestion de la migration exprimée en termes de groupes, rôles et capacités, et qui vient compléter les groupes bas-niveau.

Même si l'implémentation de cette architecture n'a été réalisée que pour en vérifier la faisabilité et sans la prétention d'être une véritable plate-forme d'agents mobiles, nous n'avons pu identifier de sérieuses failles dans les (petits) exemples que nous y avons exécutés. La représentation explicite de l'activité de migration au niveau meta semble même offrir des possibilités intéressantes : coexistence de plusieurs modèles de migrations simultanés, sécurité active (par échange d'information entre migrateurs, migration automatique ou sous le contrôle de l'agent).

8.2.3 Le TurtleKit

Le TurtleKit est un modèle d'agent, élaboré par M. Fabien et nous-même [Michel, 2000] pour valider l'adaptabilité du modèle à des architectures d'agents situés. L'objectif est de pouvoir émuler, à l'aide d'une définition d'agent spécialisée, le modèle d'exécution de STARLOGO [Resnick, 1994].

L'originalité du modèle STARLOGO est d'être singulièrement simple à mettre en oeuvre pour un concepteur d'agent[3] : les agents situés sont des "tortues" qui se déplacent sur un environnement bi-dimensionnel discret, et qui peuvent émettre des stimulus ou déposer et consulter des marques sur l'environnement. En pratique, un langage de programmation simple dérivé de Logo permet d'écrire les comportements en quelques lignes.

Notre but a été de proposer une implémentation de ce modèle, en en conservant les opérations élémentaires de déplacement, les mécanismes d'interactions avec l'environnement ou le cycle de vie. Néanmoins, le langage et l'environnement d'exécution n'étant pas le même, et l'écriture diffère. L'ensemble a été intégré à des outils spécialisés sur ce modèle (interfaces de contrôle et de visualisation spécifiques). L'objectif final est de fournir un outil intégré mis en place comme une surcouche à MADKIT et masquant l'utilisation sous-jacente des groupes et des rôles. Néanmoins, un concepteur de systèmes a toujours la possibilité de redescendre au niveau primitif pour réintégrer des agents "tortues" développés dans ce modèle avec tout autre agent MADKIT, par le jeu de la structuration organisationnelle.

Nous avons validé cette implémentation en y implémentant la plupart des exemples et tests de l'outil original (voir figure 8.12.

[3] Et en particulier, de pouvoir être utilisable par des enfants.

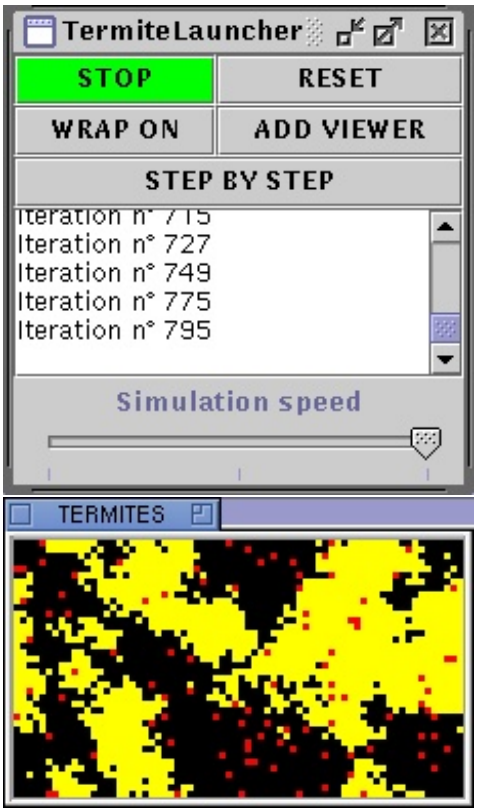


FIG. 8.12 – Un exemple de STARLOGO réimplémenté dans le TurtleKit

8.3 Applications externes

Comme nous l'avons indiqué au chapitre précédent, une partie importante à nos yeux de la validation de ce travail se fait à travers l'utilisation de notre modèle ou de la plate-forme d'implémentation dans d'autres cadres d'usages que les notres. Nous avons donc laissé l'usage de notre outil à la communauté au fur et à mesure de son évolution. Cela a conduit à près de trois mille chargements, et plus d'une centaine d'utilisateurs que l'on pourrait qualifier de "réguliers"[4]. Nous voulons donc présenter ici quelques applications notables par leur ampleur ou leur domaine d'usage inhabituel.

8.3.1 Gestion de production

Un exemple d'application dans le domaine de la gestion de production distribuée a été réalisé dans le cadre du projet OCEAN au laboratoire PRISMa par C. Bournez, où MADKIT a été utilisé comme plate-forme d'implémentation. Le modèle agent-groupe-rôle a été mis en oeuvre en tant que modèle *d'implémentation*, ce qui est assez original : les relations groupes

[4] Ayant téléchargé de nouvelles versions de l'outil, ou étant intervenus par mail ou sur les listes de diffusion

et rôles ont été utilisées pour définir une architecture multi-agent de comportement d'un agent, avec meta-comportements et apprentissage.

L'idée générale de ce projet est que face à la dynamique non négligeable de systèmes de production distribués tels que les entreprises virtuelles, un système de pilotage émergent est susceptible de mieux s'adapter aux reconfigurations des ressources physiques (production, transport, stockage...)

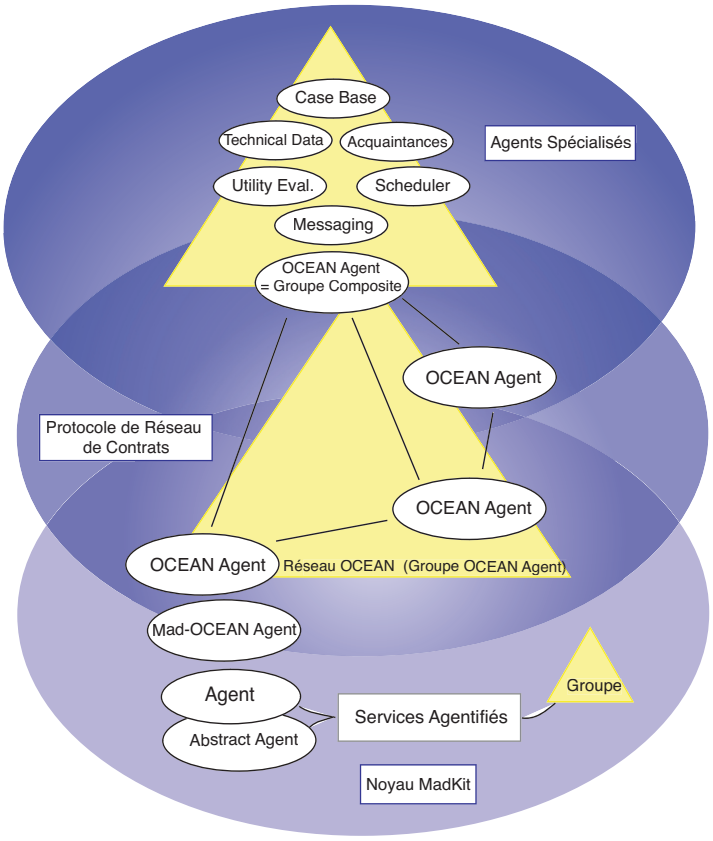


FIG. 8.13 – Architecture d'OCEAN et intégration à MADKIT (d'après [Bournez et al., 2000])

L'architecture multi-agents définie (voir figure 8.13) est réflexive, totalement hétérarchique et décentralisée, sur les bases d'un marché économique. Les mécanismes mis en jeu pour exploiter le caractère distribué du système physique de production reposent sur la coopération implicite des agents et leur coordination par un principe de marché compétitif. Leur dynamique de raisonnement est basée à la fois sur l'acquisition de comportements lors de l'exploitation du système et sur des raisonnements partiellement pré-câblés. On se réfèrera à [Bournez et al., 2000] pour plus de détail sur ce modèle.

8.3.2 Agents temps-réels

Un autre exemple d'architecture spécialisée étendant MADKIT a été réalisée à l'université du Havre [Duvallet et al., 2000], où les auteurs ont élaboré une architecture de systèmes multi-agent "any-time". Le modèle est une combinaison d'approches componentielles et prédictives. L'agent définit un modèle de tâches dont les temps d'exécution sont réévalués périodiquement. Un automate à état fini inspiré de DIMA [Guessoum, 1996], combiné à un algorithme de discrétisation du temps des différents états forme le modèle de base de l'agent. Un agent MADKIT forme la base fonctionnelle de l'agent, le comportement étant exprimé dans l'automate spécialisé (voir figure 8.14).

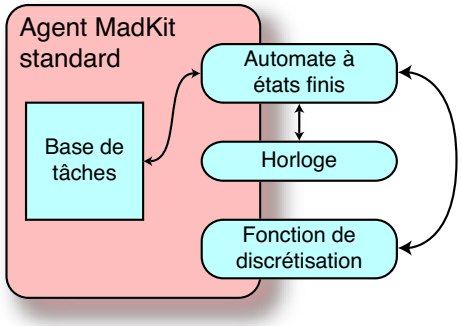


FIG. 8.14 – Modèle d'agent temps-réel étendant un agent MADKIT, d'après [Duvallet et al., 2000]

8.3.3 Représentations théatrales

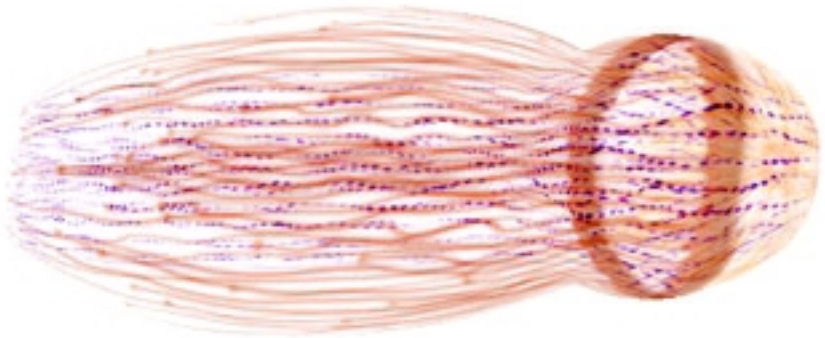


FIG. 8.15 – Un des acteurs virtuels d'Orgia

"Orgia" est la première pièce de théatre écrite par Pier Paolo Pasolini. Dans la mise en

scène de Jean Lambert-Wild, des acteurs réels et virtuels se partagent la scène. Les comportements des entités sont modelés en fonction des perceptions et des interactions avec les acteurs humains. MADKIT a été utilisé comme soubassement au système Dedalus qui assure comportement, visualisation et interaction avec le monde physique.

En voici une description plus précise, reprise de [Granit, 2000]

> *Le système Daedalus permet de créer des organismes artificiels conçus à partir d'algorithmes inspirés des organismes primitifs vivant au fond des océans. Nous les avons nommés des Posydones. Ils sont divisés en deux espèces dotées de comportements spécifiques : les Apharias et les Hyssards.*
> *Par ailleurs les comédiens sont équipés de capteurs qui enregistrent leurs niveaux de stress et d'émotion (révélés par le rythme cardiaque, l'amplitude respiratoire, la conductivité de la peau, la variation de température). Par l'intermédiaire de ces capteurs et du système informatique, les émotions générées par les comédiens exercent une influence sur le comportement des Posydones.*
> *Grâce à une illusion d'optique utilisant un miroir de la dimension de la scène, les Posydones (voir figure 8.15), perçus en trois dimensions, paraissent évoluer dans le même espace que les comédiens et réagissent en temps réel à ce qui est en train de se jouer.*

8.3.4 Jeux en ligne massivement distribués

Une autre application, plus exotique, du modèle AGR et de MADKIT a été faite par la société PEPLUM, dans le domaine des jeux vidéos en-ligne massivement distribués. Les contraintes posées par le domaine applicatifs sont intéressantes par leur variété :

- Définir une architecture extensible pouvant gérer un nombre de joueurs allant de plusieurs dizaines à un millier.
- Pouvoir accueillir plusieurs styles de jeux simultanés, avec la possibilité pour l'utilisateur de participer à plus d'une situation.
- Laisser ouverte la possibilité d'étendre l'application par le joueur (construction de mondes)
- Gérer la distribution, la synchronisation et la persistence entre portions de mondes virtuels pouvant fonctionner sur des plate-formes de différentes puissances, via des connectivités également variables.
- Pouvoir prendre en compte des jeux fonctionnant à des échelles de temps ou d'espaces distinctes (par exemple au niveau "arcade" ou au niveau stratégique).

Comme on le voit, il s'agit là de contraintes qui ne sont pas si lointaines des problématiques de la simulation distribuée.

Le choix a été fait d'une structuration basée sur le modèle agent-groupe-rôle tant pour la définition fonctionnelle des situations, que pour la gestion même de l'application. Ainsi, les problèmes de synchronisation ou d'affichage distant sont exprimés dans des structures de groupes spécifiques.

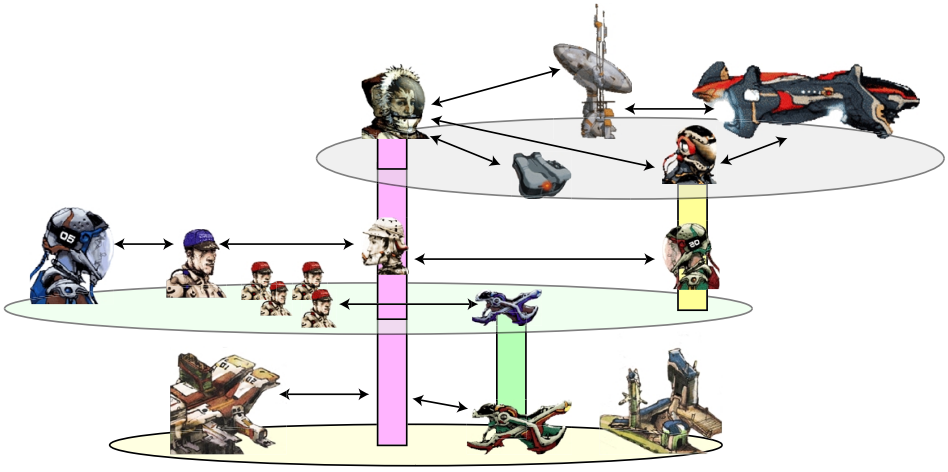


FIG. 8.16 – Une vue (ludique) d'une mise en oeuvre de notre modèle dans le prototype Peplum

Le point notable est néanmoins l'usage du modèle pour représenter l'activité d'un utilisateur au sein du jeu. L'idée générale est de pouvoir faire interragir un joueur à un instant donné en tant que "pilote", dans le même temps dans un contexte de stratégie économique, ou encore en tant que membre d'une unité d'exploration (voir la figure 8.16 pour un exemple). Les interactions entre ces joueurs (ou simples robots) sont alors définis dans des structures de groupes distinctes, correspondant aux différentes situations de jeu. Cela permet de faire évoluer aisément et rapidement l'architecture globale pour mieux répondre à l'état du monde virtuel ou aux demandes des joueurs. Le prototypage de ce système a été fait à l'aide de MADKIT.

D'un point de vue de communication de groupe, le modèle développé par PEPLUM vise à minimiser les communications. L'hypothèse étant celle d'une application massivement multi-utilisateurs, une modélisation en multi-environnement, multi-ordonnanceur (comme nous l'avons d'ailleurs vu précdemment dans un cadre de simulation plus classique) et post-syncrhonisation est utilisées pour ne transférer que les interactions de remise à jour strictement nécessaires. Le découpage du "monde" en plusieurs zones (d'activités et géographiques) s'y prête alors naturellement : tout groupe possède une partie générique (synchronisation, communication, environnement) et spécifique (activité, interactions définies selon chaque contexte).

8.4 Vers un atelier logiciel agent : SEDIT

Ce dernier exemple d'application réalisée avec MADKIT sera pour la dernière fois un retour à un point de vue réflexif. Nous allons ici évoquer SEDIT[5], une application graphique de manipulation de modèles, construite en collaboration avec J. Ferber. Ce n'est guère l'aspect d'éditeur de diagrammes générique qui nous intéresse ici, d'autant que cet aspect a été déjà (abondament et mieux) traité par le passé [Revault, 1996], mais plutôt son intégration à MADKIT.

La motivation pour la construction et l'utilisation de cet outil était d'explorer les possibilités d'avoir un atelier de génie logiciel "Agent", capable de prendre en compte les multiples modèles impliqués dans la construction d'un système multi-agent complexe (vues organisationnelles, réseaux de Petri, schémas d'interactions). L'intérêt pour ce qui nous préoccupe ici porte sur l'élaboration de cette application en tant que système multi-agents. Nous allons donc voir rapidement son architecture, et les conséquences du choix d'un "développement orienté agent".

8.4.1 Un éditeur graphique multi-formalisme

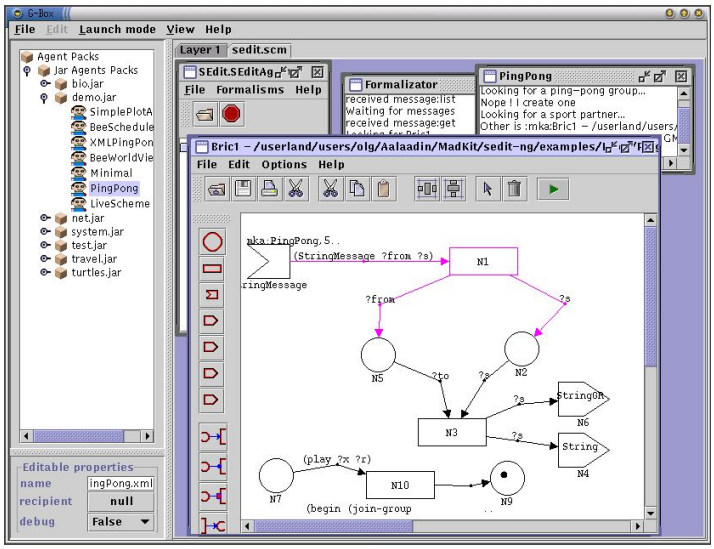


FIG. 8.17 – SEDIT avec un formalisme décrivant le comportement d'un agent

SEDIT a pour base un éditeur graphique de formalismes représentables sous formes de graphes. Il autorise également la définition de structures imbriquées (par exemple d'un paquetage à un diagramme de classes), ouvertes et liées d'un éditeur à l'autre. Les formalismes sont définis de façon externe, sous forme d'une description XML, associé à des classes

[5]Pour "Structure Editor"

supplémentaires si l'on veut contrôler finement son apparence ou son comportement (par exemple pour permettre la simulation d'un réseau de Pétri).

8.4.2 Une application multi-agent

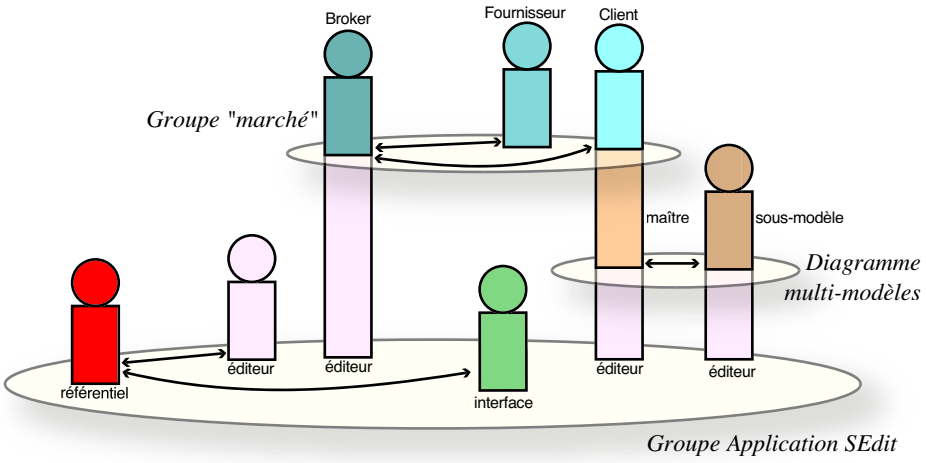


FIG. 8.18 – SEDIT comme système multi-agent

Du point de vue de MADKIT, cet outil est construit sous forme d'un ensemble d'agents. Chaque éditeur est en fait un agent MADKIT, configuré pour un formalisme donné, et de les interactions entre éditeurs (par exemple pour l'imbrication de structures) sont des interactions inter-agents classiques.

Cet outil aurait pu bien sûr être développé sous forme d'une application standard. Nous avons choisi de l'implémenter sous forme d'agents pour les raisons suivantes :

Gain de fonctionnalités via des agents externes Ainsi, l'utilisation conjointe des agents SEDIT et des agents de communication permet de faire fonctionner directement l'application en mode distribué, par exemple pour du travail collaboratif sur un modèle multi-formalisme.

Variabilité d'usage SEDIT existe en deux versions : l'une a l'apparence d'une application classique (pour faciliter l'usage de formalismes non spécifiquement agents), l'autre fonctionne à l'intérieur de la G-Box. Les agents SEDIT n'ont pas à être modifiés dans les différentes applications hôtes.

Définition d'agent "en place" Certains formalismes associés (réseaux de Pétri, BRIC, règles d'actions) permettent de définir des comportement d'agents et de les simuler. Plutôt que de générer le code des agents et les réintégrer dans MADKIT, on peut également tester les agents *en place* (voir figure 8.17), par exemple via certains noeuds du formalisme qui capturent les messages arrivant à l'éditeur et les réintégrent dans la structure

sous forme de jetons. L'agent implémentant un éditeur de modèle de comportement (par exemple par un automate) joue alors *à la fois* le rôle d'un éditeur graphique dans le "système multi-agents SEDIT" et le rôle de l'agent applicatif en train d'être élaboré (et qui peut être plongé dans ce qui sera son groupe final, sans que ses accointances s'aperçoivent qu'il s'agit d'un agent en construction).

8.4.3 Conséquences de l'intégration à la plate-forme

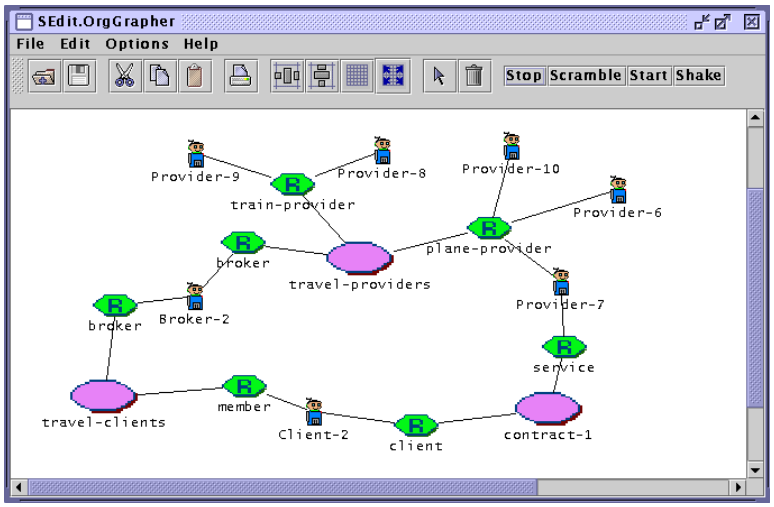


FIG. 8.19 – Agents d'analyse et représentation d'une organisation

Définir l'application comme un système multi-agent sur la plate-forme permet aussi d'observer ou modifier les opérations élémentaires. Par exemple, la figure 8.19 montre un formalisme spécialisé SEDIT qui permet, en connectant l'agent SEDIT au meta-niveau, de tracer et d'animer sous forme de graphe, les évolutions de la structure organisationnelle en temps réel.

C'est en fait une manière encore différente d'envisager le principe d'agents de surveillance que nous avons évoqué au chapitre 6. L'unification de ces différents comportement d'observation ou d'analyse reste possible via l'identification en groupe et rôle.

Quatrième partie

Perspectives et compléments

Chapitre 9

Conclusion

9.1 Contributions

Le point principal de cette contribution est bien sûr le modèle agent-groupe-rôle. Mais plus généralement, nous avons voulu défendre l'idée qu'une analyse à un niveau global et social des systèmes multi-agents était possible, et peut-être même essentielle pour pouvoir répondre à des contraintes d'hétérogénéité et d'ouverture importantes.

Cette étude faisait le jeu intellectuel de se focaliser sur des modèles purement organisationnels, en laissant un peu artificiellement de côté les modèles d'agents. Nous espérons avoir mis en évidence qu'une approche organisationnelle peut couvrir un champ plus étendu que ce qui est généralement retenu des pratiques centrées-agent habituelles. Un point significatif des modèles organisationnels développés ici est l'équilibre entre l'expressivité résultante et le choix d'un nombre réduit de concepts primitifs.

Nous avons également présenté une formalisation possible de la dynamique d'organisation en π-calcul, et examiné comment un lien peut être fait dans un même modèle formel avec d'autres architectures d'agents. Plus qu'un moyen de faire la jonction avec des modèles individuels, nous y voyons la possibilité d'exprimer contraintes et propriétés de structures purement organisationnelles, dans un contexte d'opérationalisation.

Au niveau méthodologique, nous avons présenté des possibilités d'intégration de cette perspective organisationnelle avec d'autres méthodes. La problématique des représentations et notations nous semble être un axe clé : la vitalité du champ des systèmes multi-agents se traduit par un nombre important de modèles, mais le partage de l'expérience acquise se heurte souvent à de simples problèmes de représentation et d'échange. Par ce travail, qui se veut complémentaire des efforts d'élaboration de langages de notation et de processus de conception agents, nous espérons avoir pu montrer la possibilité d'exprimer des implicites sociaux souvent masqués par les schémas d'interactions ou dans les connaissances propres des agents.

Nous retirons de la mise en oeuvre implémentatoire et applicative de ces modèles deux enseignements. D'une part, nous avons pu vérifier par les divers exemples applicatifs que nous avons construits ou qui ont été élaborés par d'autres que la gestion de l'hétérogénéité entre systèmes multi-agents ouverts et dynamiques répondait aux besoins que nous avions identifiés.

Mais d'autre part, au delà d'une implémentation et de la validation d'un modèle, l'expérience acquise ici nous a permis d'aborder la problématique des infrastructures multi-agents. Si la réalisation obtenue est bien sûr bien loin d'une plate-forme multi-agents générique et complète, nous avons montré que définir le méta-modèle d'un tel outil à un niveau d'abstraction 'agent' nous semble ouvrir des perspectives et établir des parallèles intéressants avec ce qui a pu être réalisé de comparable dans le domaine de l'objet.

9.2 Extrapolations

Que pourrions nous envisager comme prolongements à ce travail ? La première piste serait de continuer à étudier le modèle agent-groupe-rôle. Nous avons envisagé dans ces pages un usage 'direct' du modèle de construction de systèmes multi-agents, il serait intéressant d'étudier dans des contextes plus précis l'apport d'une modélisation organisationnelle. On pourrait citer la tolérance aux pannes (par le passage de rôle), la maintenance à long terme d'un système (via une division en groupes structurant les agents d'interactions compatibles), l'usage de méta-groupes et lois sociales pour structurer des organisations de grande taille.

En sortant ensuite du strict modèle organisationnel, il pourrait s'agir, d'une part de raffiner la définition de modèles spécifiques et l'articulation entre les concepts centraux et ceux rajoutés, et d'autre part de resituer ce travail dans un cadre plus large d'analyse et de résolution de problème par approches agents. Il faudrait ainsi étudier les dépendances croisées qui peuvent naître d'un travail de conception conjointe de modèles internes d'agents et de leur modèle organisationnel. Cela nous semble aussi intéressant à mener dans la définition de nouveaux modèles concrets que dans le processus de conception en lui-même qui devra prendre en compte ces interrelations entre point de vue social et individuel.

Même si la question des relations à l'environnement est apparue en filigrane lors de notre étude des modèles abstraits, il serait intéressant de pousser beaucoup plus loin cette réflexion, en étudiant les liens possibles entre modèles de groupes et modèles d'environnements physiques. De même, l'étude au niveau meta des actions et perceptions des agents par rapport à un modèle d'environnement est une perspective intéressante, qu'il faudrait explorer plus avant que le simple prototype que nous avons exposé.

La formalisation proposée ici demanderait à dépasser ce stade d'ébauche et à être examinée en parallèle avec la construction d'un système. Cela permettrait d'évaluer des propriétés spécifiques d'un modèle du point de vue de l'organisation. Parmi les conséquences les plus

immédiates, il s'agirait de pouvoir prouver la possibilité ou non de transition d'une information d'un point à un autre dans une organisation multi-agents complexe, par réécritures successives.

Une étude détaillée des apports du modèle et de sa vision méthodologique reste à faire. Cette étude devra entre autres avoir une approche comparative par rapports aux techniques de conceptions classiques et aux méthodologies et analyses typiquement agent ou multi-agents. L'exploration de "design patterns" d'organisations, ou les expérimentations d'ateliers de développement agent unifiés mériteraient de dépasser leur cadre initial et de fusionner avec d'autres propositions, pour avoir une démarche et une chaîne de conception validées de bout en bout.

Toujours dans les aspects implémentatoires, la poursuite du travail sur MADKIT et SEDIT nous parait fondamentale : il s'agit de continuer à évaluer cette plate-forme sur des applications plus importantes et, au fur et à mesure des expérimentations, tenter de déterminer si notre revendication de réutilisabilité accrue et de caractérisation de structures organisationnelles récurrente se vérifie bien. La conception à un méta-niveau agent de la plate-forme et de l'outil de conception nous a montré l'importance de gérer la multiplicité de point de vue dans un futur hypothétique atelier de conception agent.

9.3 Anticipations

Un thème que nous avons à peine évoqué transparaissait dans cette étude : celui de l'intelligence propre de ces artefacts informatiques. Que ce soit apprentissage symbolique, émergence d'une intelligence collective, nous avons plus travaillé ici sur ce qui pourrait être le contexte de ces comportements autonomes et intelligents que sur leur nature. Ce pourrait être notre souhait par rapport à ce travail : baliser quelques places pour mieux pouvoir porter l'effort sur le coeur du problème, qui ne sera pas réalisé avant longtemps mais reste l'horizon à atteindre [Nilsson, 1995].

Nous voudrions bien sûr que les thèmes des modèles centrés sur le collectif dans les systèmes à agents continuent à être abordés. S'il est certain que la prise en compte des problématiques "typiquement" agent comme l'interaction, l'émergence ou l'adaptabilité à grande échelle est essentielle, est-il finalement aussi certain que ce soit les modèles multi-agents eux-mêmes qui y répondent ? Les systèmes distribués, les services Web, les infrastructures objet sont autant de domaines - plus éprouvés peut-être - déjà amenés à explorer les mêmes problématiques. Il s'agit de modèles peut-être un peu moins adaptés, mais qui ont pour eux une pédagogie plus maîtrisée, des expériences de référence, et des acquis plus faciles à capitaliser. Ce sur quoi nous voulons donc mettre l'accent pour finir, c'est le besoin d'un travail, avant tout épistémologique, de retour sur plus d'une décennie d'expérimentations,

de modèles et de théories. Le champ qui nous a intéressé ici s'est toujours caractérisé par sa capacité à jeter des ponts avec d'autres domaines d'études, il n'en est que plus important d'en mettre en évidence les singularités.

D. Hillis[Hillis, 1986] concluait son travail en imaginant le recours à la physique pour mieux comprendre les futures architectures informatiques. De même, les modèles sociaux et les sciences du vivant peuvent offrir des analogies cruciales pour appréhender les architectures logicielles de demain. Les problèmes ne sont plus vraiment dans le champ des systèmes logiciels en tant que tels : la plupart du temps, *nous savons les construire*.

L'enjeu s'est déplacé vers l'espace interconnectant ces systèmes : le logiciel de l'interstice reste pour beaucoup à inventer.

Bibliographie

[Adam et al., 1999] Adam, E., Mandiau, R., et Kolski, C. (1999). Approche holonique de modélisation d'une organisation orientée workflow : SOHTCO. In Gleizes, M.-P. et Marcenac, P., editors, *Actes des 7èmes journées francophones sur l'Intelligence Artificielle distribuée et les systèmes multi-agents (JFIADSMA'99)*, pages 121–134. Hermès.

[Agha et Hewitt, 1987] Agha, G. et Hewitt, C. (1987). Concurrent programming using actors. In *Object-Oriented Concurrent Programming*, pages 37–53. MIT Press, Cambridge, MA.

[Agha, 1986] Agha, G. A. (1986). *Actors : a Model of Concurrent Computation in Distributed Systems*. MIT Press.

[Arnold et al., 1999] Arnold, K., Wollrath, A., O'Sullivan, B., Scheifler, R., et Waldo, J. (1999). *The Jini specification*. Addison-Wesley, Reading, MA, USA.

[Baeijs et Demazeau, 1996] Baeijs, C. et Demazeau, Y. (1996). Les organisations dans les systèmes multi-agents. In *4emes journees du GDR PRC IA*.

[Banâtre et Le Métayer, 1990] Banâtre, J.-P. et Le Métayer, D. (1990). The GAMMA model and its discipline of programming. *Science of Programming*, 15(1) :55–77.

[Barendregt, 1984] Barendregt, H. P. (1984). *Studies in Logic and the Foundations of Mathematics*, volume 2, chapter The Lambda Calculus : Its Syntax and Semantics. North Holland, 2 edition.

[Baumann et al., 1997] Baumann, J., Hohl, F., Radouniklis, N., Rothermel, K., et Strasser, M. (1997). Communication concepts for mobile agent systems. In *First International Workshop on Mobile Agents 97*, Berlin.

[Baumann et Radouniklis, 1997] Baumann, J. et Radouniklis, N. (1997). Agent groups in mobile agent systems. In *IFIP WG 6.1, International Conference on Distributed Applications and Interoperable Systems (DAIS 97)*.

[Baumgärtel et al., 1996] Baumgärtel, H., Bussmann, S., et Klosterberg, M. (1996). Combining multi-agent systems and constraint techniques in production logistics. In *Proceedings of the International Conference on AI, Simulation and Planning in High Autonomy Systems*, pages 361–367.

[Beck, 1999] Beck, K. (1999). *Extreme Programming Explained : Embracing Change*. Addison-Wesley.

[Berners-Lee et Fischetti, 1999] Berners-Lee, T. et Fischetti, M. (1999). *Weaving the Web : The Original Design and Ultimate Destiny of the World Wide Web by its Inventor*. Harper.

[Berners-Lee et al., 2001] Berners-Lee, T., Hendler, J., et Lassila, O. (2001). The semantic web. *Scientific American*.

[Berry et Boudol, 1992] Berry, G. et Boudol, G. (1992). The chemical abstract machine. *Theoretical Computer Science*, 96(1) :217–248.

[Boudol, 1992] Boudol, G. (1992). Asynchrony and the Pi-calculus. Technical Report RR-1702, Inria.

[Boudol, 1993] Boudol, G. (1993). Some Chemical Abstract Machines. In deBakker, J., de-Roever, W., et Rozenberg, G., editors, *Lecture Notes in Computer Science*, volume 803, pages 92–123. Springer-Verlag, Berlin.

[Bournez et al., 2000] Bournez, C., Beslon, G., et Favrel, J. (2000). A multi-agent system for distributed factory decentralized emergent control. In *Proceedings of the 7th ISPE International Conference on Concurrent Engineering (CE'2000)*, pages 445–452.

[Bouron, 1992] Bouron, T. (1992). *Structure de communication et d'organisation pour la coopération dans un univers-multi-agents*. PhD thesis, Université Paris 6.

[Bousquet et al., 1998] Bousquet, F., Bakam, I., Proton, H., et Le Page, C. (1998). Cormas : Common-pool resources and multi-agents systems. *Lecture Notes in Computer Science*, 1416 :826–837.

[Bouzouba et Moulin, 1997] Bouzouba, K. et Moulin, B. (1997). La négociation des relations sociales dans les conversations multi-agents. In Quinqueton, J., Thomas, M.-C., et Trousse, B., editors, *Actes des 5èmes journées francophones JFIADSMA'97*, pages 63–75. Hermès.

[Bradner, 1996] Bradner, S. (1996). The internet standards process. Technical Report RFC 2026, IETF.

[Bray et al., 1998] Bray, T., Paoli, J., et Sperberg-McQueen, C. M. (1998). Extensible markup language (xml) 1.0. W3C Recommendation. http ://www.w3.org/TR/1998/REC-xml-19980210.

[Brazier et al., 1997] Brazier, F., Dunin, B., Jennings, N., et Treur, J. (1997). Desire : modelling multi-agent systems in a compositional formal framework. *International Journal of Cooperative Information Systems*. Special issue on Formal Methods in Cooperative Information Systems.

[Breton, 1990] Breton, P. (1990). *La tribu informatique*. Métailié.

[Bäumer et al., 1996] Bäumer, D., Knoll, R., Gryczan, G., et Züllighoven, H. (1996). Large scale object-oriented software-development in a banking environment - an experience

report. In Cointe, P., editor, *ECOOP 1996 - Object Oriented Programming, 10th European conference*, pages 73–90. Springer.

[Burkhart, 1997] Burkhart, R. (1997). Schedules of activity in the swarm simulation system. In *OOPSLA'97 Workshop on OO Behavioral Semantics*.

[Burmeister, 1996] Burmeister, B. (1996). Models and methodology for agent-oriented analysis and design. In *Proceedings of the KI'96 workshop on agent-oriented programming and distributed systems*. DFKI.

[Burmeister et Sundermeyer, 1990] Burmeister, B. et Sundermeyer, K. (1990). COSY : Towards a methodology of multi-agent systems. In *In International Working Conference of Cooperating Knowledge-Based Systems*, Keele, UK.

[Calderoni et al., 1997] Calderoni, S., Courdier, R., Leman, S., et Marcenac, P. (1997). Construction expérimentale d'un modèle multi-agents. In Quinqueton, J., Thomas, M.-C., et Trousse, B., editors, *Actes des 5èmes journées francophones JFIADSMA'97*, pages 109–123. Hermès.

[Cammarata et al., 1983] Cammarata, S., McArthur, D., et Steeb, R. (1983). Strategies of cooperation in distributed problem solving. In *International Joint Conference on Artificial Intelligence*, pages 767–770.

[Cardon, 1999] Cardon, A. (1999). *Conscience artificielle et systèmes adaptatifs*. Eyrolles.

[Carle et al., 1994] Carle, P., Collinot, A., et Zeghal, K. (1994). Concevoir des organisations : la méthode Cassiopée. In *Actes de la journée sur les systèmes multi-agents - PRC-GDR Intelligence Artificielle*.

[Castelfranchi, 1992] Castelfranchi, C. (1992). Dependence relations among autonomous agents. In Werner, E. et Demazeau, Y., editors, *Decentralized Artificial Intelligence*, pages 49–62, Amsterdam. Elsevier.

[Castelfranchi, 1995] Castelfranchi, C. (1995). Commitments : from individual intentions to groups and organisations. In Lesser, V., editor, *Proceedings of ICMAS'95, first interntaional conference on multi-agent systems*. AAAI Press - MIT Press.

[Castelfranchi et al., 1990] Castelfranchi, C., Micelli, M., et Cesta, A. (1990). Social power : a missed point in multi-agent, DAI and HCI. In Demazeau, Y. et Muller, J., editors, *Decentralized Artificial Intelligence*, pages 49–62, Amsterdam. Elsevier.

[Cerf et Kahn, 1974] Cerf, V. G. et Kahn, R. E. (1974). A protocol for packet network interconnection. *IEEE Transactions on Communications Technology*, 22(5) :627–641.

[Chavez et al., 1997] Chavez, A., Moukas, A., et Maes, P. (1997). Challenger : A multi-agent system for distributed resource allocation. In Johnson, W. L. et Hayes-Roth, B., editors, *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*, pages 323–331, New York. ACM Press.

[Coen, 1994] Coen, M. (1994). Sodabot : a software agent environment and construction system. Technical report, MIT AI Lab.

[Collinot et Drogoul, 1998] Collinot, A. et Drogoul, A. (1998). Using the cassiopeia method to design a soccer robot team. *Applied Articial Intelligence (AAI) Journal*, 12(2-3) :127–147.

[Collins, 1981] Collins, R. (1981). On the microfoundation of macrosociology. *American Journal of Sociology*, 86(5) :984–1014.

[Coulondre, 2000] Coulondre, S. (2000). *Samovar : un modèle pour les objets persistants avec rôles*. PhD thesis, Université Montpellier II.

[Crozier et Freidberg, 1981] Crozier, M. et Freidberg, E. (1981). *L'acteur et le système. Les contraintes de l'action collective*. Le Seuil.

[Degenne et Forsé, 1994] Degenne, A. et Forsé, M. (1994). *Les réseaux sociaux : une analyse structurale en sociologie*. Armand Colin.

[Demazeau, 1997] Demazeau, Y. (1997). Steps towards multi-agent oriented programming. In *First international workshop on multi-agent systems IWMAS'97*.

[Deutsch, 1989] Deutsch, P. L. (1989). Design reuse and frameworks in the Smalltalk-80 system. In Biggerstaff, T. J. et Perlis, A. J., editors, *Software reusability, Volume II : Applications and experiences*, pages 57–71. Addison-Wesley.

[Di Caro et Dorigo, 1998] Di Caro, G. et Dorigo, M. (1998). Ant colonies for adaptive routing in packet-switched communications networks. *Lecture Notes in Computer Science*, 1498 :673.

[Dignum et al., 2000] Dignum, F., Morley, D., Sonenberg, E., et Cavedon, L. (2000). Towards socially sophisticated BDI agents. In *Fourth International Conference on Multi-Agent Systems (ICMAS'00) Proceedings*. IEEE.

[Drogoul, 2000] Drogoul, A. (2000). Systèmes multi-agents situés. Mémoire d'habilitation à diriger des recherches. Université Paris 6.

[Drogoul et Collinot, 1997] Drogoul, A. et Collinot, A. (1997). Entre réductionnisme méthodologique et stratégie intentionnelle, l'éthologie, un modèle alternatif pour l'I.A.D. ? In Quinqueton, J., Thomas, M.-C., et Trousse, B., editors, *Actes des 5èmes journées francophones JFIADSMA'97*, pages 307–322. Hermès.

[Drogoul et Collinot, 1998] Drogoul, A. et Collinot, A. (1998). Applying an Agent-Oriented Methodology to the Design of Artifical Organizations : A Case Study in Robotic Soccer. *Journal of Autonomous Agents and Multi-Agent Systems*, 1(1) :113–129.

[Durand, 1996] Durand, B. (1996). *Simulation Multi-Agents et epidemiologie operationnelle : etude d'epizooties de fievre aphteuse*. PhD thesis, Universite de Caen.

[Durfee et Lesser, 1991] Durfee, E. H. et Lesser, V. R. (1991). Partial global planning : a coordination framework for distributed problem hypothesis formation. *IEEE Transactions on Man, Systems and Cybernetics*.

[Durkheim, 1895] Durkheim, E. (1895). *Les Règles de la méthode sociologique*. Flammarion.

[Duvallet et al., 2000] Duvallet, C., Sadeg, B., et Cardon, A. (2000). An anytime multi-agents systems to manage electronic commerce transactions. In *Proceedings of the 6th conference on Object-Oriented Information Systems (OOIS'2000)*, pages 121–128, Londres. Springer.

[Ellis et Lasser, 2000] Ellis, K. et Lasser, J. (2000). Falling apart at the seams. *SecurityFocus*, commentary(80).

[Epstein et Axtell, 1996] Epstein, J. et Axtell, R. (1996). *Growing artificial societies : social science from the bottom up*. MIT Press.

[Ferber, 1989] Ferber, J. (1989). *Objets et agents : une étude des structures de communication en Intelligence Artificielle*. Thèse de doctorat d'état, Université Pierre et Marie Curie.

[Ferber, 1995] Ferber, J. (1995). *Les Systemes Multi-Agents : vers l'Intelligence Collective*. Inter-Editions.

[Ferber et Carle, 1991] Ferber, J. et Carle, P. (1991). Actors and agents as reflective concurrent objects : A Mering IV perspective. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(6) :1420–1436.

[Ferber et al., 2000] Ferber, J., Gutknecht, O., Jonker, C., Muller, J.-P., et Treur, J. (2000). Organization models and behavioural requirements specification for multi-agent systems. In *Proceedings of the ECAI 2000 Workshop on Modelling artificial societies and hybrid organizations*.

[Ferber et Muller, 1996] Ferber, J. et Muller, J.-P. (1996). Influences and reaction : A model of situated multiagent systems. In Kyoto, M. T. E., editor, *Proceedings of Second International Conference on Multi-Agent Systems (ICMAS'96)*. AAAI Press.

[Ferrand, 1996] Ferrand, N. (1996). De l'apport potentiel de la sociologie pour l'ingéniérie des systèmes sociaux artificiels. In *6èmes Journées de Rochebrune : Rencontres interdisciplinaires sur les systèmes complexes naturels et artificiels*.

[Fianyo et al., 98] Fianyo, E., Treuil, J.-P., Perrier, E., et Demazeau, Y. (98). Multi-agent architecture integrating heterogeneous models of dynamical processes : the representation of time. In *Multi-agent Based Simulation '98*.

[Finin et Fritzson, 1994] Finin, T. et Fritzson, R. (1994). KQML — A language and protocol for knowledge and information exchange. In *Proceedings of the 13th Intl. Distributed Artificial Intelligence Workshop*, pages 127–136, Seattle, WA, USA.

[FIPA, 1997a] FIPA (1997a). Agent Communication Language specification. Specification 2.

[FIPA, 1997b] FIPA (1997b). Agent Management Specification. 97.AM.1.

[FIPA, 1997] FIPA (1997). Application design test : Personal travel agent. 97.PTA.4.

[Foisel, 1998] Foisel, R. (1998). *Modèle de réorganisation de systèmes multi-agents : une approche descriptive et opérationnelle*. PhD thesis, Université Henri Poincaré - Nancy 1.

[Fowler, 1996] Fowler, M. (1996). *Analysis Patterns*. Addison-Wesley.

[Fox, 1981] Fox, M. (1981). An organizational view of distributed systems. *IEEE Transactions on Man, Systems and Cybernetics*, 11(1) :70–80.

[Franklin et Graesser, 1997] Franklin, S. et Graesser, A. (1997). Is it an agent, or just a program ? In Muller, J.-P., Wooldridge, M., et Jennings, N., editors, *Intelligent Agents III*, number 1193 in Lecture Notes in Artificial Intelligenc, pages 21–36. Springer-Verlag.

[Friedman-Hill, 1998] Friedman-Hill, E. J. (1998). *Jess, The Java Expert System Shell*. Sandia National Laboratories, Livermore, CA. Version 4.0.

[Gamma et al., 1995] Gamma, E., Helm, R., Johnson, R., et Vlissides, J. (1995). *Design Patterns : Elements of Reusable Object-Oriented Software*. Addison-Wesley.

[Gasser, 1992] Gasser, L. (1992). Boundaries, aggregation and identity : plurality issues for multi-agent systems. In Werner, E. et Demazeau, Y., editors, *Decentralized Artificial Intelligence*, pages 49–62, Amsterdam. Elsevier.

[Gasser, 2000] Gasser, L. (2000). Multi-agent systems infrastructure definitions, needs, and prospects. In Wagner, T. et Rana, O., editors, *Proceedings of the first Workshop on Infrastructure for Scalable Multi-Agent Systems*, number A paraitre in Lecture Notes in Computer Science. Springer.

[Gasser et al., 1987] Gasser, L., Braganza, C., et Herman, N. (1987). MACE : a flexible testbed for distributed AI research. In Huhns, M. N., editor, *Distributed Artificial Intelligence*, pages 119–152. Pitman Publishers.

[Gasser et Briot, 1998] Gasser, L. et Briot, J.-P. P. (1998). Actors & agents : Agents and concurrent objects : Interview with Les Gasser. *IEEE Concurrency*, 6(4) :74–77, 81.

[Giddens, 1984] Giddens, A. (1984). *The constitution of society : outline of the theory of structuration*. University of California Press, Berkeley.

[Ginot et Le Page, 1998] Ginot, V. et Le Page, C. (1998). Mobidyc, a generic multi-agents simulator for modeling populations dynamics. *Lecture Notes in Computer Science*, 1416 :805–814.

[Giroux, 1993] Giroux, S. (1993). *Agents et systèmes, une nécessaire unité*. PhD thesis, Université de Montréal, Departement d'Informatique et de Recherche Opérationnelle.

[Glance et Huberman, 1995] Glance, N. S. et Huberman, B. A. (1995). Organizational fluidity and sustainable cooperation. *Lecture Notes in Computer Science*, 957 :89.

[Glaser, 1997] Glaser, N. (1997). The CoMoMAS methodology and environment for multi-agent system development. *Lecture Notes in Computer Science*, 1286 :1.

[Gottlob et al., 1996] Gottlob, G., Schrefl, M., et Rock, B. (1996). Extending object-oriented systems with roles. *ACM transactions on Information Systems*, 14(3) :268–296.

[Gouldner, 1954] Gouldner, A. W. (1954). *Patterns of industrial bureaucracy*. Free Press.

[Granit, 2000] Granit, T. (2000). Orgia, dossier descriptif. Dossier de presse, également sous http ://www.theatre-granit.asso.fr/orgia/cadreorgia.html.

[Gray et al., 1998] Gray, D. N., Hotchkiss, J., LaForge, S., Shalit, A., et Weinberg, T. (1998). Modern languages and Microsoft's component object model. *CACM*, 41(5) :55–65.

[Guessoum, 1996] Guessoum, Z. (1996). *Un environnement opérationnel de conception et de réalisation de systèmes multi-agents*. PhD thesis, Université Paris 6.

[Gutknecht et al., 1998] Gutknecht, O., Ferber, J., et Lieurain, E. (1998). Des modèles hétérogènes de simulation par systèmes multi-agents. In Ferrand, N., editor, *Actes du colloque Modèles et Systèmes multi-agents pour la gestion de l'environnement et du territoire (SMAGET) 1998*. Cemagref.

[Gutknecht et al., 2001] Gutknecht, O., Michel, F., et Ferber, J. (2001). Integrating tools and infrastructures for generic multi-agent systems. In *Proceedings of Autonomous Agents 2001 conference*.

[Hannouns et al., 1999] Hannouns, M., Boissier, O., Sichman, J. S., et Sayettat, C. (1999). Moise : un modèle organisationnel pour la conception de systèmes multi-agents. In Gleizes, M.-P. et Marcenac, P., editors, *Actes des 7èmes journées francophones sur l'Intelligence Artificielle distribuée et les systèmes multi-agents (JFIADSMA'99)*, pages 105–118. Hermès.

[Haynes et al., 1995] Haynes, T. D., Wainwright, R. L., et Sen, S. (1995). Evolving cooperating strategies. In Lesser, V., editor, *Proceedings of the first International Conference on Multiple Agent Systems*, page 450, San Francisco, USA. AAAI Press/MIT Press. Poster.

[Heitsch et al., 1999] Heitsch, S., Martens, M., et Moldt, D. (1999). Petri nets with synchronous channels applied to a sociological example. In *Proceedings of the High-level Petri-net applications workshop*.

[Hewitt et Inman, 1991] Hewitt, C. et Inman, J. (1991). DAI betwixt and between : From 'intelligent agents' to open systems science. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(6) :1409–1419.

[Hilaire et al., 2000] Hilaire, V., Koukam, A., Gruer, P., et Müller, J.-P. (2000). Formal specification and prototyping of multi-agent systems. In Ominici, A., Tolksdorf, R., et Zambonelli, F., editors, *First international workshop on Engineering societies in the agents world*, number 1972 in Lecture notes in artificial intelligence, pages 114–127. Springer.

[Hillis, 1986] Hillis, W. D. (1986). *The Connection Machine*. The MIT Press.

[Hubert Proton, 1997] Hubert Proton, François Bousquet, P. R. (1997). Un outil pour observer l'organisation d'une société d'agents. le cas d'une société d'agents chasseurs agriculteurs. In *Actes des 5emes journees JFIADSMA*. Hermes.

[Iglesias et al., 1998a] Iglesias, C. A., Garijo, M., Gonzalez, J. C., et Velasco, J. R. (1998a). Analysis and design of multiagent systems using MAS-CommonKADS. *Lecture Notes in Computer Science*, 1365 :313.

[Iglesias et al., 1998b] Iglesias, C. A., Garijo, M., et González, J. C. (1998b). A survey of agent-oriented methodologies. In *Proceedings of the Workshop on Agent Theories, Architectures and Languages (ATAL'98*, volume INTELLIGENT AGENTS V of *LNCS*.

[Jacobson et Nowack, 1999] Jacobson, E. E. et Nowack, P. (1999). Frameworks and patterns : architectural abstractions. In Fayad, M. E., Schmidt, D. C., et Johnson, R. E., editors, *Building application frameworks : object-oriented foundations of framework design*, pages 29–54. Wiley.

[Jennings, 2000] Jennings, N. R. (2000). On agent-based software engineering. *Artificial Intelligence Journal*, 177 :277–296.

[Jung et Fischer, 1998] Jung, C. et Fischer, K. (1998). Methodological comparison of agent models. Research Report RR-98-01, Deutsches Forschungszentrum für Künstliche Intelligenz.

[Karjoth, 2000] Karjoth, G. (2000). Secure mobile agent-based merchant brokering in distributed marketplaces. In *Proceedings of the Second International Symposium on Agent Systems and Applications and Fourth International Symposium on Mobile Agents (ASA/MA2000)*, volume 1882 of *Lecture Notes in Computer Science*, pages 44–56, Zurich, Switzerland. Springer-Verlag.

[Kendall, 1999] Kendall, E. A. (1999). Role model designs and implementations with aspect-oriented programming. *ACM SIGPLAN Notices*, 34(10) :353–369.

[Kendall et al., 1995] Kendall, E. A., Malkoun, M. T., et Jiang, C. H. (1995). A methodology for developing agent based systems. In Zhang, C. et Lukose, D., editors, *First Australian Workshop on Distributed Artificial Intelligence*, Canberra, Australia.

[Kinny et Georgeff, 1996] Kinny, D. et Georgeff, M. (1996). Modelling and design of multi-agents systems. In *Proceedings of the Workshop on Agent Theories, Architectures and Languages (ATAL'96*, number 1193 in Lecture notes in artificial intelligence. Springer-Verlag.

[Kinny et al., 1994] Kinny, D., Ljungberg, M., Rao, A., Sonenberg, E., Tidhar, G., et Werner, E. (1994). Planned team activity. In Castefranchi, C. et Werner, E., editors, *Proceedings of the 4th European Workshop on Modelling Autonomous Agents in a Multi-Agent World : Artificial Social Systems*, volume 830 of *LNAI*, pages 227–258, Berlin. Springer.

[Kitano et al., 1998] Kitano, H., Asada, M., Kuniyoshi, Y., et Noda, I. (1998). RoboCup : A challenge problem for AI and robotics. *Lecture Notes in Computer Science*, 1395.

[Knorr-Cetina et Cicourel, 1981] Knorr-Cetina, K. et Cicourel, A. V., editors (1981). *Advances in social theory and methodology : toward an integration of micro and macro-sociologies*, Boston. Routledge and Kegan Paul.

[Labrou et Finin, 1997] Labrou, Y. et Finin, T. (1997). Comments on the specification for FIPA '97 agent communication language. UMBC WWW.

[Lafaye, 1996] Lafaye, C. (1996). *La sociologie des organisations*. Nathan.

[Lashkari et al., 1994] Lashkari, Y., Metral, M., et Maes, P. (1994). Collaborative interface agents. In *Proceedings of AAAI'94*.

[Lechilli et Chaib-draa, 1996] Lechilli, K. et Chaib-draa, B. (1996). Structures relationnelles d'activités entre agents. In Muller, J.-P. et Quinqueton, J., editors, *Actes des 4èmes journées francophones sur l'Intelligence Artificielle distribuée et les systèmes multi-agents (JFIAD-SMA'96)*. Hermès.

[Magnin, 1996] Magnin, L. (1996). *Modélisation et simulation de l'environnement dans les systèmes multi-agents. Application aux robots footballeurs*. PhD thesis, Université Paris 6.

[Marcenac et Courdier, 1999] Marcenac, P. et Courdier, R. (1999). A java agent-oriented development environment. In *Object-Oriented Application Frameworks*. Wiley and Sons Books.

[March et Simon, 1958] March, J. G. et Simon, H. A. (1958). *Organizations*. Wiley.

[Maruichi, 1989] Maruichi, T. (1989). *Organizational Computation : A Framework for Distributed Cooperative Problem-Solving using Autonomous Agents and Their Groups*. PhD thesis, Department of Electrical Engineering, Keio University, Yokohama, Japan.

[Maruichi et al., 1990] Maruichi, T., Ichikawa, M., et Tokoro, M. (1990). Modeling Autonomous Agents and Their Groups. In Demazeau, Y. et Muller, J., editors, *Decentralized AI, Proc. 1st European Workshop on Modelling Autonomous Agents in a MultiAgent World*, pages 215–234, Cambridge, UK. North-Holland.

[Matsuoka et al., 1991] Matsuoka, S., Watanabe, T., et Yonezawa, A. (1991). Hybrid group reflective architecture for object-oriented concurrent reflective programming. In America, P., editor, *Proceedings ECOOP'91*, LNCS 512, pages 231–250, Geneva, Switzerland. Springer-Verlag.

[Mayo, 1933] Mayo, E. (1933). *The human problems of an industrial civilization*. MacMillan.

[Merlat, 1999] Merlat, W. (1999). *Adaptation dynamique de l'organisation dans les Systèmes Multi-Agents. Application à la conception des Systèmes Coopératifs Distribués et Ouverts*. PhD thesis, Université Paris 6.

[Meyer, 1989] Meyer, B. (1989). *Object-oriented software construction*. Prentice Hall, New York.

[Michel, 2000] Michel, F. (2000). Une approche méthodologique pour la conception et l'analyse de simulateur multi-agents. In *5èmes Rencontres Nationales des Jeunes Chercheurs en Intelligence Artificielle*, pages 269–279, Lyon.

[Milner, 1984] Milner, R. (1984). *Lectures on a Calculus for Communicating Systems*, volume 197 of *LNCS*. Springer-Verlag, New York, NY.

[Milner, 1989] Milner, R. (1989). *Communication and concurrency*. Prentice-Hall.

[Milner et al., 1992] Milner, R., Parrow, J., et Walker, D. (1992). A calculus of mobile processes, I, II. *Information and Computation*, 100(1) :1–77.

[Minsky, 1985] Minsky, M. (1985). *The Society of Mind*. Simon and Schuster, New York.

[Monson-Haefel, 1999] Monson-Haefel, R. (1999). *Enterprise JavaBeans*. O'Reilly & Associates, Inc., 981 Chestnut Street, Newton, MA 02164, USA.

[Morin, 1977] Morin, E. (1977). *La méthode : la nature de la Nature*. Le Seuil.

[Moulin et Brassard, 1996] Moulin, B. et Brassard, M. (1996). A scenario-based design method and environment for the development of multi-agent systems. In *Proceedings of the first australian workshop on distributed artificial intelligence*, number 1087 in Lecture notes in artificial intelligence, pages 216–231. Springer.

[Muller, 1998] Muller, J.-P. (1998). Vers une méthodologie de conception de systèmes multi-agents de résolution de problèmes par émergence. In Barthes, J.-P., editor, *Actes des Journées Francophones en Intelligence Artificielle Distribuée et Systèmes Multi-Agents 1998*. Hermés.

[Newell, 1982] Newell, A. (1982). The Knowledge Level. *Artificial Intelligence*, 18 :87–127.

[Nilsson, 1995] Nilsson, N. J. (1995). Eye on the prize. *AI Magazine*, 16(2) :9–17.

[Nodine, 1998] Nodine, M. (1998). The InfoSleuth agent system. *Lecture Notes in Computer Science*, 1435 :19.

[Occello et Demazeau, 1997] Occello, M. et Demazeau, Y. (1997). Vers une approche de conception et de description récursive en univers multi-agents. In Quinqueton, J., Thomas, M.-C., et Trousse, B., editors, *Actes des 5èmes journées francophones JFIADSMA'97*, pages 63–75. Hermès.

[OMG, 1992] OMG (1992). *The Common Object Request Broker : Architecture and Specification*. Object Management Group, Inc., 492 Old Connecticut Path, Framingham, MA 01701, 1.1 edition.

[OMG, 1999a] OMG (1999a). Agent working group reports.

[OMG, 1999b] OMG (1999b). The Unified Modeling Language specification version 1.3. http ://www.omg.org.

[OMG, 2000] OMG (2000). *XML Metadata Interchange (XMI) Specification*. Object Management Group, Inc., 492 Old Connecticut Path, Framingham, MA 01701.

[OMG, 2001] OMG (2001). *An introduction to the Model Driven Architecture*. Object Management Group, Inc., 492 Old Connecticut Path, Framingham, MA 01701.

[Ossowski, 1999] Ossowski, S. (1999). *Co-ordination in artificial agent societies : social structures and its implications for autonomous problem-solving agents*, volume 1535 of *Lecture Notes in*

Computer Science and Lecture Notes in Artificial Intelligence. Springer-Verlag Inc., New York, NY, USA.

[Ouzrout et al., 1996] Ouzrout, Y., Kabachi, N., et Vincent, L. (1996). Une société d'agents pour la prise de décision dans les organisations productives. In Muller, J.-P. et Quinqueton, J., editors, *Actes des 4èmes journées francophones sur l'Intelligence Artificielle distribuée et les systèmes multi-agents (JFIADSMA'96)*. Hermès.

[Parsons, 1951] Parsons, T. (1951). *The Social System*. The Free Press, Glencoe, IL.

[Parunak, 1998] Parunak, H. V. D. (1998). What can agents do in industry, and why ? an overview of industrially-oriented R&D at CEC. *Lecture Notes in Computer Science*, 1435 :1.

[Parunak et Odell, 2001] Parunak, H. V. D. et Odell, J. (2001). Representing social structures in UML. In *Proceedings of Autonomous Agents 2001*.

[Parunak et al., 1998] Parunak, H. V. D., Savit, R., et Riolo, R. L. (1998). Agent-based modeling vs. equation-based modeling : A case study and users guide. *Lecture Notes in Computer Science*, 1534 :10–25.

[Parunak et Vanderbok, 1997] Parunak, H. V. D. et Vanderbok, R. S. (1997). Managing emergent behavior in distributed control systems. In *Proceedings of ISA Tech '97, Instrument Society of America*.

[Pierce, 1996] Pierce, B. C. (1996). Foundational calculi for programming languages. In Tucker, A. B., editor, *Handbook of Computer Science and Engineering*, chapter 139. CRC Press.

[Pitt et Mandani, 1999] Pitt, J. et Mandani, A. (1999). Some remarks on the semantics of FIPA's agent communication language. *Autonomous Agents and Multi-Agent Systems*, 2(4) :333–356.

[Platt, 2001] Platt, D. S. (2001). *Introducing the .NET platform*. Microsoft Press.

[Prasad et al., 1995] Prasad, M. V. N., Lesser, V. R., et Lander, S. E. (1995). Learning organizational roles in a heterogeneous multi-agent system. In Lesser, V., editor, *Proceedings of the First International Conference on Multi–Agent Systems*. MIT Press.

[R. Kelsey, 1998] R. Kelsey, W. Clinger, J. R. (1998). Revised5 report on the algorithmic language scheme. *Higher-Order and Symbolic Computation*, 11(1).

[Rao, 1996] Rao, A. S. (1996). AgentSpeak(L) : BDI agents speak out in a logical computable language. In van Hoe, R., editor, *Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, Eindhoven, The Netherlands.

[Rashid et al., 1989] Rashid, R., Baron, R., Forin, A., Golub, D., Jones, M., Julin, D., Orr, D., et Sanzi, R. (1989). Mach : A foundation for Open Systems. In *Proceedings of the 34th Computer Society Ithe Second Workshop on Workstation Operating Systems(WWOS2)*.

[Reenskaug et al., 1996] Reenskaug, T., Wold, P., et Lehne, O. A. (1996). *Working with objects : the OOram Software Engineering Method*. Manning Publications Co., Greenwich.

[Resnick, 1994] Resnick, M. (1994). *Turtles, termites, and traffic jams : Explorations in massively parallel microworlds*. Bradford Books/MIT Press, Cambridge, Mass.

[Reticular Systems, 2000] Reticular Systems, I. (2000). AgentBuilder, an integrated toolkit for constructing multi-agent systems. Technical Report.

[Revault, 1996] Revault, N. (1996). *Principes de méta-modélisation pour l'utilisation de canevas d'applications à objets (MetaGen et les frameworks)*. Thèse d'Informatique, Université Pierre et Marie Curie (Paris VI), LAFORIA.

[RMP, 1999] RMP (1999). Potlatch, ants, and interstitial thoughts. In *RMP 1999 Cont. Proceedings*, Montpellier.

[Rocher, 1968] Rocher, G. (1968). *Introduction à la sociologie générale*. Points Seuil.

[Sadek, 1991] Sadek, M. (1991). *Attitudes mentales et interaction rationnelle : vers une théorie formelle de la communication*. PhD thesis, Université de Rennes I.

[Schillo et al., 2000] Schillo, M., Fischer, K., et Klein, C. T. (2000). The micro-macro link in DAI and sociology. In Moss, S. et Davidsson, P., editors, *Proceedings of the second international workshop on multi-agent-based simulation (MABS'2000)*, number 1979 in Lecture Notes in Computer Science. Springer.

[Servat et al., 1998] Servat, D., Perrier, E., Treuil, J.-P., et Drogoul, A. (1998). When agents emerge from agents : Introducing multi-scale viewpoints in multi-agent simulations. *Lecture Notes in Computer Science*, 1534 :183–198.

[Shapiro et al., 1998] Shapiro, S., Lespérance, Y., et Levesque, H. J. (1998). Specifying communicative multi-agent systems with ConGolog. In Wobcke, W., Pagnucco, M., et Zhang, C., editors, *Agents and Multi-Agent Systems—Formalisms, Methodologies, and Applications*, pages 1–14. Springer-Verlag, Berlin.

[Shoham, 1991] Shoham, Y. (1991). AGENT0 : A simple agent language and its interpreter. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, pages 704–709, Anaheim, CA.

[Sichman, 1995] Sichman, J. (1995). *Du raisonnement social chez les agents : une approche basée sur la théorie de la dépendance*. PhD thesis, Institut National Polytechnique de Grenoble.

[Sierra et Agustí, 1991] Sierra, C. et Agustí, J. (1991). Colapses : Towards a methodology and a language for knowledge engineering. In *11th Intl. Conference on Expert Systems AVIGNON'91*, pages 407–423.

[Singh, 1991] Singh, M. P. (1991). Group ability and structure. In Demazeau, Y. et M"uller, J.-P., editors, *Decentralized A.I. 2 : Proc. of the 2nd European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, pages 127–145. North-Holland, Amsterdam.

[Smith, 1980] Smith, R. G. (1980). The contract net protocol : high-level communication and control in a distributed problem solver. *IEEE transactions on computers*, 29(12) :1104–1113.

[Steimann, 2000] Steimann, F. (2000). A radical revision of UML's role concept. In Evans, A., Kent, S., et Selic, B., editors, *UML 2000, Proceedings of the third international conference on the Unified Modeling Language*, number 1939 in Lecture Notes in Computer Science, pages 194–209. Springer.

[Steiner, 1997] Steiner, D. (1997). An overview of FIPA 97. Technical report, FIPA Consortium.

[Tambe, 1997] Tambe, M. (1997). Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7 :83–124.

[Tambe et Zhang, 2000] Tambe, M. et Zhang, W. (2000). Towards flexible teamwork in persistent teams : Extended report. *Autonomous Agents and Multi-Agent Systems*, 3(2) :159–183.

[Treur, 1998] Treur, J. (1998). Report of 1st SIG meeting. Methodology and Software Engineering Special Interest Group. AgentLink Organization.

[Warmer et Kleppe, 1999] Warmer, J. B. et Kleppe, A. G. (1999). *The Object Constraint Language : precise modeling with UML*. Addison-Wesley.

[Weber, 1995] Weber, M. (1995). *Economie et société*. Agora.

[Wegmann et Genilloud, 2000] Wegmann, A. et Genilloud, G. (2000). The role of "roles" in use case diagrams. In Evans, A., Kent, S., et Selic, B., editors, *UML 2000, Proceedings of the third international conference on the Unified Modeling Language*, number 1939 in Lecture Notes in Computer Science. Springer.

[Werner, 1992] Werner, E. (1992). The design of multi-agent systems. In Werner, E. et Demazeau, Y., editors, *Decentralized AI 3 — Proceedings of the Third European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW-91)*, pages 3–30. Elsevier Science Publishers B.V. : Amsterdam, The Netherlands.

[White, 1996] White, J. E. (1996). Telescript technology : Mobile agents. In Bradshaw, J., editor, *Software Agents*. AAAI Press/MIT Press, Menlo Park, Cal. Also available as General Magic White Paper.

[Wieringa et al., 1994] Wieringa, R., de Jonge, W., et Spruit, P. (1994). Roles and dynamic subclasses : a modal logic approach. In Tokoro, M. et Pareschi, R., editors, *Proceedings of the 8th European Conference on Object-Oriented Programming, ECOOP'94*, volume 821 of *Lecture Notes in Computer Science*, pages 32–59. Springer.

[Willmott et Burg, 2000] Willmott, S. et Burg, B. (2000). Agentcities : an open testbed for FIPA agent based services and applications. Technical report, FIPA Consortium.

[Willmott et al., 1999] Willmott, S., Faltings, B., Calisti, M., Macho-Gonzales, S., Belakhdar, O., et Torrens, M. (1999). Constraint choice language (CCL), language specification v2.0. Technical Report 99/320, EPFL Departement d'Informatique.

[Winch, 1958] Winch, P. (1958). *The idea of a social science*. Humanities Press International, Atlantic Highlands, NJ.

[Wooldridge, 1996] Wooldridge, M. (1996). A logic for BDI planning agents. In Schobbens, P.-Y., editor, *Working Notes of 3rd ModelAge Workshop : Formal Models of Agents*, Sesimbra, Portugal.

[Wooldridge et al., 1999] Wooldridge, M., Jennings, N. R., et Kinny, D. (1999). A methodology for agent-oriented analysis and design. In *Proceedings of the Third International Conference on Autonomous Agents (Agents'99)*, pages 69–76, Seattle, WA, USA. ACM Press.

[Wooldridge et al., 2000] Wooldridge, M., Jennings, N. R., et Kinny, D. (2000). The gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems*, 3(3) :285–312.

[Yoo, 1999] Yoo, M.-J. (1999). *Une approche componentielle pour la modélisation d'agents coopératifs et leur validation*. PhD thesis, Université Paris 6.

[Yuasa, 1990] Yuasa, T. (1990). Design and implementation of Kyoto Common Lisp. *Journal of Information Processing*, 13(3) :284–295.

Annexe A

Le langage PICOL

Le langage PICOL (pour "PI-Calculus Oriented Language") est utilisé à quelques reprises dans ce document pour décrire le comportement de systèmes multi-agents. Nous allons en donner ici une définition précise. PICOL est une extension (syntaxique) du π-calcul qui en conserve les caractéristiques principales tout en en facilitant l'usage pour la description de processus.

A.1 Syntaxe

Voici la syntaxe de PICOL . Les metavariables suivantes seront utilisées : P et Q désignent des processus, E une expression, C dénote des conditions et x ou v des noms ou variables. F désignera des fonctions que l'on supposera primitives.

$$
\begin{array}{lll}
P, Q & ::= & \texttt{nil} \\
 & & P \mid Q \\
 & & x\texttt{(}v_1\texttt{,..,}v_n\texttt{).}P \\
 & & \overline{\texttt{x}}\texttt{(}E_1\texttt{,..,}E_n\texttt{).}P \\
 & & C\texttt{?}P \\
 & & C\texttt{?}P\ \texttt{:}Q \\
 & & C_1\texttt{?}P_1\ \texttt{;..;}C_n\texttt{?}P_n \\
 & & \nu(v)P \\
 & & \texttt{!}P \\
 & & \texttt{match(x)(}f_1 \rightarrow P_1; ..; f_k \rightarrow P_k\texttt{)} \\
 & & A(E_1, .., E_n) \\
C & ::= & \texttt{(}E_1\ \textit{rel-op}\ E_2\texttt{)} \\
 & & \texttt{(}C_1\ \texttt{or}\ C_2\texttt{)} \\
 & & \texttt{(}C_1\ \texttt{and}\ C_2\texttt{)} \\
 & & \texttt{not}\ C
\end{array}
$$

E	::=	$(E_1 \ \ op \ \ E_2)$
		$F(E)$
		x
		numerical constant
		"string"
op	::=	$+, *, -, /, ..$
rel-op	::=	$=, \neq, <, >, \leq, \geq, \in, \notin$

A.2 Sémantique

La sémantique opérationnelle de PICOL en termes de relations de réduction est la suivante :

Concurrence		
$\{\|(P \mid Q)\|\}_{\mathtt{g}}$	$\rightleftharpoons$	$\{\|P, Q\|\}_{\mathtt{g}}$
$\{\|\overline{\mathtt{x}}e_1, .., e_n.P,$		
$x(v_1, .., v_n).Q\|\}_{\mathtt{g}}$	$\rightarrow$	$\{\|P, Q[e'_1/v_1, .., e'_n/v_n]\|\}_{\mathtt{g}}$
		if $\forall i \, e_i \rightarrow_p e'_i$
$\{\|\nu(v)P\|\}_{\mathtt{g}}$	$\rightarrow$	$\{\|P\|\}_{\mathtt{g}}$
		if $v \notin$ free variables of P
Conditions		
$\{\|(C \, ?P)\|\}_{\mathtt{g}}$	$\rightarrow$	$\{\|P\|\}_{\mathtt{g}}$ if $C \rightarrow_p$ True
	$\rightarrow$	$\{\|\mathtt{nil}\|\}_{\mathtt{g}}$ if $C \rightarrow_p$ False
$\{\|(C \, ?P \, :Q)\|\}_{\mathtt{g}}$	$\rightarrow$	$\{\|P\|\}_{\mathtt{g}}$ if $C \rightarrow_p$ True
	$\rightarrow$	$\{\|Q\|\}_{\mathtt{g}}$ if $C \rightarrow_p$ False
$\{\|C_1 \, ?P_1 \, ; .. \, ; C_n \, ?P_n\|\}_{\mathtt{g}}$	$\rightarrow$	$\{\|P_i\|\}_{\mathtt{g}}$ if $C_i \rightarrow_p$ True
Primitives		
$(e_1$ op $e_2)$	$\rightarrow_p$	$Apply(op, e'_1, e'_2)$
		if $e_1 \rightarrow_p e'_1$ and $e_2 \rightarrow_p e'_2$
$F(e)$	$\rightarrow_p$	$Apply(F, e')$ if $e \rightarrow_p e'$
$(e_1$ and $e_2)$	$\rightarrow_p$	$e'_1 \wedge e'_2$ if $e_1 \rightarrow_p e'_1$
		and $e_2 \rightarrow_p e'_2$
$(e_1$ or $e_2)$	$\rightarrow_p$	$e'_1 \vee e'_2$ if $e_1 \rightarrow_p e'_1$
		and $e_2 \rightarrow_p e'_2$
not e	$\rightarrow_p$	$\neg e'$ if $e \rightarrow_p e'$

A.3 Primitives

Pour faciliter l'expression, des constructions de haut niveau sont utilisées. On fait la supposition qu'une structure de liste (telle que définie en SCHEME ou LISP) est primitive. On utilisera des primitives telles que `cons`, `car`, `cdr`, `assoc`, `member` et `map` avec la même définition qu'en Scheme [R. Kelsey, 1998]. Pour juger de la validité de l'expression d'une structure de liste en π-calcul, on se tournera vers [Milner et al., 1992] qui en donne une expression.

Un dictionnaire est défini comme une liste associative. On a donné aux fonctions `at`, `atput`, `values`, `keys`, et `hasKeys` la même définition qu'en SMALLTALK. On pourra noter que dans la fonction `hasKeys` le dictionnerait envoie un message `keys` à lui-même et traite la liste de clés via une continuation :

```
Dictionary(id,vals) =def
  id(msg).
    match(msg)(
      at(key,c) → c̄tl(assoc(key, vals)),
      atput(key,val) →
        Table(id,(cons(cons(key,val),vals))),
      atputlist(key,val) →
        ν(c1).(īdat(key, c1).c1(lst).
          īdatput(key, cons(val, lst))
      values(c) → c̄map(cdr, vals),
      keys(c) → c̄map(car, vals),
      hasKeys(key,c) →
        ν(c1)(īdkeys(c1).c1(lst).
        c̄member(key, lst)))
```

Annexe B

Modèles de données

Nous donnons ici les définitions complètes des structures de données de description d'organisations abstraites et concrétisées évoquée au chapitre 7. Le format retenu pour les exprimer est un modèle de document [1] XML tel que définit par le W3 Consortium [Bray et al., 1998].

B.1 Description de structures organisationnelles : A-ORGML

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!-- Définition d'un type de document XML pour la représentation
     de structures d'organisation -->

<!-- Extensions

  Ces trois définitions permettent à un concepteur d'imaginer des
  descriptions de type de document basées sur celle-ci, mais en
  rajoutant des concepts spécifiques à son domaine aux notions de
  groupe, rôle ou organisation. Cela se fait simplement en
  redéfinissant ces trois entités, ici laissées vides -->

<!ENTITY % organisation-def-extension "" >
<!ENTITY % group-def-extension "" >
<!ENTITY % role-def-extension "" >

<!--
  La liste des éléments permis: une structure d'organisation
  est définie comme une suite de définitions de groupe, qui
  comprendront des définitions de rôles et éventuellement des
  indications d'interaction -->

<!ELEMENT organisation-def (group-def+ %organisation-def-extension;)>
<!ELEMENT group-def (role-def*, interaction* %group-def-extension;)>
<!ELEMENT role-def (constraint*, definition* %role-def-extension;)>
```

[1] Plus exactement une DTD, Document Type Definition

```
<!ATTLIST organisation-def
        id ID #IMPLIED
        description CDATA #IMPLIED>

<!ATTLIST group-def
        id ID #REQUIRED
        description CDATA #IMPLIED>

<!ATTLIST role-def
        id ID #REQUIRED
        description CDATA #IMPLIED>


<!--
  L'interaction se définit entre points finaux, identifiés par
  leurs rôles -->

<!ELEMENT interaction (endpoint*, definition*)>
<!ELEMENT endpoint EMPTY>

<!ATTLIST interaction
        id ID #REQUIRED
        description CDATA #IMPLIED>

<!ATTLIST endpoint
        role IDREF #REQUIRED
        type (optional | initiator) #IMPLIED
        arity CDATA #IMPLIED>

<!--
  La contrainte peut être représentée directement, soit à un niveau
  générique (par l'arité), soit en étant exprimée dans un formalisme
  spécifique -->

<!ELEMENT constraint (#PCDATA)>

<!ATTLIST constraint
        type CDATA #IMPLIED
        arity CDATA #IMPLIED
        description CDATA #IMPLIED>

<!--
  Une définition peut être utilisée lors de l'utilisation de
  définition de rôle ou de groupe, pour donner une spécification
  externe (par exemple des règles pour un comportement, un
  protocole précis pour une interaction, ...)  -->

<!ELEMENT definition (#PCDATA)>

<!ATTLIST definition
        type CDATA #IMPLIED
```

```
          ref CDATA #IMPLIED>
```

B.2 Description d'organisations concrétisées : C-ORGML

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!-- Définition de type de document XML pour la représentation
     d'organisations concrètes -->


<!-- Extension

  Cette extension permet à un concepteur d'imaginer des
  descriptions de type de document basées sur celle-ci, mais en
  rajoutant des concepts spécifiques à son architecture d'agent
  (par exemple pour pouvoir stocker l'état interne de l'agent à
  un moment donner afin de recréer à l'identique le système plus
  tard). Cela se fait simplement en redéfinissant l'entité, ici
  laissée vide -->

<!ENTITY % agent-extension "" >

<!ELEMENT multi-agent-system (individuals,organisation+)>
<!ELEMENT organisation (group+)>

<!ELEMENT group (role+)>
<!ELEMENT role EMPTY>

<!ELEMENT individuals (agent+)>
<!ELEMENT agent (property* %agent-extension;)>

<!ELEMENT property EMPTY>
<!ATTLIST property
    name   NMTOKEN  #REQUIRED
    value  CDATA    #REQUIRED
>



<!-- Element Attributes -->

<!ATTLIST organisation
        name CDATA #IMPLIED
def-id CDATA #IMPLIED
>

<!ATTLIST group
        name CDATA #IMPLIED
def-id CDATA #IMPLIED
```

```
>

<!ATTLIST role
        name CDATA #REQUIRED
        handler IDREF #REQUIRED
>

<!ATTLIST agent
        id ID #REQUIRED
type CDATA #IMPLIED
>
```

Proposition d'un modèle organisationnel générique de systèmes multi-agents et examen de ses conséquences formelles, implémentatoires et méthologiques.

Résumé

Cette thèse présente le résultat de notre travail sur l'étude des structures organisationnelles dans les systèmes multi-agents. L'approche proposée met l'accent sur la primauté de l'organisation multi-agents sur les architectures individuelles pour répondre aux besoins d'hétérogénéité,d'adaptation et de contrôle dans les systèmes de taille importante. Le coeur de ce travail est donc un modèle générique de description d'organisation, basé sur les notions de groupe, agent et rôle. Il est montré comment cette description structurelle et fonctionnelle permet de fédérer des systèmes multi-agents ayant des architectures de contrôle, des buts ou des modèles d'interactions différents. Nous dégageons alors plusieurs conséquences de ce modèle, selon les points de vue de la formalisation, de l'implémentation, de la conception ou des applications. Nous présentons en particulier une expression dans une variante du pi-calcul, que nous illustrons par l'étude du meta-niveau dans cette formalisation. L'aspect opératoire a été abordé par la réalisation une plate-forme générique de développement et exécution de systèmes multi-agents, basée sur le modèle d'organisation proposé et conçue pour permettre l'accueil de systèmes ayant des architectures fortement hétérogènes. Nous montrons alors comment un lien peut être fait avec d'autres approches en conception multi-agent, et proposons quelques pistes pour l'utilisation dans le cadre des langages de modélisation. La validation expérimentale de ce travail est finalement abordée par des expérimentations ciblées sur le modèle et la plate-forme proposée ainsi que par des applications à visée plus générique.

Mots-clés

Intelligence artificielle distribuée, systèmes multi-agents, organisations, plate-formes agents, méthodologie

---

Proposal for a generic organization model for multi-agent systems. Study of its methodological, formal and implementation consequences.

Abstract

This work investigates the use of organizational abstractions in multi-agent systems. The thesis shows how adopting a social-based point of view on these systems, instead of a classic individual-based assumption, may be an answer to increasing needs in heterogeneity, adaptation and control in large-scale agent systems. The main proposition is a generic model of agent organizations, based on the concepts of agent, group and role. We show how a structural and functional decomposition can federate multi-agent system. We focus on agents with highly heterogeneous internal architectures, possibly antagonist goals and with different interaction models. The model is then formally expressed in a pi-calculus variant. We illustrate its operational semantics through an overview of meta-level design. Implementation aspects are also discussed through the presentation of MadKit, a generic agent infrastructure toolkit, based on our organization model and designed to host multiple architectures. Empirical results validate our contribution, through architecture-specific evaluations and real-world applications.

Keywords

Distributed artificial intelligence, multi-agent systems, organizations, agent platforms, methodology