Numéro d'identification :

Académie de Pau

Université de Pau et des Pays de l'Adour

— **Sciences et Techniques de Pau** —

Thèse

présentée à l'Université de Pau et des Pays de l'Adour
pour obtenir le diplôme de DOCTORAT

Spécialité	: Informatique
Formation Doctorale	: *Informatique*
Ecole Doctorale	: *Information, Structures, Systèmes*

Un modèle de gestion distribuée de groupes ouverts et dynamiques d'agents mobiles

par

Saber Mansour

Soutenue le 10 Décembre 2007 devant le Jury composé de :

M. Guy Gouardères , Professeur, Université De Pau et Des Pays de l'Adour, ... Directeur de Thèse
M. Jacques Ferber , Professeur, Université Montpellier II, ... Directeur de Thèse

A mon père et ma mère.

Moralité

Un commerçant possédait un perroquet plein de dons. Un jour, il décida de partir en Inde et demanda à chacun quel cadeau il désirait qu'on lui rapporte du voyage. Quand il posa cette question au perroquet, celui-ci répondit :

« En Inde, il y a beaucoup de perroquets. Va les voir pour moi. Décris-leur ma situation, cette cage. Dis-leur : " Mon perroquet pense à vous, plein de nostalgie. Il vous salue. Est-il juste qu'il soit prisonnier alors que vous volez dans le jardin de roses ? Il vous demande de penser à lui quand vous voletez, joyeux, entre les fleurs. " ? »

En arrivant en Inde, le commerçant se rendit en un lieu où il y avait des perroquets. Mais, comme il leur transmettait les salutations de son propre perroquet, l'un des oiseaux tomba à terre, sans vie. Le commerçant en fut très étonné et se dit :

« Cela est bien étrange. J'ai causé la mort d'un perroquet. Je n'aurais pas dû transmettre ce message. »

Puis, quand il eut fini ses achats, il rentra chez lui, le coeur plein de joie. Il distribua les cadeaux promis à ses serviteurs et femmes. Le perroquet lui demanda :

« Raconte-moi ce que tu as vu afin que je sois joyeux moi aussi. »

A ces mots, le commerçant se mit à se lamenter et à exprimer ses regrets.

« Dis-moi ce qui s'est passé, insista l'oiseau. D'où te vient ce chagrin ? »

Le commerçant répondit : « Lorsque j'ai transmis tes paroles à tes amis, l'un d'eux est tombé à terre sans vie. C'est pour cela que je suis triste. » A cet instant, le perroquet du commerçant tomba lui même aussi dans sa cage, inanimé. Le commerçant, plein de tristesse, s'écria :

« Ô mon perroquet au langage suave ! Ô mon ami ! Que s'est-il donc passé ? Tu étais un oiseau tel que Salomon n'en avait jamais connu de semblable. J'ai perdu mon trésor ! »

Après avoir longtemps pleuré, le commerçant ouvrit la cage et jeta le perroquet par la fenêtre. Aussitôt, celui-ci s'envola et alla se percher sur une branche d'arbre. Le commerçant, encore plus étonné, lui dit : « Explique-moi ce qui se passe ! »

Le perroquet répondit :

« Ce perroquet que tu as vu en Inde m'a expliqué le moyen de sortir de prison. Par son exemple, il m'a donné un conseil. Il a voulu me dire : " Tu es en prison parce que tu parles. Fais donc la mort." Adieu, ô mon maître ! Maintenant je m'en vais. Toi aussi, un jour, tu rejoindras ta patrie. »

Mawlana Djalal al-Din Rumi (Le Mesnevi- La cage)

Remerciements

Je remercie respectueusement mes directeurs de thèse, les Professeurs Guy Gouardères et Jacques Ferber, pour leurs conseils, leurs encouragements, leurs confiance, tout ce par quoi ils ont su orienter ma liberté de recherche.

Mes respects et ma gratitude vont également aux membres de mon jury qui m'ont fait l'honneur de juger ce travail en venant pour certains de fort loin et qui tous, par leur disponibilité, leurs observations et leurs rapports m'ont permis de l'enrichir.

∞

Mener à terme un doctorat comporte essentiellement deux aspects. Le premier concerne bien sûr la rédaction du document de thèse. Le deuxième a trait à l'organisation de l'événement si particulier que constitue le jour de sa soutenance. Faire l'une ou l'autre de ces deux choses sans l'aide et le soutien d'autres personnes est certainement impossible. Merci à xxxxx pour avoir immensément contribué à chacun de ces deux aspects. Merci à Olivier Simonin pour toute l'aide qu'il m'a apportée de nombreuses fois durant toutes ces années. Merci à xxxxx de m'avoir fait participer à une activité de recherche si passionnante et inspirante (thank you Danny). Merci à xxxxx pour leur disponibilité lors des pré soutenances. Merci à xxxxx pour son immense contribution à l'organisation du jour J. A ce propos, je remercie Nicole Olivet, Nadine Tilloy et Florence Picone pour leur précieuse aide. Je tiens à remercier xxxxx pour m'avoir si amicalement accueilli et pour les précieux conseils qu'ils m'ont donnés pour la soutenance. Merci à xxxxx pour ses nombreux coups de main (recherche et autres). Merci à xxxxx pour m'avoir fait l'amitié d'être présent le jour de ma soutenance.

Mener à terme un doctorat suppose aussi bien sûr que les choses de la vie vous en aient donnés l'occasion. je pense à tous mes professeurs sans lesquels ce travail n'existerait pas et plus spécialement à xxxxx qui, à Lyon, m'a orienté vers les systèmes multi-agents. J'ai une pensée pour xxxxx qui m'a fait découvrir le domaine de l'Intelligence Artificielle. Merci à xxxxx pour m'avoir fait partager son expérience du doctorat alors que j'étais en maîtrise. Je remercie xxxxx pour m'avoir permis de faire mes premières armes dans l'enseignement en tant que tuteur à l'université.

Mener à terme un doctorat est aussi une expérience humaine. Je remercie tous ceux qui m'ont aidé dans cette "aventure". Merci à xxxxx. Merci à xxxxx pour avoir été à mes côtés le jour où il le fallait..

Je remercie ma mère et je remercie mon père.

Merci à xxxxx.

« The most incomprehensible thing about the universe is that it's comprehensible. »

Albert Einstein

Sommaire

Chapitre 1

Introduction

La conception et le déploiement de systèmes complexes largement distribués ouverts et dynamiques constituent aujourd'hui un enjeu majeur pour les chercheurs dans plusieurs disciplines de l'informatique. En effet, la vulgarisation d'Internet et l'émergence de la grille de calcul, combinées à l'explosion du nombre de supports à grande capacité de calcul ont posé de nouveaux problèmes de dynamique, instabilité et surcharge aux systèmes informatiques. Comment concevoir des systèmes informatiques tirant profit des performances des nouvelles technologies matérielles et adaptés aux nouveaux besoins des utilisateurs de plus en plus exigeants ? L'enjeu majeur auquel sont confrontés les chercheurs aujourd'hui est de résoudre ces problèmes tout en réduisant le fossé entre les nouveaux besoins et les techniques existantes. L'idée de représenter l'utilisateur par un agent qui prend en charge ses besoins, offre ses services et interagit avec les autres agents du réseau afin de le satisfaire, est l'une des pistes qui est pour plusieurs chercheurs la plus prometteuse pour résoudre ces problèmes.

En effet, les Systèmes Multi-Agents SMA ont atteint un certain degré de maturité qui laisse entrevoir en cette technologie la réponse à plusieurs problématiques. Les SMA apportent des concepts nouveaux tels que l'autonomie, la pro activité, sociabilité permettant une plus grande flexibilité et comblant des manques de certaines technologies existantes : Web/Grid-Services ou les réseaux P2P. C'est dans ce contexte que Foster dans « Brain Meets Brawn : Why Grid and Agents Need Each Other» [Foster *et al.* , 2004] compare les agents au cerveau et la grille aux muscles. Il explique pourquoi et comment les systèmes multi-agents apportent la flexibilité nécessaire à la grille de calcul. Combinés à ces technologies émergentes les SMA trouveront sans doute leurs lettres de noblesse et dépasseront le cadre des applications fermées pour être déployés à grande échelle sur des réseaux ouverts et dynamiques.

La thématique du discours que nous allons entreprendre gravite autour de cette idée : comment concevoir et bâtir des systèmes multi-agents permettant de répondre aux nouveaux besoins logiciels. Nous montrerons qu'en adoptant un point de vue essentiellement social, des systèmes multi-agents, axé sur les concepts d'ouverture de dynamique et de distribution peut faciliter le déploiement à grande échelle des SMA qui pourraient devenir plus rapidement un paradigme de programmation comme un autre. La finalité de ces recherches est de proposer des pistes pour comprendre, concevoir et mettre en oeuvre des systèmes multi-agents de très grande taille, ouverts dynamiques et totalement décentralisés. Nous essayons de faciliter et de rendre plus naturel l'intégration des SMA avec d'autres technologies destinées à une utilisation à grande échelle.

Le lecteur croisera, en parcourant ce document, plusieurs modèles, méthodologies et implémentations qui pourront l'aider à mieux appréhender des problèmes complexes dans différents

domaines avec une vue organisationnelle. Nous tentons de réduire le fossé entre un besoin, et l'élaboration du système qui l'incarnera. Nous donnerons des exemples concrets de résolution de problèmes complexes grâce à la méthodologie que nous proposons et au déploiement de notre modèle.

1.1 Contexte de la thèse

C'est dans ce contexte celui de la conception des systèmes multi-agents organisationnels, ouverts, dynamiques, distribué et déployés à grande échelle, que cette thèse s'inscrit. Notre intérêt pour ce type d'approche s'est manifesté en étant confronté au cas particulier de la qualification des connaissances acquises de façon informelle [Minko & Gouardères, 2004] par des étudiants répartis dans l'environnement dynamique de la grille de calcul [Foster & Kesselman, 1999].

Ce processus de qualification s'effectue sur deux phases. La première est quand l'étudiant humain est entrain d'effectuer de l'apprentissage formel (sur les simulateurs), les agents enregistrent ses activités et détectent ses erreurs par rapport à ce qui est attendu. La deuxième phase, celle qui nous intéresse le plus, c'est lors du passage de l'agent humain de l'apprentissage formel à la partie informelle. L'étudiant doit entrer en contact informel avec d'autres membres. De ces interactions, l'agent humain va pouvoir acquérir de nouvelles connaissances et compétences. Le rôle des agents artificiels est de résoudre deux problèmes : mettre l'humain dans le groupe le plus adapté à ses compétences et connaissances, sachant que cette position n'est pas figée et qu'elle change tant que les connaissances de l'agent humain et des autres membres évoluent. Le deuxième point c'est de pouvoir qualifier les connaissances acquises par l'humain de façon informelle [Gouardères *et al.* , 2006].

Il est évident que plusieurs difficultés se présentent pour atteindre ces objectifs. Nous ne nous intéressons pas au problème épineux de la communication entre agents humains et artificiels, pour lequel les solutions proposées par différentes études sont loin de satisfaire tous les besoins. Notre tâche était de trouver les mécanismes pour qualifier les connaissances acquises de façon informelle par les agents humains, de permettre à des agents à priori hétérogènes d'interagir afin de former des groupes cohérents, et finalement de pouvoir gérer des groupes dynamiques et aux frontières fluides formées par des agents artificiels autonomes instables, mobiles et dynamiques. La difficulté est d'autant plus importante que même la composition et les règles qui régissent ces groupes ne sont pas connues à l'avance et que, une fois établies, elles sont susceptibles de changer selon l'évolution des connaissances des membres.

Nous avons constaté rapidement, en étudiant les solutions existantes [Marcio & Yu, 2002], que ce dernier point dépasse le cadre de notre application et constitue sans doute un axe de recherche très important et un enjeu pour la discipline multi-agents afin de répondre aux besoins des nouvelles applications logicielles. Notre travail se focalisera en premier lieu sur l'étude, la résolution et l'implémentation de ce problème de gestion des groupes d'agents distribués, dynamiques aux frontière fluides et déployés à grande échelle. Ensuite, nous reviendrons sur le problème initial de la qualification en expliquant comment la solution que nous proposons est utilisée pour résoudre ce problème.

1.2 Contribution

La particularité de ce mémoire est de proposer une approche organisationnelle pour les SMA ouverts, dynamiques et déployés à grande échelle. Il en résulte un modèle nommé AGRS

(pour Agent Groupes Rôles Serviceables) [Mansour & Ferber, 2007b], dont le développement a nécessité la résolution de différents problèmes :

Modèle organisationnel ouvert et dynamique : Comme nous le verrons au chapitre 2, les approches organisationnelles, basées sur le concept de rôle, l'étudient tantôt du point de vue de l'organisation (les agents sont liés aux rôles), tantôt du point de vue de l'agent considéré (dans ce cas) comme une entité à laquelle des rôles sont rattachés. Certains travaux essaient de trouver une association entre ces différentes appréhensions du concept de rôle et de son utilisation dans les organisations SMA. Notre première contribution, détaillée dans le chapitre 3, est de proposer un modèle qui combine ces deux aspects mais qui, en plus, étudie le rôle à la fois du côté de l'agent qui le joue et celui qui l'utilise. Nous introduisons le concept de service associé aux rôles. Nous expliquerons comment la recherche/découverte et la publication des services permettent la dynamique et l'ouverture des groupes d'agents.

Groupes décentralisés : Problème principal de notre thèse la gestion des groupes ouverts d'agents, déployés à grande échelle est réalisée de façon décentralisée. Nous présentons les types de gestion pour les mémoires partagées et expliquons pourquoi nous situons notre travail dans le cadre du « knowledge pump ». Nous définissons les différents types de règles qui régissent les groupes et les rôles ainsi que la gestion événementielle des mises à jour.

Architecture hybride : Pour réaliser ce modèle de manière souple et robuste, nous avons choisi de combiner l'architecture agent avec les concepts Pair à Pair (P2P)[Wai-Sing Loo, 2007]. Nous expliquons ainsi le besoin d'utiliser les concepts P2P et les différents travaux combinant architecture agent et application P2P [Chen & Yeager, 2003]. L'architecture, ainsi conçue, profite des avantages des SMA à savoir autonomie, pro-acvitité et sociabilité et des avantages de systèmes P2P distribution, et déploiement à très grande échelle.

Mise en oeuvre : Le modèle, ainsi que l'architecture hybride proposées sont mis en oeuvre dans le noyau de la plateforme Madkit [Gutknecht, 2000] que nous avons complètement réimplémentée afin de prendre en compte les concepts du modèle AGRS et ceux des principes P2P pour nous permettant d'assurer la distribution à grane échelle.

Eléments de tests : la E-Qualficiation : Le point de départ de notre thèse (dans le cadre du projet ELeGI [Dimitrakos & Ritrovato, 2004] [Gaeta & Ritrovato, 2004]), à savoir, qualifier les échanges de connaissances informels entre étudiants travaillant dans un environnement distribué ouvert et dynamique, est repris pour tester et valider l'apport du modèle AGRS.

1.3 Plan de la thèse

Dans le chapitre 2, nous présenterons le contexte de la qualification dans le e-Learning [Allison *et al.* , 2003]. Nous donnerons tout d'abord quelques définitions de l'apprentissage formel et informel permettant de se faire une idée globale sur le problème que nous devons résoudre, à savoir, qualifier l'apprentissage acquis de façon informelle. Ceci nous amènera à identifier, d'une façon plus précise, les questions auxquelles nous devons apporter des réponses. Après cela, nous détaillerons les concepts fondamentaux de la e-Qualification, la solution que nous proposerons pour résoudre notre problème initial de qualifier l'apprentissage informel. C'est à partir de ces concepts de e-Qualification que nous abordons la problématique de cette thèse.

Le chapitre 3 consiste en une étude des concepts de base des modèles organisationnels où le rôle suscite un intérêt considérable. Nous commencerons par évaluer les différentes approches

en présentant les avantages et les défauts de chacune d'elles. Nous présenterons ensuite les propriétés qui manquent aux différentes approches pour répondre aux besoins de dynamique, ouverture et distribution des systèmes multi-agents. Parmi ces propriétés nous identifions la gestion distribuée des groupes. Les concepts des systèmes P2P nous permettent de mettre en oeuvre cette propriété.

Le chapitre 4 est le coeur de notre contribution. Un modèle d'organisation générique basé sur 4 notions clés d'agents, de groupe, de rôle et de service y est présenté. Ce modèle palliera aux défauts des modèles organisationnels existants d'une part, et permettra, d'autre part, de répondre aux besoins identifiés pour réaliser la e-Qualification. Nous verrons notamment qu'il pose les bases pour un modèle inspiré du modèle AGR [Ferber & Gutknecht, 1998] et qui prend en compte à la fois l'ouverture et la dynamique.

Dans le chapitre 5, nous exposons le problème de la gestion de la distribution dans les SMA. Nous commencerons par faire une étude de l'existant en expliquant comment les groupes d'agents peuvent être considérés comme des mémoires partagées. Nous exposerons ensuite les techniques existantes et la solution que nous proposons avec une gestion événementielle.

Le début du chapitre 6 sera consacré à l'étude du paradigme P2P et de son utilisation avec les SMA dans la littérature. Ensuite, nous présenterons les principaux protocoles de JXTA ainsi que l'architecture générale de la plateforme. Cela nous permettra de mieux expliquer comment nous combinerons les concepts P2P (publication, recherche, décentralisation, services...) avec ceux des SMA, et pourquoi la plateforme JXTA constitue une solution idéale pour des SMA ouverts, dynamiques, décentralisés et surtout déployés à très grande échelle.

Le chapitre 7, reprend l'idée de fondation pour notre modèle AGRS, mais cette fois-ci selon une perspective d'implémentation. L'intégration de la plateforme JXTA à la plateforme Madkit sera exposée. Nous détaillerons aussi les modifications apportées au noyau Madkit pour supporter les concepts de AGRS.

Dans le chapitre 8, nous reviendrons à notre problème de départ à savoir la e-Qualification nous montrerons comment le modèle AGRS permet de répondre aux questions que nous nous sommes posées au chapitre 2. Nous expliquerons comment nous exploiterons l'architecture hybride proposée et la platforme Madkit ainsi modifiée pour faciliter l'intégration des composants [Lacouture & Mansour, 2007].

Il ne nous restera donc plus qu'à revenir sur ce travail, pour en questionner le champ d'action et envisager quelques pistes pour renforcer le modèle et assurer sa robustesse et sa sécurité.

Chapitre 2

Le E-Learning et le processus de e-Qualification

La plupart des outils actuels conçus pour le e-Learning sont basés sur une évaluation formelle des apprentis. Ceci s'explique naturellement par le type de ces outils : QCM, exercices formels... Mais une observation concernant l'aspect social semble évidente. Cet aspect est peu ou pas pris en compte dans les outils de e-Learning actuels. Pourtant présent lors des phases de « debreifing » ou lors des échanges (Enseignant/Apprenant ou Apprenant/Apprenant) expliquant à l'apprenant ses erreurs ou plus généralement celui qui est passé sous silence dans les activités formalisées. Pour prendre en compte cet aspect social et tirer profil des discussions informelles qui ont lieu pendant avant et après les sessions de formation un autre type d'apprentissage est nécessaire il s'agit de l'apprentissage informel [Livingstone, 2006] en opposition aux techniques traditionnelles de l'apprentissage formel.

Dans l'apprentissage informel, l'accent est mis plus sur l'apprentissage comme pratique sociale que comme une stratégie pédagogique. Dès lors qu'il y a, dans une des phases de l'enseignement, une relation transactionnelle [Moore, 2006] apprenants/instructeurs et un contexte social, les dialogues, collaborations et les groupes d'apprentissages sont privilégiés. La connaissance et la réalité sont aussi vues comme des constructions sociales. Notre contribution vise à proposer une méthode pour évaluer et qualifier l'apprentissage acquis par les apprenants en intégrant l'aspect informel. La e-Qaulification est le processus que nous proposons pour atteindre cet objectif.

Nous essayerons en premier lieu de présenter les différentes théories d'apprentissage existantes : Comportemental et Constructiviste. Nous situerons ensuite le processus de e-Qualification dans ces deux paradigmes. Cela nous permettra d'introduire l'apprentissage collaboratif et la stratégie de réciprocité que nous considérons comme étant le meilleur contexte de l'apprentissage informel.

2.1 Apprentissage comportementaliste Vs Apprentissage constructiviste

2.1.1 La théorie comportementale

Avant d'exposer les relations qui existent entre les deux formes d'apprentissage formel et informel d'une part et les deux théories comportementalistes et constructiviste, nous allons définir les deux formes d'apprentissage.

- L' *apprentissage formel* rassemble les cours officiels approuvés autorisés et planifiés au sein de l'école et les sessions formelles. Cet apprentissage est structuré, il peut aboutir à une reconnaissance formelle (diplômes et certificats...).
- L' *apprentissage informel* C'est le processus de longue durée par lequel chaque personne acquiert et accumule connaissances et compétences à travers les interactions avec les autres étudiants avec lesquels partage leurs expériences et connaissances. Ce type d'apprentissage n'a pas lieu dans le cadre des sessions formelles. Il a lieu plutôt dans les processus qui ne sont pas structurés et qui ne sont pas contrôlés par le tuteur ni par les autres superviseurs.

2.1.2 La théorie comportementale

La théorie comportementale a émergé au début du 20ème siècle par Watson entre 1878-1958 [Watson, 1930]. Il a souligné le travail empirique basé sur des observations des comportements. La description comportementale du processus d'apprentissage ne prend en compte que les résultats qui sont : observables et mesurables négligeant les descriptions du développement individuel des stratégies cognitives et métacognitives et des autres processus internes.

Cette théorie est basée sur le modèle stimulus/réponse : Un enseignant pose une question et selon sa réponse, l'étudiant recevra une récompense ou un punition appropriée. Dans cette théorie, on ne considère que l'apprentissage a vraiment eu lieu que si on constate un changement observable et mesurable dans le comportement de l'apprenant. Le cerveau de l'étudiant est considéré comme une sorte de « boite noire » en ignorant les raisonnements possibles qui peuvent avoir lieu durant les leçons. L'apprentissage qui peut avoir lieu en dehors des leçons formelles est totalement ignoré. L'enseignant est donc le centre d'intérêt et l'apprentissage, lui et le monde extérieur possèdent toutes les connaissances et l'apprenant obtient ces connaissances grâce aux répétitions et à l'encouragement par les récompenses.

Ce qui manque à cette théorie, c'est l'absence de tout sens de toute prise en compte de la complexité de la psychologie des êtres humains, leurs interactions sociales et leurs relations avec leur environnement. A la fin des années 50, de nouvelles théories d'apprentissage moins formelles ont émergé.

2.1.3 La théorie constructiviste

A l'opposé de la théorie comportementaliste, la théorie constructiviste ne considère pas l'apprenant de l'unique angle des changements de ses comportements. Au lieu de cela, elle se focalise sur le raisonnement et sur la façon de penser de l'apprenant et considère que ce dernier se base d'une façon ou d'une autre sur ce qu'il a précédemment acquis pour comprendre toute nouvelle connaissance. Introduire les facteurs cerveau et raisonnement interne de l'apprenant

complique davantage l'étude de l'apprentissage car cela diminue nécessairement le nombre de paramètres prédictibles. L'apprentissage est donc vu comme un processus d'auto assimilation, d'augmentation et de réorganisation des structures mentales incomplètes [Soloway, 1996]. Les apprenants sont plus pro-actifs et ont un contrôle partiel ou total sur la construction de leurs propres connaissances.

Un autre point important introduit dans [Rogers & Freiberg, 1994], est la prise en compte des réflexions et des retours : l'apprenant est encouragé à retourner ses propres commentaires sur ce qu'il a effectué et pourquoi et comment il l'a fait. Ceci est une partie importante de la théorie non seulement par ce que ça permet aux enseignants de vérifier que les apprenants ont bien assimilé les connaissances transmises, mais aussi, parce que ça encourage les échanges entre apprenants. Ainsi les étudiants rendent visible leur façon d'interpréter et aborder la connaissance qu'ils ont assimilée. De ce fait, cette théorie perçoit les apprenants comme des contributeurs actifs de l'apprentissage qu'ils assimilent en étant capable de refléter leurs propres visions des connaissances perçues, leurs expériences et décisions [Ravitz, 1997]. Cette réflexivité des connaissances peut prendre la forme d'un travail collaboratif dans le cadre de groupes où les étudiants partagent leurs connaissances et compétences. Ils essaient dans ces groupes d'améliorer leurs compréhension à travers des échanges sociaux actifs. Plus le nombre d'apprenants est important, meilleures seront les opportunités de comprendre. Face à un même problème, les étudiants peuvent trouver différentes solutions, et donc s'ils ne comprennent pas une façon de résoudre un problème il peuvent le faire avec une autre qui peut être différente de celle proposée par le tuteur.

2.1.4 Conclusion

Nous pouvons dire que les apprentissages formels et informels représentent les deux bouts d'un même tunnel, la principale différence réside dans la façon avec laquelle l'apprentissage est structuré en fonction du degré de contrôle organisationnel effectué durant le processus dans les deux types. Comparé aux paradigmes constructiviste et comportementaliste, nous pouvons dire que l'apprentissage formel est plus proche de l'approche comportementaliste. En effet, cet apprentissage est très structuré et ne tient compte que des résultats de l'apprenant contrôlés et prévus à l'avance. Par contre, l'apprentissage informel est plus compatible avec l'approche constructiviste car il encourage les interactions sociales entre apprenants (et l'environnement externe d'une façon générale). D'autre part, les résultats ne sont pas contrôlés à l'avance de la même façon que pour l'approche comportementaliste.

L'apprentissage informel a toujours coexisté avec l'apprentissage formel, et continuera de l'être. Le problème majeur avec l'apprentissage informel se situe dans la façon avec laquelle on peut valider les nouvelles connaissances qu'acquièrent les apprenants graçe aux interactions non structurées. Comment exploiter ces interactions de façon positive si elles ne sont pas organisées et structurées ? Si les apprenants interagissent dans le cadre d'organisation, pouvons nous encore parler d'apprentissage informel ? La e-Qualification tente de fournir des réponses à toutes ces questions, et de proposer un modèle qui satisfait tous ces pré-requis. Dans le paragraphe suivant, nous présentons l'apprentissage collaboratif dans lequel nous situons le processus de e-Qualification et la stratégie de réciprocité sur laquelle se base ce processus.

2.2 L'Apprentissage collaboratif

2.2.1 La théorie

Dans ce type d'apprentissage l'idée principale est de mettre l'étudiant au centre du système [Dillenbourg, 1999]. Le système classique où l'apprenant est passif ne peut plus s'appliquer, ce dernier participe activement dans le processus d'apprentissage. De plus, les compétences que l'apprenant est supposé acquérir durant ce processus proviennent, non seulement des interactions avec ses tuteurs, mais en plus de tous les échanges qu'il peut avoir à l'extérieur du cadre du cours [Gokhale, 1995].

En réalité, les étudiants discutent, partagent des connaissances, posent et répondent à des questions tout au long du processus d'apprentissage. Toutes ces interactions doivent être utilisées pour bâtir des connaissances communes. Les apprenants ne sont plus vus d'une façon individuelle où chaque individu travaille seul dans son coin sur un problème privé. Ils sont vus comme une communauté d'apprenants qui partagent un savoir commun et essayent de résoudre leurs problèmes de façon collective. Quand un membre d'une certaine communauté pose une question, plusieurs réponses devraient lui être données par les autres apprenants lui offrant ainsi plus de chance de compréhension.

De nouveaux problèmes émergent avec l'apprentissage collaboratif. Comment partager le savoir d'une même communauté d'apprenants ? Comment et quand ce savoir partagé doit-il être mis à jour. Un autre problème concerne l'absence de membre privilégié dans ce type d'apprentissage. En effet, tous les membres sont égaux et traités de la même façon. En d'autres termes, la communauté d'apprenants doit pourvoir survivre à l'absence d'un ou de plusieurs de ses membres. A l'inverse des autres théories, où l'absence du tuteur ou de l'enseignant signifierait la fin du cours et le blocage du processus. Cela s'explique en partie par le fait que tout apprenant est considéré à la fois comme demandeur potentiel de savoir et en même temps comme étant capable de répondre à des questions donc fournisseur aussi de savoir. D'ailleurs, plusieurs autres noms sont proposés pour ce type d'apprentissage comme le « Peer learning » ou l'apprentissage à pairs.

Nous distinguons dans la littérature trois types de stratégies pour l'apprentissage collaboratif. Le processus d'étude que suivra l'étudiant dépendra de la stratégie adoptée. Les deux premiers types sont les stratégies parallèle et séquentielle. Nous focalisons notre étude sur la troisième stratégie puisqu'elle correspond au mieux à notre compréhension et appréhension de l'apprentissage collaboratif. Nous définissons dans le paragraphe suivant cette stratégie, cela nous permettra de mieux cerner les problèmes qui se posent quant à la qualification de l'apprentissage acquis de façon informelle.

2.2.2 La stratégie de réciprocité

Selon cette stratégie les membres d'une même communauté doivent constamment veiller à ce que leurs activités et connaissances soient en cohérence avec l'ensemble du groupe [Flatcher, 2002]. De ce fait, tous les membres doivent partager un ensemble minimal de connaissances. Afin d'y parvenir, tous les membres doivent travailler conjointement en échangeant des informations et leurs connaissances doivent être en permanence mises à jour. Le terme réciprocité tient ses origines du fait que dans cette stratégie, les membres s'influencent mutuellement grâce aux échanges d'informations et aux mises à jours des connaissances minimales partagées.

Dans notre cas d'étude, où les agents humains sont représentés par des agents artificiels, travailler avec une telle stratégie, nécessite la prise en compte et la résolution de plusieurs problèmes. Premièrement, il faut résoudre le problème de la gestion des groupes d'agents où tous les membres sont considérés comme égaux. Deuxièmement, comment définir les règles d'appartenance à un groupe spécifique et qui permettent d'accepter ou de refuser à un agent d'entrer dans un groupe ? Il faudra ensuite définir pour un groupe donné le savoir minimal partagé, qui quand et comment le mettre à jour. Il faudra troisièmement, assurer la communication entre les différents agents en tenant compte des fréquences d'interactions élevées et de la présence aléatoire des membres. Quatrièmement, il faudra prendre en compte qu'avec l'évolution du niveau d'une communauté (avec des règles pour chaque communauté), certains agents auront des niveaux, soit en dessus soit en dessous du niveau général. De ce fait, ces agents devront quitter le groupe pour respecter la règle de cohérence. Il est donc important de mettre en place des mécanismes assurant la cohérence des groupes et obligeant certains membres à chercher d'autres groupes adaptés à leurs niveaux. Cette contrainte doit être associée à l'absence d'administrateur pour respecter la propriété de parité entre les différents agents du groupe.

2.3 Le processus de e-Qualification et les problèmes rencontrés

2.3.1 Définition

Le processus de e-Qualification est composé de deux phases [Gouardères *et al.* , 2007]. Il faut remarquer qu'à chaque apprenant humain correspond un ou plusieurs agents artificiels qui le représentent dans les groupes d'agents. Dans la première phase, les agents suivent l'activité de l'apprenant humain et essaient d'identifier son évolution par rapport à son niveau avant le début de la session formelle d'apprentissage. Ils détectent les erreurs et les difficultés rencontrées par l'étudiant et essaient de chercher les apprenants qui peuvent l'aider à résoudre ces problèmes. A chaque fois qu'un apprenant aide un autre apprenant sur un problème précis (les agents respectifs des apprenants les mettent en relation), et si cette aide s'avère positive, le premier se voit attribuer des points positifs qui valorisent ses connaissances dans cet axe de l'enseignement. Inversement, si l'aide apportée n'apporte pas le résultat escompté, l'apprenant peut être pénalisé (cela dépend des règles du système).

Dans la deuxième phase, les agents tenteront de mettre l'apprenant dans le groupe le plus compatible avec ses compétences et son degré de savoir, nous parlons de communauté de compétences [Rheingold, 1993]. En réalité, initialement l'instructeur organise les apprenants selon des groupes sur la base de leurs capacités et niveaux (qualifié formellement : notes, moyennes, diplômes ...). A la fin de chaque session de pratique les agents se répartissent en des groupes qui sont plus révélateurs des capacités et connaissances de chaque étudiant, en fonction de ses résultats pratiques et de la qualification informelle réalisée par les agents. Chacune de ces phases pose des problèmes organisationnels différents au niveau de l'agent.

2.3.2 Problématiques

Lors de la première phase les agents doivent trouver des pairs capables de les aider à résoudre les problèmes que rencontre leurs étudiants humains. Cela signifie que les agents doivent organiser et mettre en public les connaissances et compétences de leurs étudiants. D'un autre côté, les agents doivent être munis des outils nécessaires pour effectuer des recherches

ciblées et avoir des résultats rapides quant à leurs requêtes, vu l'évolution en temps réel de l'apprenant. En effet, l'aide apportée à l'apprenant humain doit se faire en même temps que la réalisation de son exercice formel. D'autre part, il faut mettre en place les mécanismes permettant de juger si une aide apportée à un étudiant est valide ou non et appliquer les sanctions ou accorder les récompenses relatives. Si on étudie le problème d'un point de vue client/fournisseur, il s'agit là d'un mécanisme de publication et recherche de services avec validation de la qualité de service. Le service étant l'aide apportée par un apprenant à un autre la demandant sur un problème précis. L'agent aideur possède donc une compétence permettant d'exécuter le service répondant au besoin de l'agent demandeur.

Nous remarquons que lors de la deuxième phase, du point de vue purement agents, des problèmes organisationnels se posent. En effet, aucun agent n'est figé dans une position dans le groupe auquel il appartient. D'une part, l'activité de l'agent est tributaire de l'évolution de l'étudiant qu'il assiste et supervise, d'autre part, elle dépend aussi de l'évolution de l'ensemble des membres de son groupe. La structure du groupe évolue, sa composition varie et ses règles changent en continu. Nous parlons de groupes à frontières fluides. Non seulement les règles d'appartenance à un groupe ne sont pas connues à l'avance, mais en plus, elles varient continuellement. Les agents peuvent découvrir les groupes et demander de les rejoindre. La découverte et publication des services des agents dans chaque groupe sont nécessaires pour garantir leur ouverture à l'extérieur. Nous parlons de groupes ouverts. L'idée d'un seul agent qui gère les accès au groupe, garantie sa cohérence et assure la mise à jour des connaissances communes à tous ses membres, est incompatible avec le principe de parité et d'absence d'administrateur. En plus, il n'y a pas de garantie sur la présence continue des agents qui dépendent de leurs étudiants humains. A tout moment, pour un problème de connexion ou suite à la volonté de l'apprenant, l'agent peut quitter le groupe : lui associer des responsabilités centralisées, serait un handicap pour le fonctionnement général du groupe. La gestion de ces groupes ouverts et dynamiques doit donc se faire donc d'une façon décentralisée. Ceci permettra de garantir la bonne gestion du groupe indépendamment de la présence ou non d'un ou plusieurs de ses membres.

Nous pouvons résumer l'ensemble des caractéristiques et les besoins que nous devons satisfaire pour résoudre les problèmes auxquels nous sommes confrontés dans ce tableau :

TAB. 2.1 – Tableau de répartition des besoins par caractéristique des groupes

Caractéritiques	Besoins
Ouverture	- Accepter de nouveaux membres. - Structurer les services offerts par les membres de chaque groupe. - Pouvoir publier et rechercher les services de chaque groupe.
Dynamique	- Instaurer des règles pour vérifier le respect de la cohérence du groupe. - Instaurer des règles pour gérer les demandes d'appartenance au groupe. - Instaurer des méta règles pour assurer la dynamique des règles.
Gestion distribuée	- Garantir l'égalité de tous les agents du point de vue de la gestion. - Garantir le fonctionnement du groupe en l'absence d'un ou plusieurs membres.

A ces besoins nous pouvons ajouter la nécessité d'un déploiement à grande échelle de notre solution. En effet, le projet e-Qualificaiton fait partie du projet ELeGI [Dimitrakos & Ritrovato, 2004] [Gaeta & Ritrovato, 2004] traitant de l'apprentissage sur l'environnement de grille. Nous tien-

drons compte de cette contrainte pour permettre le dépoilement à grande échelle de notre modèle organisationnel.

Chapitre 3

Les modèles organisationnels des systèmes multi-agents

Un système multi-agents peut être analysé de deux points de vues distincts. Le système peut être étudié du point de vue individuel au niveau de l'agent, on parle d'approche centrée agent. On peut aussi étudier le système multi-agents d'un angle social. Aux débuts des systèmes multi-agents, les recherches se sont concentrées plus sur les architectures où l'agent était le centre des préoccupations. Ce type d'approche reste valable et réalisable si les agents sont homogènes et le système fermé. Face à la complexité croissante des applications traitées dans le cadre des SMA, le besoin de structurer les agents et de les traiter d'un point de vue social s'est fait ressentir. Nous partageons dans ce sens le constat de [Gutknecht, 2001], qui remarque que plus on se situe dans une vue centrée agent, plus on fige le système en émettant des suppositions au niveau de l'interprétation et de la mise en oeuvre de son activité collective.

Bien que les modèles sociaux en systèmes multi-agents soient légion [Gutknecht, 2001], les deux approches ont toujours existé et ne sont pas contradictoires. En effet, pour concevoir un SMA et après avoir mis en place sa structure sociale, il faudra programmer les agents. Pour ce faire, chaque concepteur adoptera un modèle d'agent qui répond le mieux aux besoins de son application. En réalité, pour garantir leur généricité, les approches sociales se déclarent souvent indépendantes de l'architecture interne de l'agent. Nous pouvons donc conclure qu'une étude complète d'un SMA doit se faire sur l'aspect social sans négliger l'architecture interne de l'agent bien qu'en théorie les deux axes soient indépendants.

Nous allons étudier au début de ce chapitre les approches centrées agents en déterminant les axes d'analyse. Nous allons par la suite distinguer les différentes façons d'aborder l'aspect social dans les systèmes multi-agents. Ensuite, nous parcourrons quelques approches ayant adoptées l'aspect social. Nous mettrons l'accent sur les modèles considérant le concept de rôle en précisant à chaque fois les manques de ces modèles que nous essayons de combler dans le notre.

3.1 Les approches individuelles

Les approches centrées agents sont celles qui mettent l'agent au centre du système. Elles étudient l'agent essentiellement selon deux axes distincts : le fonctionnement interne et le comportement externe [Yoo, 1999]. Nous parlons d'architecture d'agents. [Ferber, 1999] va jusqu'à les considérer comme un cas particulier d'organisation. L'axe de fonctionnement in-

terne concerne les caractéristiques de base d'un agent (autonomie, réactivité, pro activité, interprétation, planification, décision). L'axe de comportement externe, étudie la relation avec l'accointance et l'environnement (perception, communication, exécution).

Bien qu'en théorie la plupart de ces architectures ont été conçues dans un esprit de généricité, on remarque qu'elles sont souvent difficilement applicables à tous les types d'agents. En réalité, selon le type de problème auquel on est confronté et les caractéristiques du système cible, il sera préférable (ou possible) d'utiliser une architecture plutôt qu'une autre.

3.1.1 Le recueil des approches individuelles

Nous allons brièvement décrire chacune des principales architectures d'agents les plus connues.

- L'architecture modulaire horizontale : C'est le type d'architecture d'agents le plus répandu. Cette architecture consiste en un assemblage de modules ayant chacun la charge d'une fonction particulière (la perception, la communication, la maintenance de la base des croyances, la gestion des engagements, la gestion des buts, la prise de décision, la planification, etc.). Un type caractéristique de la mise en oeuvre d'une telle architecture est le modèle SRK(Skill-Rule-Knowledge) inspiré du modèle SRK de RASMUSSEN [Rasmussen, 1986] et adapté aux SMA par Chaib Draa dans [Chaib-draa & Levesque, 1996]. Dans cette architecture, il y a deux phases. Une phase d'écoute et de réception des informations de l'extérieur (axe externe). Un module de décision évalue le degré de complexité de l'information et aiguille l'activité de l'agent (axe interne). S'il s'agit d'un signal, une action est immédiatement exécutée par le module action (axe externe). S'il s'agit d'un signe ou symbole on délivre au module planification qui prendra pour chaque cas la décision qui s'impose, planifie et établit l'ensemble des actions à exécuter.
- L'architecture du tableau noir : Cette architecture a connu un franc succès grâce à sa simplicité. Elle est basée sur des modules indépendants qui n'ont pas de relations directes, ils interagissent par partage d'informations. Ces modules sont appelés sources de connaissances (ou knowledge sources ou KS). Les informations partagées par ces sources de connaissances sont stockées dans le tableau noir, une sorte de base de données partagée contenant données finales, hypothèses, résultats intermédiaires... Ce partage d'informations peut générer des conflits : le rôle du troisième composant du modèle, le contrôleur est de gérer ces conflits. La plupart des réalisations à base de tableau noir ont divergé sur la façon de réaliser ce contrôle.
- L'architecture de la subsomption : A l'inverse de l'architecture modulaire horizontale, cette architecture est organisée de manière verticale. Les modules composants une architecture à base de subsomption sont spécialisés dans un comportement spécifique. Les conflits entre les tâches existantes d'un agent subsomptif sont gérés par des règles de priorité et de domination d'une tâche sur une autre.
- L'architecture des tâches compétitives : Contrairement aux architectures précédentes les liens entre les modules de cette architecture ne sont pas fixes. Un module particulier appelé sélecteur, se charge d'élire les tâches à exécuter à un moment donné, en fonction de l'état actuel de l'agent et de son environnement.
- L'architecture du système de production : C'est l'une des architectures les plus connues de l'intelligence artificielle. Elle repose sur trois composants. Une base de faits contenant l'ensemble des connaissances de l'agent. Une base de règles, dont la validité est vérifiée

régulièrement. Un moteur d'inférence qui vérifie si les conditions qui valident une règle sont réunies, l'active et exécute une action donnée

- L'architecture de classificateur : Cette architecture est une variante des systèmes à base de production essentiellement utilisée pour créer des comportements adaptatifs. L'activation d'une règle est suivie d'une évaluation de son efficacité à résoudre un problème ou à atteindre un but. Le classificateur augmente la note des règles qui ont été efficaces et la diminue pour les autres. Les règles les mieux notées sont déclenchées en premier.
- L'architecture connexionniste : Ces modèles sont inspirés du fonctionnement du cerveau humain. L'architecture est à base de neurones. Chaque neurone calcule ses valeurs de sortie et les transmet aux neurones auxquels il est connecté. Les différentes variantes, de cette architecture diffèrent dans la façon de définir les poids des connexions entre neurones.
- L'architecture dynamique : Cette architecture est typiquement utilisée en robotique pour contrôler l'inclinaison des roues et la vitesse d'avancée en fonction de l'état de capteurs. Le comportement d'un agent est décrit par les équations qui relient les valeurs des capteurs à celles des commandes appliquées aux effecteurs.

3.1.2 Les écueils des approches individuelles

A l'issue de la description de l'ensemble des architectures individuelles, nous pouvons dégager deux remarques importantes. La première est que toutes ces architectures n'étudient pas les interactions et les relations qui existent entre les agents du système. L'étude s'arrête au mieux aux frontières externes de l'agent et à la façon d'interpréter les messages ou influences reçus ou les messages à envoyer et actions à effectuer. La deuxième remarque est de supposer que l'agent se trouve dans un monde qui lui est totalement familier.

La deuxième remarque est une interprétation logique de la première. Les modèles centrés agents, parce qu'ils n'étudient le système que du seul point de vue de l'agent doivent faire des suppositions sur l'interprétation de l'activité sociale des agents. En effet, les événements susceptibles de produire des interactions entre agents, sont définis à l'avance. Les messages échangés sont connus et interprétables par tous les agents du système ce qui suppose l'existence d'une ontologie commune partagée. Notons aussi, l'absence d'hétérogénéité, puisque tous les agents sont d'une même architecture. En réalité dans la plupart des cas, ces systèmes sont conçus et développés par le même concepteur, destinés à une utilisation précise et leurs agents sont connus à l'avance. Certes, ces restrictions garantissent une fiabilité et une prévisibilité du système, mais est ce que ça répond aux raisons d'être des systèmes multi-agents ? Une conception basée sur une telle vision, ferme le système dans la mesure où toute entrée d'un nouvel agent est un risque de perturbation qui n'est pas prévu à l'avance et donc insupportable par le système.

La vraie différence entre les systèmes multi-agents et l'intelligence artificielle classique réside justement dans la considération de la sociabilité. Un système multi-agent est par nature ouvert, mais si on n'applique pas ces restrictions imposées par les architectures individuelles le système devient intenable.

3.2 Les approches Sociales

A l'origine du concept d'agents il y a une tentation de reproduire la réalité humaine dans un cadre informatique. Les approches centrées agents ont mis l'agent dans le centre du système en essayant de représenter l'intelligence humaine jusqu'à essayer de reproduire le raisonnement du cerveau humain comme on l'a vu dans l'architecture connexionniste issue de la neurologie. Des modèles d'interaction inspirés des travaux des linguistiques pour analyser les interactions et la communication des agents. Les sciences cognitives ont apporté des réponses aux problèmes comportementaux des agents.

Les approches sociales essaient d'étudier l'agent dans son environnement et ses interactions avec sa société d'agents. Les travaux sur ces approches prennent leurs racines dans la sociologie. Cela s'inscrit dans la même tentation, même inconsciente, de reproduire l'activité humaine dans un système informatique. Avant de faire l'état de l'art des approches organisationnelles, qui nous intéressent dans le cadre de notre thèse, nous allons présenter la vue sociologique de l'organisation et comment les chercheurs du domaine des systèmes multi-agents ont projeté les concepts de la sociologie sur les systèmes multi-agents.

3.2.1 L'organisation d'un point de vue sociologique

Il est très difficile de définir ce qu'est une organisation. Selon la discipline dans lequel elle est étudiée l'organisation peut avoir des définitions différentes voir contradictoires sur certains points. [Plane, 2003] met en évidence les paradoxes et ambiguïtés d'une organisation :

> " *Une organisation apparaît comme une réponse structurée à l'action collective, un ensemble relativement contraignant pour les personnes et, simultanément, comme une construction collective dynamique favorisant l'accomplissement des projets communs.*"

L'organisation est donc l'association de plusieurs concepts et visions contradictoires et concurrentes. D'une part elle est le lieu d'expression de l'individu, d'autre part elle est le théâtre d'une activité collective. Selon ses besoins et sa position, l'observateur de l'organisation peut l'étudier d'une façon globale en ne considérant que l'évolution générale du système. Il peut aussi étudier les interactions des individus de façon individuelle. L'organisation est aussi l'espace de coopération des individus mais un endroit de concurrence entre eux. L'organisation est d'une part contraignante par le biais de ses règles mais favorise la dynamique par les interactions entre les membres. A partir de ce constat et de l'existence de toutes ces contradictions, il est évident que l'étude des organisations a sollicité l'intérêt des chercheurs dans le domaine de la sociologie. Différentes théories essaient de réduire ces difficultés à étudier l'organisation. La théorie de la structuration est une analyse des formes et des dynamiques, proposée par Anthony Giddens [Giddens], elle essaie de réduire cette complexité. Ce dernier propose de reconsidérer la relation entre l'action et la structure ou plus précisément entre l'organisé et l'organisant. Il propose de passer d'une vision de dualisme à une position de dualité. En effet, Giddens a une vision circulaire de l'organisation et estime que d'une part, la structure est constituée des actions des acteurs, et d' autre part, elle est le cadre qui permet cette action. Cette théorie permet de mettre en avant un concept d'organisation qui désigne deux propriétés essentielles de tout système d'activités : le caractère structure (forme) et structurant (processus) de l'action [KECHIDI, 2005].

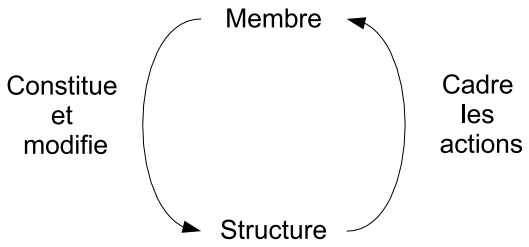

FIG. 3.1 – Vision circulaire de l organisation .

Ce qui est manifeste de cette conception de l'organisation est de ne pas considérer l'agent et son action d'un côté et l'ensemble des agents et leurs cohabitations d'un autre. Ces deux perspectives sont complémentaires et l'étude de l'organisation ne peut se faire qu'en les associant. De telles notions nous portent à conclure qu'étudier l'organisation revient à définir sa structure et l'effet des actions de ses membres sur cette structure. Des concepts comme (groupes, institutions, symboles, rôle, normes, statuts, compétence et dynamiques) sont les éléments de base de ces structures. Pour les systèmes multi-agents, Ferber dans [Ferber, 1999] met l'accent sur cette dualité et la considère comme un facteur accroissant la complexité d'étudier les systèmes multi-agents. Une autre complexité s'ajoute à ce constat c'est celle de positionner la structure organisationnelle. Où se situe réellement la structure d'une organisation ? Y a-t-il une indépendance entre les membres de l'organisation et la structure elle-même ? Comment se projette l'aperçu extérieur du monde sur les croyances et connaissances de l'individu ?

Les réponses à ces questions se trouvent dans les études sociologiques. La constitution des membres et celle des structures, sont donc deux phénomènes liés. Les propriétés structurelles des organisations sociales sont en même temps le moyen et le résultat des actions des membres qu'elles structurent. La transformation de la perception et des croyances (caractéristique ou capacités de chaque chose ou individu), a des implications au niveau structurel utilisées par les individus pour catégoriser les objets eux-mêmes et les autres individus ; elle est expliquée par trois courants différents selon [Mehan, 1982]. Cela dépend du lieu où on situe les structures de perception. Il y a ceux qui situent la structure exclusivement dans la tête de l'observateur et adoptent une attitude constitutive. Pour cette approche, l'individu en tant que membre d'une société s'engage à construire sa propre réalité personnelle en créant de la cohérence à partir du chaos (puisque tout ce qu'il perçoit n'est pas structuré). Supposer que toute la structure est dans les choses perçues du monde extérieur c'est adopter une vision environnementale déterministe. Heman refuse l'affirmation qui situe la structure perceptive soit dans « la tête » soit dans « le monde » et prouve qu'en réalité les structures sont dans l'interaction entre les individus et leur environnement. Il est donc difficile de situer la structure totalement à l'extérieur des membres. En réalité, d'une part elle a une existence concrète et d'autre part elle est observée et mémorisée par chacun des individus et existe en tant que traces mémoires [Rojot, 2000].

Il est plus facile d'étudier et de concevoir des systèmes informatiques en s'appuyant sur des modèles structurés. L'étude de l'activité sociale des agents passe par l'étude des structures qui régissent ces organisations. C'est ce cadre d'analyse qui guidera notre étude de différents modèles sociaux d'agents. Comme pour la sociologie, la représentation de la structure organisationnelle constitue un point de séparation pour les différents modèles organisationnels agents. Comme on le verra dans la suite, certaines approches situent toute la connaissance de la structure sociale dans les agents, alors que d'autres vident les agents de cette connaissance.

L'action de l'individu dans l'organisation est une expression de ses compétences dans un contexte particulier, celui de l'organisation à laquelle il appartient. La projection de la compétence dans le cadre de l'organisation est appelée compétence sociale. Il ne s'agit donc pas des compétences spécifiques des individus mais de la mise en oeuvre et de la coordination de ces compétences dans un contexte particulier. Mais si l'organisation c'est le lieu d'expression de l'individu permettant la mise en oeuvre de l'activité collective, la question qui se pose est celle de fonder une structure par laquelle les compétences des individus puissent s'exercer et se partager. En d'autres termes, comment ajouter les mécanismes nécessaires à la structure organisationnelle pour permettre à ces compétences de s'exécuter. Nous pouvons adopter cette définition de la compétence sociale :

> " *La capacité d'un individu à mettre en oeuvre une série d'habilités (sociales, cognitives, affectives, comportementales), lui permettant d'atteindre des buts sociaux et de s'adapter dans un contexte social* "

[Bowen & al, 2004].

La notion d'acteur compétent développée par A.Giddens [Giddens] est intéressante afin d'appréhender cette dimension de "compétence" dans une structure organisationnelle. Ce dernier postule que « les acteurs sont jugés compétents s'ils sont capables de se construire une représentation de leur environnement d'action et de tenir compte de cette même représentation dans la réalisation de leurs actions ». En plus, de la conscience interne de l'existence de la structure et de sa compréhension, les compétences des individus, génératrices de leurs actions doivent donc être réalisées dans le cadre de la structure de l'organisation où elle a lieu.

Un des problèmes de notre thèse, est de permettre à des agents situés à l'extérieur d'une organisation donnée, de pouvoir profiter le plus rapidement et facilement des compétences sociales proposées par les agents de cette organisation. En effet, quand on n'est pas dans une organisation on ne voit pas son contenu, et quand on ne voit pas ce qu'on ne voit pas, on ne voit même pas qu'on ne voit pas, pourrait-on faciliter la découverte et l'accès aux compétences entre différentes organisations. Dans les systèmes multi-agents actuels la compétence de chaque agent est soit mémorisée uniquement dans la tête de l'agent compétent sans aucun moyen de la découvrir par les autres, soit décrite de façon implicite dans la structure organisationnelle. Dans le premier cas chaque agent possède, de façon explicite (à sa création), la liste des agents proposant les compétences dont il a besoin. Dans le second cas, on passe à l'agent le moyen d'atteindre les compétences dont il a besoin (Agent facilitateur, annuaires de pages blanches...). Dans les deux cas, la solution est soit fermée et rigide, soit contraignante pour l'autonomie de l'agent. Nous utiliserons le concept organisationnel de rôle comme miroir des compétences individuelles des membres d'un groupe d'agents. Le rôle, en plus de sa tâche structurelle, aura la fonction de fédérer et publier les compétences des membres de son groupe.

Considérées dans leur ensemble, les études examinées dans ce paragraphe recommandent de considérer la dualité de la relation entre structure et individu formant l'organisation. Les activités des membres d'une organisation, modifient sa structure malgré qu'ils soient cadrés par cette même structure. En outre, ces études soulignent que situer la structure est un problème qui peut être résolu de trois façons. Soit la structure est totalement dans la tête de l'individu, soit dans le monde extérieur, soit enfin dans l'interaction entre l'individu et le monde extérieur. Enfin, ces études mettent l'accent sur l'importance d'exprimer les compétences des individus d'une organisation dans le cadre structurel de celle-ci.

Avant d'entamer ce parcours des modèles sociaux, et en poursuivant cette analyse du point de vue des systèmes multi-agents, nous constatons que ces recommandations se traduisent ainsi :

- La dualité entre structure statique de description de société abstraite et dynamique du processus d'organisation ou de réorganisation [Foisel, 1998].
- L'origine non définie du processus d'organisation, qui peut être imposé par le concepteur et mémorisé dans la tête des agents ou découvert puis adapté et modifié par les agents eux-mêmes au cours de leurs interactions.
- La nécessité de doter le modèle structurel des moyens de modélisation de la compétence sociale. En l'absence d'une structure générale et d'une vue globale de tout le système, les agents doivent pouvoir découvrir facilement, à travers la structure, ce qu'englobe chaque organisation comme compétences.

3.2.2 Les modèles organisationnels agents

Face à la complexité croissante des systèmes multi-agents déployés dans des environnements instables, dynamiques et ouverts, exigeants une décentralisation de gestion, nous pensons que l'organisation doit être exprimée explicitement. Cela permet la modélisation et la structuration et ce, dès la phase conceptuelle des compétences et des comportements sociaux des agents. D'autre part, dans la phase d'exploitation, cela permet aux agents de raisonner eux-mêmes sur cette organisation afin de l'adapter et de se réorganiser dynamiquement.

Bien que les modèles organisationnels sociaux pour les systèmes multi-agents trouvent leur inspiration dans la sociologie et dans l'organisation humaine, il reste néanmoins évident que la complexité de l'organisation réelle de la société humaine combinée aux réductions de reproduction, concédés au niveau de l'agent par rapport à l'homme, rend impossible la modélisation parfaite d'une telle organisation. Des simplifications et abstractions au niveau des concepts se sont imposées afin de réduire la complexité du monde réel. Les approches organisationnelles sont donc des approches plus structurelles, où l'accent est mis sur l'identification et la conception d'une ossature organisationnelle pour le système. Le but est d'exprimer plus facilement les interactions, dépendances et associations entre agents. L'agent n'est plus étudié isolément mais plutôt vu comme partie prenante d'une organisation qui le guide et dont il doit respecter les normes. Dans un système ouvert une organisation n'a pas nécessairement un objectif global. De même, les agents la constituant, ne partagent pas forcément les mêmes intérêts et peuvent être en concurrence. Etant donné l'organisation et sa structure, tout agent doit se positionner et interagir avec les autres agents tout en étant en cohérence avec ses objectifs et capacités.

On peut donc considérer qu'une organisation est l'association d'agents formant une unité et ayant une activité qui n'est pas nécessairement convergente. L'organisation nécessite donc un minimum de structuration afin que l'ensemble, formé par des membres à priori hétérogènes, soit cohérent [Ferber, 1999]. L'aspect dynamique se manifeste par les relations entre les éléments, c'est l'organisation qu'on qualifie de dynamique. Différents types de structures organisationnelles existent (les groupes, hiérarchique, ad hoc...). La structure organisationnelle est considérée comme statique. Voici les trois axes selon lesquels le système doit être étudié selon [Ferber, 1999] :

- Un axe structurel : Conception d'une ossature organisationnelle au travers des groupes, rôles, interactions...

– Un axe fonctionnel : Identification des objectifs et buts du système et les moyens pour les atteindre.

– Un axe normatif : Déterminer les règles et normes qui régiront l'activité et les interactions des agents au sein de l'organisation.

Même si toute organisation est nécessairement dynamique la majorité des modèles existants sont rigides et la dynamique demeure un problème au centre de la recherche, bien que plusieurs travaux se soient penchés dessus. Avant de faire un survol de la littérature et de présenter quelques modèles organisationnels, traitant de l'ouverture et la dynamique organisationnelle, nous allons faire le point sur la problématique principale qui guide notre démarche.

La problématique dans le cadre des SMA

Nous avons vu que prendre un point de vue organisationnel des systèmes multi-agents pouvait être une clé de simplification de la complexité de ces systèmes. Dans ce contexte, les systèmes multi-agents organisationnels peuvent offrir une solution intéressante répondant à plusieurs besoins des applications actuelles et ressemblant aux problèmes sociaux réels. Ainsi, la dynamique, l'ouverture et la décentralisation deviennent les axes d'études incontournables des systèmes multi-agents organisationnels.

De plus en plus, la spécification et la conception de la structure organisationnelle de ces modèles font intervenir deux concepts importants : les types de rôles et la classification des agents par types de rôles. L'idée principale est que l'analyse de toute application revient à la définition de la structure organisationnelle, exprimée en terme de rôles, en tant que position sociale et medium d'interactions entre agents [Dastani & al, 2004]. Les rôles sont donc devenus, rapidement, l'un des concepts fondamentaux utilisés en même temps comme briques aidant à construire l'ossature de la structure organisationnelle des SMA, comme outils de spécification et de structuration des interactions entre agents, et aussi comme moyen de spécification des actions des agents et d'expression de leurs compétences sociales (dans le sens sociologique précédemment présenté) [Hameurlain, 2004].

Utiliser le concept de rôle dans la modélisation des SMA, permet de séparer les agents du système. En d'autres termes, le concept de rôle permet de faire abstraction du raisonnement et de l'architecture interne des agents. Cette abstraction entre architecture interne des agents et structure organisationnelle est indispensable dans des systèmes ouverts où des agents de différents types peuvent entrer et sortir à tout moment. En effet, toute la difficulté dans les systèmes ouverts et dynamiques est d'associer les rôles aux agents. Comment passer des types de rôles définis au niveau méta aux affectations des agents aux rôles au stade de l'implémentation ? Quels sont les critères pour associer un agent à un rôle (classification) ? Ceci est d'autant plus difficile que dans les systèmes ouverts et dynamiques, contexte de notre travail, le nombre d'agents, leurs types, les frontières du système et ses objectifs sont inconnus à l'avance et peuvent varier en cours d'exécution.

Dans les systèmes fermés, classifier les agents selon des types permettant de jouer ou non un type de rôle, trouve ses raisons dans le fait que dans ce genre de systèmes, les types d'agents existants dans le système, ainsi que les actions qu'ils peuvent entreprendre sont connus d'avance. A partir de ce constat, et dès lors qu'aucun événement externe ne peut perturber la structure établie, il est donc suffisant de définir au préalable des types d'agents. Il ne reste plus au stade de l'implémentation que classer les agents selon ces types pour qu'ils se rattachent directement aux rôles que peuvent leur permettre les classes auxquelles

ils appartiennent. Cela explique pourquoi certaines approches, comme on le verra dans la suite, vont jusqu'à ignorer l'existence de la structure organisationnelle en dehors des agents, adoptant ainsi une attitude constitutive, contestée par Heman, comme nous l'avons expliqué dans l'analyse sociologique de l'organisation. Nous remarquons que de telles approches, en plus d'être fermées, ne permettent pas la dynamique et ne la considèrent pas comme axe d'étude de la structure organisationnelle : l'organisation est figée, la vision circulaire (stipulant que la structure est construite par l'activité des individus et est en même temps leur cadre), défendue par Higgens et reprise par Ferber [Ferber, 1999] et d'autres dans la définition des organisations d'agents, n'est pas respectée.

De telles remarques nous portent à conclure que la notion de rôle semble être un axe clé pour la représentation des structures organisationnelles des systèmes multi-agents ouverts. Afin de mieux situer notre travail, nous proposons un aperçu des différentes approches organisationnelles des systèmes multi-agents utilisant le concept de rôle. Nous expliquons pourquoi ces différentes approches ne répondent pas aux problématiques d'ouverture, dynamique et large déploiement qui nous préoccupent dans le cadre de ce mémoire.

Le recueil des modèles organisationnels à base de rôle

Nous avons vu que pour définir l'organisation des systèmes multi-agents (SMA) le concept de rôle est très largement exploité. Le terme « rôle » peut donc prendre des sens très variés selon le modèle et le contexte dans lequel il est utilisé [Odell *et al.* , 2003]. Plusieurs approches l'exploitent au niveau de la modélisation, d'autres au niveau de l'implémentation. Tantôt le rôle est étudié du point de vue de l'organisation (les agents sont liés au rôles) [Cabri *et al.* , 2004a], tantôt du point de vue de l'agent considéré (dans ce cas) comme une entité à laquelle des rôles sont rattachés [Wai-Sing Loo, 2007]. Certains travaux essaient de trouver une association entre ces différentes appréhensions du concept de rôle et de son utilisation dans les organisations SMA [Odell *et al.* , 2004a].

Dans cette section nous proposons un aperçu des différentes approches organisationnelles des systèmes multi-agents utilisant le concept de rôle. Chacune de ces approches aborde le concept de rôle différemment. Par exemple les deux premières approches focalisent plus sur l'aspect conceptuel, tandis que les autres approches essaient de résoudre les problèmes de faisabilité liés à l'utilisation des rôles dans les SMA tels que la relation entre agents et rôles.

– AGR [Ferber *et al.* , 2003] [Ferber & Gutknecht, 1998] est basé sur l'association de trois concepts clés : l'agent, le groupe et le rôle. Un agent appartient à un ou plusieurs groupes et peut jouer un ou plusieurs rôles dans les groupes auxquels il appartient. Dans AGR un rôle est une représentation abstraite d'une fonction ou d'une position sociale occupée par un agent dans un groupe. Le rôle n'est dans ce cas qu'une simple étiquette qui sert à catégoriser les agents au niveau organisationnel et servant à encadrer leurs interactions et facilitant leurs localisations. Les permissions et autorisations de jouer un rôle ne sont pas exprimées au niveau du modèle bien que présentes à l'implé. Le modèle ne permet pas la description des fonctionnalités du rôle. Le concept de groupe permet d'avoir une séparation claire des modules d'une application et offre une grande clarté organisationnelle. Le groupe n'est qu'une agrégation de rôles. Deux rôles d'un même nom dans deux groupes différents ne sont pas nécessairement identiques. Il n'y a aucune relation (hiérarchie ou inclusion) structurelle entre les rôles d'un même groupe. De même, deux groupes ne peuvent pas être liés structurellement. Les seuls échanges entre groupes se font à travers les agents.

La pauvreté descriptive au niveau des rôles, si elle a l'avantage d'assurer la généricité, oblige néanmoins d'avoir une dépendance assez forte entre le rôle et les agents qui le jouent. En effet, c'est aux développeurs que sera incombée la tâche d'implémenter les comportements, services droits et obligations attendus par le rôle. L'absence de cette description limite le système aux agents conçus par le même développeur ou dans le cadre d'une même application où ces fonctionnalités, droits et obligations sont partagés implicitement. C'est un défaut majeur de AGR justifié, peut être, par l'obsession de généricité exprimée par les auteurs du modèle. D'autre part, la dynamique du modèle se trouve limitée au niveau structurel par l'absence des propriétés que peuvent avoir les rôles et groupes. Ainsi, il est à la charge des développeurs d'exprimer au stade de l'implémentation des contraintes de cardinalité et de comptabilité capables de fournir des moyens pour bâtir une dynamique au niveau de l'organisation. Finalement, aucun moyen, au niveau organisationnel, ne permet la découverte des fonctionnalités des agents (par le biais des rôles qu'ils jouent). Cela signifie qu'il faut décrire explicitement au niveau des agents, les rôles qu'ils sont susceptibles d'utiliser. De telles remarques nous conduisent à conclure que le modèle AGR est victime de son point fort : sa généricité. Sa simplicité et le manque de description au niveau des rôles et groupes, rendent impossible son déploiement pour des applications ouvertes et dynamiques. Notre modèle est bâti à partir de ces constatations.

– GAIA [Wooldridge *et al.* , 2000], est en réalité une méthodologie de conception de SMA basée sur un paradigme organisationnel centré sur le concept de rôle. Elle accompagne le concepteur de la phase d'analyse jusqu'à la réalisation. La méthodologie fait la distinction entre le niveau concret et le niveau abstrait. Le premier niveau concerne la phase d'analyse et permet la spécification du modèle des rôles et des interactions. Le deuxième niveau est celui de la conception, où les modèles d'agents, services et accointances sont définis.
  Un rôle est créé pour réaliser une tâche, pour ce faire, il possède des fonctionnalités nommées responsabilités. Exécuter ces responsabilités, requiert l'utilisation de certaines ressources. Pour réaliser ces responsabilités, on associe des activités aux rôles. Les activités effectuent les calculs nécessaires pour réaliser les responsabilités, ces calculs peuvent être exécutés par l'agent sans l'aide d'un autre agent. Pour utiliser des ressources, les rôles doivent avoir les permissions nécessaires. Les rôles sont liés par des protocoles. Chaque relation entre deux rôles est un protocole qui précise les types de messages échangés, les protagonistes, le contenu ... La phase de conception débute par l'identification des types d'agents et les instances de ces types. Le type d'agent est un ensemble de rôles que peut tenir l'agent. A chaque rôle correspond un ensemble de services. Un service est une fonction de l'agent. A chaque activité spécifiée durant la phase d'analyse, correspond un service. Le dernier modèle spécifié durant la phase de conception est le modèle d'accointance qui défini les liens qui existent entre les types d'agents. C'est un graphe orienté, où les noeuds sont les types d'agents et les arcs sont les chemins de communication.
  GAIA est sans doute la méthodologie SMA la plus complète. Le concept de rôle est le centre du paradigme organisationnel proposé. Les concepts de permission et de responsabilité offrent une plus grande facilité de description du rôle. Le concept de service vu comme un composant du concept de rôle et que peut contenir différents agents réduit la tâche des concepteurs et facilite la réuilisabilité. Toutefois, GIAI ne peut pas être ouvert ou utilisable à grande échelle pour diverses raisons. En premier lieu, la méthodologie fixe à l'avance les types d'agents que peut contenir le système et qui ne changent pas par la

suite. Deuxièmement, les relations entre agents sont statiques et guidées par les rôles. Troisièmement, le concept d'organisation n'a pas de trace réelle, et même les relations entre rôles ne sont pas structurées. Enfin, les permissions, et droits définis pour les rôles concernent des ressources qui sont supposées être connues à l'avance ce qui est inconcevable dans un environnement ouvert. Nous devons cependant faire la différence entre la méthodologie et le modèle organisationnel. GAIA est destiné pour des applications fermées figées où les relations entre les types d'agents et les types de rôles sont fortement couplées.

– [Colman & Han, 2003] étudient le rôle du point de vue organisationnel où il est vu comme « exécutant une fonction dans l'organisation ». L'organisation est donc définie en termes de relations entre les rôles et des règles maintenant la cohérence de ces relations, et répondant aux changements qui peuvent modifier le système. De ce point de vue le rôle est indépendant des agents qui vont le jouer, dès lors qu'ils possèdent les capacités nécessaires pour exécuter les fonctions qui lui sont rattachées. De ce fait, le rôle en tant que composant structurel existe et ce sont les agents qui se rattachent et se détachent de lui selon leurs états et besoins. Le travail définit cinq types de dépendance entre rôle et agent qui le joue. Ces types de dépendance variant de l'absence d'autonomie de l'agent vis-à-vis du rôle à l'autonomie intentionnelle dont le but est la résolution de conflits qui peuvent apparaître dans des organisations d'agents ouvertes. Selon le type d'application et l'environnement où elle sera plongée, l'organisation change de niveau de dépendance. Cette approche a le mérite de se pencher sur le problème des limites qui existent entre le rôle et l'agent qui le joue. D'autre part, les processus et règles, associés à la structure organisationnelle, permettent de répondre à toute perturbation du système. Cela garanti une dynamique structurelle adaptée à des environnement ouverts et dynamiques. Cependant, le modèle se concentre sur la relation qui existe entre le rôle et l'agent qui le joue, et néglige le rôle du rôle comme interface de communication entre agents. Le rôle est vu comme moyen de réaliser des fonctions (rôle serveur de cafés), où l'agent selon le degré d'autonomie (parmi les cinq définis) devra exécuter les fonctions incombées à ce rôle. Le concept de capacité (ou service) que peut offrir un agent joueur d'un rôle aux autres agents du système est complètement négligé. Les interactions entre agents ne sont donc pas structurées et restent à la charge du développeur. Cela empêche une conception d'applications ouvertes où les agents peuvent s'appuyer sur l'organisation pour interagir et s'échanger des services.

– Dans ROMAS [Yan *et al.* , 2003] le rôle est défini selon deux perspectives : conceptuelle où il est « une contrainte sous laquelle l'agent prend part à certaines interactions et évolue d'une certaine façon. Les agents s'exécutent et se comportent dans le cadre des rôles. ». Du point de vue implémentation le rôle est : « une encapsulation de certains attributs et comportements de l'agent auquel il est lié ». Le rôle est vu comme le miroir de l'agent permettant d'indiquer aux autres agents le moyen d'interagir avec lui. L'agent est défini avec un ensemble d'attributs, de règles et de capacités. Tout agent selon son état mental et sa situation peut jouer un ou plusieurs rôles. Des relations d'héritage et d'inclusion existent entre les rôles. L'organisation est totalement définie avec les relations entre les rôles qui sont structurés de façon ad hoc.
L'approche prétend que les rôles existent durant toutes les phases de l'analyse jusqu'au développement. RoMAS génère les rôles dès la phase d'analyse et plus précisément des cas d'utilisations. Les agents peuvent même être générés grâce aux descriptions des rôles en fonction des capacités.

Les rôles dans RoMAS sont définis dès la phase d'analyse, les relations entre ces rôles sont figées. L'ajout et la modification d'un rôle ou la modification des liens entre rôles ne sont pas possibles. L'organisation n'existe que dans les relations entre les rôles qui ne sont pas rassemblés en groupes. Dans un système ouvert cela accentue la complexité de modélisation.

– [Odell *et al.* , 2004a] propose l'utilisation de classificateurs, pour l'agent qui permet de catégoriser les agents et les classifier selon différentes façons. Il distingue classificateur physique et classificateur des rôles. Le classificateur physique définit les agents selon les primitives, besoins, capacités et attributs qu'ils possèdent indépendamment des rôles qu'ils jouent où peuvent jouer. Certains aspects sont partagés par tous les agents (nom, adresse) et d'autres non (l'envoie ou réception de messages). Le classificateur par rôles défini pour un agent quels types de rôles il peut jouer à un instant donné. Un classificateur de rôle peut appartenir à plusieurs groupes en même temps. Les deux classificateurs peuvent être associés pour spécifier quels classificateurs de rôles sont permis pour un classificateur physique donné. Après, c'est selon les capacités individuelles de chaque agent, dans le classificateur physique, il pourra oui ou non jouer un des rôles spécifiés par l'association. Ainsi, pour jouer un rôle, un agent doit appartenir à un classificateur physique en association avec un classificateur de rôle, et qu'il doit avoir les capacités requises pour jouer ce rôle. Un agent peut changer de rôle suite à un changement dans ses besoins ou ses capacités.
  L'approche tient compte de la séparation de la relation associative agent-rôle et agent affecté à un rôle. Les instances de la classe AgentRoleAssignment associent un agent à un rôle dans un groupe. La relation entre un agent, rôle et groupe est une relation ternaire qui associe un agent à un rôle dans un groupe donné. L'instance d'une classe AgentRoleAssignment ne contient aucune information supplémentaire.
  Le modèle tente de résoudre le problème d'organisation dynamique des agents selon leurs besoins et capacités pendant leurs existences. Le problème majeur de cette approche est la rigidité des structures organisationnelles. En effet, si les agents peuvent changer de rôles selon leurs états et besoins, les rôles eux sont fixes et les conditions de jouer un rôle, ainsi que le comportement attendu d'un agent jouant le rôle sont spécifiés à l'avance et ne peuvent plus changer. L'entité AgentRoleAssignment reste vide de toute propriété pouvant contenir des données et informations concernant l'agent en tant que joueur du rôle comme nous le proposerons dans notre solution.

– MOISE/MOISE+, MOISE est un modèle organisationnel normatif où l'action individuelle des agents au sein de l'organisation est restreinte par les rôles qu'ils jouent. Les interactions et échanges entre agents sont structurés par les liens organisationnels entre les rôles. Un rôle est un ensemble de missions. Une mission est définie par des combinaisons et autorisations sur les buts et les actions. Une mission spécifie un comportement attendu d'un agent. Le modèle est destinée aux agents BDI ce qui explique l'utilisation des concepts de buts, plans, actions et ressources. Le plan est défini par rapport à un but à atteindre. Un but peut être décomposé en sous buts et actions. Les actions sont associées à des ressources.
  MOISE/MOSIE+ distingue structure organisationnelle et entité organisationnelle (organisation effective). La structure est formée par les groupes et les liens entre les rôles de ces groupes. L'organisation effective est associée à une structure. Elle est formée par les groupes, les associations entre agents et rôles et les liens entre agents. Le groupe est alors défini par ses rôles et les missions qui leurs sont liées.

MOISE+ enrichit l'aspect structurel du modèle original MOISE avec les propriétés d'héritage et inclusion entre les rôles. Le modèle introduit aussi des contraintes de comptabilité et de cardinalité pour exprimer une meilleure dynamique structurelle.
Malgré les enrichissements au niveau structurel, permettant une meilleure souplesse d'expression de l'organisation, qu'a apporté MOISE+, le modèle reste difficilement utilisable dans un environnement ouvert. En effet, en cas d'évènement imprévu les agents n'ont aucun moyen permettant de résoudre les conflits possibles. Cela est d'autant plus problématique que ces derniers peuvent jouer plusieurs rôles. D'autre part le modèle est orienté vers une vision BDI et tous les concepts structuraux servent cette cause. Cette dépendance entre modèle organisationnel et architecture d'agents limite le modèle à des types d'agents particuliers.

– YAMAM [Saval, 2003]est un modèle basé sur quatre concepts importants : Agent, Rôle, compétence et tâche. Un agent joue un ou plusieurs rôles. Un rôle peut être un service, une fonction ou une position identifiant l'agent. Jouer un rôle revient à exécuter une ou plusieurs tâches. Une tâche est une action élémentaire qui nécessite des compétences. Ainsi, jouer un rôle requière la possession de certaines compétences qu'il ne les possède pas, l'agent va chercher à les acquérir. Les compétences s'échangent entre agents. Le concept de groupe n'a pas de modélisation concrète dans le modèle. Les agents n'ont aucune vision des groupes auxquels ils appartiennent. En réalité, c'est le noyau qui décide de regrouper les agents selon des critères déterminés. Si le noyau (représenté par un agent particulier), juge que des agents partagent des caractéristiques, des rôles ou des buts en commun, il les regroupe. Pour échanger des compétences, les agents demandent à au noyau de le faire. C'est là que la notion de groupe, pour les concepteurs de Yamam, tire son intérêt :

  "*On va plus facilement demander des services à des agents possédant des rôles similaires puisque ceux-ci possèdent des compétences proches des nôtres. L'intérêt des accointances est de minimiser la fonction énergie pour la recherche d'agents travaillant dans le même domaine.*"

  L'agent gère son réseau d'accointance dans un tableau dynamique contenant les adresses des agents voisins ou plutôt lui « ressemblant ». Pour Yamam

  " *Un agent va ajouter un autre agent dans son carnet d'adresse lorsqu'il possède des choses en commun avec lui.*"

  Si nous accordons un intérêt particulier au concept de compétence et au rôle comme moyen d'exécuter des tâches, nous émettons certaines réserves sur certains aspects du modèle Yamam, et notamment la vision d'accointance et de recherche de compétences qui en découle. Nous considérons même qu'elle est très réductrice de la complexité réelle, voir erroné dans la plupart des cas. Si un agent demande des services c'est qu'il ne peut les possède pas et que normalement ses semblables non plus. Regrouper les agents par centres d'intérêts, rôles joués ou compétences et une façon de classifier les agents par type de domaine ou par type de compétences fournis mais surtout pas recherchés. Cela peut être un atout pour gérer la décentralisation comme nous le proposons. En effet, chaque agent d'un même type (classifié par compétences ou rôles) peut répondre à une requête d'un agent n'appartenant pas à ce groupement d'agent. A partir de là, l'absence d'un ou plusieurs de ces agents ne cause pas un blocage du système. D'autre part, laisser à l'agent la charge de créer son propre réseau d'accointance à partir du néant, nous semble être un déplacement du problème vers le bas, vers le niveau d'implémentation. En plus, créer le réseau d'accointance ainsi défini, est une tâche lourde en temps et difficile à définir, les critères d'accointance étant assez vagues et peu précis.

Pour conclure, nous pouvons dire que nous partageons l'association de concept de compétence et de rôle ainsi que l'idée d'échange de compétences. Cependant, la suppression de la notion de groupe au niveau du modèle, et la garder uniquement au niveau du noyau, constitue une perte d'un moyen de structuration important pour des systèmes ouverts. Les réseaux d'accointances définis par le modèle, sont une façon de représenter des groupes dans la tête des agents. Cette façon de présenter les structures est totalement irréalisable dans un système ouvert, le contrôle est le maintien de la structure devient une tâche très difficile.

– RIO (Rôles, interactions et organisations) : RIO est un modèle organisationnel à structure hiérarchique facilitant l'échange de compétences entre agents et implémenté sur la plateforme MAGIQUE. Une compétence est un ensemble de fonctionnalités. Les agents sont initialement considérés comme coquilles vides capables uniquement d'acquérir de nouvelles compétences et de communiquer. Les compétences sont les prérequis pour jouer des rôles. Le mécanisme d'acquisition de compétences, appelé apprentissage dans le modèle (par référence à l'apprentissage humain), se fait soit à la naissance de l'agent par son développeur, soit apprises en cours de vie par d'autres agents. L'apprentissage se fait par communication entre agents. La structure d'une organisation dans RIO étant hiérarchique, la communication est donc essentiellement verticale. Une hiérarchie dans RIO, est un arbre dont les feuilles sont des spécialistes et les autres noeuds sont les superviseurs et ont la responsabilité de gérer les agents qui sont à leurs charges. Afin d'offrir plus de souplesse au modèle, les liens direct entre agents sont possibles, passant la structure d'une hiérarchie à un graphe. Le point intéressant de ce modèle est la dynamique assurée surtout en ce qui concerne la recherche et la délégation de tâches. En effet, si un agent a besoin d'une compétence pour exécuter une tâche, il vérifie s'il ne la possède pas auquel cas il essaie de chercher chez d'autres agents. La recherche se fait de deux façons, soit l'agent connaît une accointance possédant la compétence recherchée, sinon il délègue la recherche à son superviseur. L'idée de base derrière cette organisation est de ne pas figer les relations entre demandeurs et compétence et agents possesseurs de cette compétence. A une requête donnée, plusieurs agents pourront répondre. Les superviseurs cherchent à chaque fois un agent fournisseur de la compétence, cela permet une dynamique et offre plus de fiabilité au système. Les destinataires des requêtes ne seront donc pas nommés à l'avance et l'exécuteur peut changer d'une demande à une autre. Ainsi, l'absence la saturation ou le refus d'un agent lors de son invocation, pour un transfert de compétence, ne sera pas préjudiciable ni bloquant pour le système.
L'idée de voir l'agent comme étant une encapsulation de compétences est une contrainte forte sur l'architecture des agents, imposée par le modèle RIO. Exprimer les conditions d'acquisition des rôles n'est pas suffisant, des contraintes sur les attributs peuvent être des conditions pour jouer un rôle. Mais notre critique principale de ce modèle porte sur la façon de rechercher les compétences. Comme pour Yamam, RIO délègue cette tâche aux agents qui doivent faire leurs réseaux d'accointance. « ...les agents doivent disposer d'un moyen pour trouver le "bon agent". C'est pourquoi un mécanisme de recherche de routage par défaut des messages et une structure d'accointances par défaut doivent être fournis pour permettre d'atteindre le "bon agent", au moins par des intermédiaires. » [Routier hdr 2003]. Tout dans RIO s'exprime en terme de compétences, mais les rôles et groupes ne sont utilisés que pour positionner les agents. Les compétences ne sont utilisées pour les rôles et groupes que comme limites d'octroi de rôles ou d'entrée dans des groupes. Nous pensons que le concept de compétence, peut être placé à un autre niveau par rapport à la structure. En effet, l'échange de compétences ne doit pas se

faire librement entre agents mais dans le cadre des rôles dans un groupe donné. Cela permettrai une meilleure structuration des agents et des compétences et donc une plus grande rapidité de recherche de compétences. Finalement, l'échange de compétences se fait de façon bilatérale et sans contrainte entre deux agents, ce qui peut produire des agents très lourds et un système ingérable dans la mesure où tous les agents chercheront à avoir le maximum de compétences.

– MOCA, le modèle MOCA [Amiguet, 2003] [Amiguet & Baez, 2004]est une approche organisationnelle qui combine le concept d'agents et de composants pour assurer la dynamique des structures sociales. Ce modèle agent componentiel se base sur les concepts de groupes, rôles et compétences. Un groupe est l'instanciation d'une organisation d'agents jouant des rôles. Un rôle dans MOCA, est un composant qui s'instancie de l'état d'un rôle décrit au niveau de l'organisation. Deux rôles de même nom (instanciant une même description de rôle), situés dans deux groupes, instanciant une même organisation, auront les mêmes comportements. Deux agents ne peuvent communiquer que si au moins ; deux des rôles qu'ils jouent sont en relation. L'agent n'est autonome que dans le choix des rôles qu'il joue ou qu'il lâche. Les compétences sont dans MOCA les bornes d'entrée et de sortie des composants. Pour un rôle, les bornes sont les compétences requises et fournies. Pour jouer un rôle un agent doit posséder les compétences nécessaires. Les rôles s'échangent des influences. L'agent gère la communication entre ses composants et la réception des influences émises des rôles des autres agents, qu'il dispatche à ses rôles. L'agent doit aussi contrôler l'exécution des composants et éviter les conflits.
  Comme pour RIO, les agents dans MOCA sont vus comme des coquilles vides qu'on enrichi avec des composants. En plus d'être des prérequis pour jouer un rôle, le concept de compétence est utilisé pour décrire le comportement des rôles. Nous partageons cette utilisation du concept de compétence comme entrée et sortie des rôles, cependant nous émettons plusieurs critiques sur l'utilisation des rôles et leur représentation dans le système. Dans MOCA, c'est aux rôles d'accepter ou refuser l'exécution d'une compétence demandée par un agent externe. L'autonomie de l'agent n'a plus de sens et est complètement déplacée au niveau des rôles. Ces derniers prennent l'allure d'agents eux même avec des capacités de raisonnement, de communication, l'agent se transforme plus en environnement transmettant les influences et gérant les conflits. D'autre part, nous ne partageons pas l'idée d'insérer les rôles dans l'architecture des agents, cela est un handicape pour l'ouverture du système multi-agents. Le modèle ainsi conçu impose une architecture interne des agents, la structure organisationnelle et le modèle d'agents sont dépendants. Enfin, MOCA ne propose pas des mécanismes permettant la gestion distribuée du système, l'utilisation d'un rôle page jaune joué par un agent n'est qu'une solution centralisée et inefficace pour des systèmes largement déployés.

– BRAIN (RoleSystem/ RoleX), dans le cadre du framework BRAIN [Cabri *et al.* , 2003a] Cabri and al définissent le rôle comme un ensemble de capacités sous forme d'actions, que l'agent peut exécuter grâce à son rôle, et de comportements prévus sous forme d'évènements que l'agent est sensé gérer afin d'agir conformément à ce qui est spécifié par le rôle. Les interactions entre agents sont complètement définies par rapport à un système d'interactions entre rôles. Ces interactions sont formées par le couple (actA, evB) où actA est une des actions offertes par un rôle A que joue l'agent initiateur de l'interaction et où evB est un des évènements supportés par un rôle B joué par l'agent destinataire de l'interaction. La translation entre action et évènement est gérée par le système d'interaction qui sauvegarde les différentes connexions possibles entre les rôles.

L'action représente l'aspect proactif de l'agent, alors que l'évènement en est le côté réactif.
Deux approches d'infrastructures d'interactions : RoleSystem [Cabri *et al.* , 2003b] et RoleX [Cabri *et al.* , 2004b] sont implémentés pour mettre en oeuvre la définition de rôle précédemment énoncée.

- RoleSystem est une approche centralisée où la relation s'effectue entre un rôle et ses agents de façon directe à travers une simple connexion. Le système gère de façon centralisée, l'obtention et libération des rôles par les agents à travers un système de certificat garantissant la validité des conditions d'obtention d'un rôle donné. Le certificat est aussi un moyen de connexion entre l'agent et le rôle qu'il joue. L'agent exécute les actions offertes par le rôle grâce à son certificat. L'activité de l'agent se limite à l'exécution de l'action et la réception des évènements que renvoie le rôle.
- RoleX est destiné aux applications à large échelle de déploiement et comble le défaut de centralisation de RoleSystem. L'agent et le rôle qu'il désire jouer sont fusionnés. En fait, l'agent est recréé de nouveau avec la fusion de son code source avec celui du rôle qu'il vient de jouer. L'opération inverse est effectuée quand l'agent lâche un rôle. L'approche se base sur XML pour décrire les rôles, les événements et les actions. Ces descriptions offrent une meilleure visibilité des agents pour choisir le rôle convenant au mieux à ses besoins.
  Les deux approches de BRAIN, RoleSystem et RoleX sont les plus proches de la vision que nous nous faisons de la fonction du rôle dans les organisations. Notre intérêt porte surtout sur le moyen d'exploiter au mieux le concept de rôle en utilisant la description et en associant des services aux rôles (actions). Mais la grande nouveauté de cette approche est la prise en compte de l'aspect « utilisation » du rôle. Le rôle se trouve l'interface entre agent exécutant une action et agent la requérrant. Les communications entre agents sont explicitement représentées par les rôles et définies en terme d'actions à exécuter et évènements à traiter.
  Cependant, l'approche ne fait pas la séparation entre rôle joué et agent jouant le rôle. Les propriétés liées à cette relation sont soit dans l'agent (RoleX) soit dans le rôle (RoleSystem). RoleSystem est centralisée et ne répond pas au besoin de décentralisation exigée par un environnement ouvert. D'autre part, fusionner agent et rôle dans (RoleX)[Cabri *et al.* , 2005], en plus d'être conceptuellement anormale, est très difficilement réalisable dans des vrais systèmes dynamiques. En effet, la fusion de code est loin d'être une opération banale, surtout si cette opération s'effectue assez souvent. Des problèmes se posent tant au niveau de la consommation de ressources que celui de la stabilité, la cohérence, sûreté et sécurité du système.

La vision : utilisateur d'un rôle et joueur d'un rôle est, à notre avis, mal exprimé dans le modèle. En effet, le couple (demandeur/fournisseur) est situé à un niveau structurel plus haut que sa position dans la réalité. Un agent qui demande un service, se trouve obligé de jouer un rôle permettant d'établir le couple (action, évènement) où évènement permet la réalisation de l'action par un agent et donc l'exécution du service. La décomposition de la relation (demandeur/fournisseur) sur deux rôles est inutile et accentue la complexité du système.

Nous proposerons, comme le montre la figure **??** ci-dessous, de simplifier le chemin de l'initiateur de l'interaction vers le récepteur en évitant de la placer au niveau des rôles. La notion de rôle client (déclanchant une action) et rôle serveur (recevant un évènement) est remplacée par une position d'agent utilisateur du rôle (fournisseur de l'action). Le rôle est joué par l'agent exécutant l'action. Nous ne parlerons plus d'action et évènement mais plutôt de service et requête.

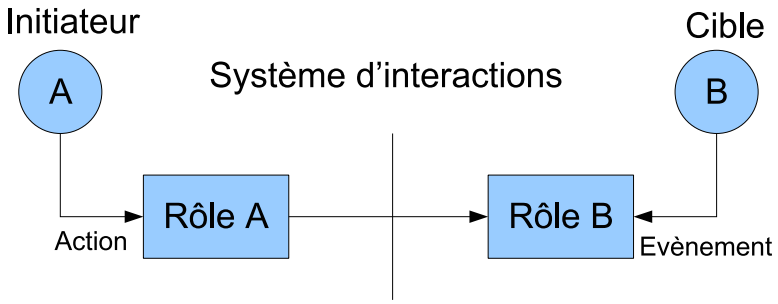

FIG. 3.2 – L'utilisation des rôles selon BRAIN.

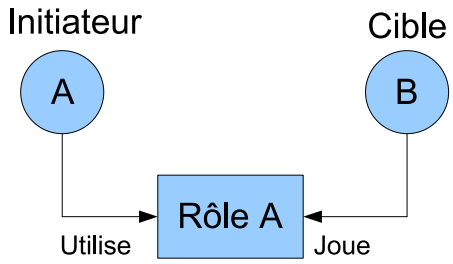

FIG. 3.3 – L'utilisation des rôles.

Conclusion

Compte tenu des éléments qui viennent d'être retracés, nous pouvons replacer cette thèse dans son contexte, pour à la fois dégager les axes essentiels de chaque modèle, relever les points forts et faibles par rapport au contexte des systèmes ouverts et dynamiques et rendre justice aux travaux qui nous ont inspirés pour la conception de notre approche.

Mais avant de passer à la présentation de notre modèle qui sera l'objet du prochain chapitre, nous désirons identifier les axes de base que nous jugeons nécessaires pour l'ouverture et la dynamique de toute organisation. Certains de ces axes ont été utilisés dans les approches étudiées, d'autres ajoutés par notre contribution.

– Dynamique organisationnelle : Un agent doit pouvoir jouer et quitter un rôle (appartenir ou quitter un groupe) selon ses besoins et de façon dynamique. Certaines approches (Gaia) figent les relations agents-rôles et suppriment ainsi toute dynamique. D'autres comme (Odell and al), sont moins rigides et figent les classes d'agents avec des classes de rôles mais permettent à un agent la liberté de choisir les rôles à jouer parmi ceux permis par la classe physique à laquelle il appartient. Il est important, pour une utilisation dans un système ouvert et dynamique, de laisser à l'agent le libre choix, sans aucune contrainte organisationnelle, de jouer un rôle. Ensuite, il faut situer au niveau du rôle les contraintes filtrant les agents qui peuvent être acceptés. Pour ce faire, l'ajout de propriétés de cardinalité et comptabilité au niveau du rôle, comme pour MOISE+, reste, sans doute, la meilleure solution permettant la dynamique organisationnelle.

– Dynamique structurelle : L'organisation doit avoir des mécanismes lui permettant de changer de fonctionnement quand cela est nécessaire. Nous parlons d'organisation à structures évolutives. L'expression des concepts de la structure doit permettre l'application de règles qui, en fonction de certains paramètres, modifient la structure de l'organi-

sation en ajoutant supprimant ou modifiant certains rôles ou groupes. Nous proposons l'utilisation de règles au niveau des rôles et des groupes qui modifient leurs structures et compositions.

- Attribution de comportements aux rôles (explicitement) : Un rôle dans la plupart des approches organisationnelles décrit un comportement attendu par l'agent qui le joue. Afin de laisser à l'agent le libre choix de jouer un rôle, il doit savoir pourquoi il joue un rôle et interpréter quel comportement on attend de lui en jouant le rôle. Faire le lien entre ses besoins et les comportements des rôles lui permettent de décider de façon délibérée quel rôle jouer à un instant donné. Certaines approches expriment les comportements en terme de capacités (RoMAS, Odell and al), d'autres en actions (BRAIN), compétences (RIO, YAMAM, MOCA) ou services (GAIA). BRAIN va jusqu'à décrire les rôles en terme de description XML des capacités qu'ils offrent, facilitant, pour l'agent, la recherche des rôles à jouer. Dans un système ouvert, la description des rôles en terme de comportement s'avère nécessaire.
- Le rôle comme interface d'interaction (jouer/utiliser) : Le rôle est un pattern d'interaction, il doit par conséquent faciliter la mise en relation de deux agents pour collaborer, communiquer ou simplement s'échanger des informations. Toutes les approches étudiées, ne prennent pas en compte l'aspect utilisation du rôle de façon concrète. Le rôle est considéré uniquement du côté de l'agent joueur. La relation entre deux agents passe par la relation entre deux rôles qu'ils jouent. Nous proposons de considérer le rôle comme interface entre agent joueur exécutant le comportement décrit par le rôle et agent utilisateur qui interagit avec un agent joueur pour exécuter un comportement.

Nous allons voir dans le chapitre suivant une proposition de modèle organisationnel multi-agents ouvert et dynamique, basé sur les principes que nous venons de dégager.

Chapitre 4

Le modèle AGRS

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'ouverture et la dynamique des groupes d'agents. Nous sommes bien toujours dans le cadre d'applications hétérogènes déployés dans des systèmes vastes.

Nous proposons donc un modèle organisationnel par rapport à ces besoins et en respectant deux contraintes fortes :

Abstraction applicative : Le modèle organisationnel doit être conçu sans préjugés sur le domaine applicatif dans lequel il sera utilisé. L'objectif qu'il faut garder à l'esprit est de garantir l'indépendance par rapport aux applications.

Contrainte d'opérationnalité : Notre modèle doit être d'une part, assez indépendant et abstrait par rapport aux applications mais d'autre part, il doit être utilisable avec n'importe quel type d'architecture d'agents.

Ce dernier aspect opératoire nous conduit à considérer le point commun entre l'organisation et les agents quelque soit leurs types et architectures internes. Le point d'intersection entre agent et organisation se situe dans les comportements. L'agent est autonome et proactif, mais agit pour répondre à des besoins internes et se positionne dans l'organisation en fonction de ses besoins et selon ses capacités. Pour garantir une indépendance par rapport à l'architecture des agents, le modèle doit s'exprimer en fonction des comportements attendus ou réalisés par les agents. L'étude de l'état de l'art, réalisée au précédent chapitre, a fait paraître des concepts comme service, capacité, action ou compétences pour exprimer les comportements des agents. Indépendamment de la terminologie utilisée, tous ces concepts sont plus au moins identiques et expriment les capacités de l'agent et ce qu'il est sensé pouvoir faire dans son environnement ou offrir aux autres agents. Nous adopterons le terme service pour exprimer ce concept dans notre modèle. Le rôle sera exprimé en fonction des services que peut offrir ou requérir un agent dans l'organisation.

Nous allons dans ce chapitre nous attacher à présenter le modèle que nous proposons, en expliquons d'abord comment il répond aux besoins évoqués, puis en énonçant les concepts primitifs et les relations (statiques ou dynamiques) qui les structurent. Pour ce faire, on étudiera un exemple concret de la vie quotidienne, dans un environnement ouvert et dynamique pour mieux matérialiser les besoins identifiés et justifier les concepts et propriétés que nous énoncerons dans notre proposition. Nous examinerons ensuite quelques conséquences et propriétés. Nous verrons alors quelques expressions possibles d'organisations multi-agents. Nous terminerons en montrant comment la flexibilité de ce modèle permet de l'utiliser facilement

dans d'autres domaines où la dynamique l'ouverture et les autres propriétés des systèmes multi-agents, permettent de résoudre un très grand nombre de problèmes.

4.1 Proposition

Dans ce travail, notre préoccupation principale est de proposer un modèle et une implémentation pour permettre de prendre en compte l'ouverture et la dynamique des systèmes. Avant de présenter notre modèle, il est judicieux de définir les axes principaux de notre thèse : Ouverture, dynamique.

SMA ouverts : Un SMA est dit ouvert s'il n'a pas de frontières définies, cela signifie que des agents, de différents types, entrent et sortent de temps en temps durant le cycle de vie du système. Cela pose un problème aux concepteurs dans la mesure où ils ne peuvent pas définir à la phase de conception les types d'agents qui peuvent faire partie d'un groupe donné [Xinjun & Yu, 2004].

SMA dynamiques : Un SMA est qualifié de dynamique quand ses constituants, en particulier les relations entre agents et les services que propose et requiert chacun d'eux, peuvent changer après le lancement du système. Le système évolue avec l'évolution des agents et structures le composant. Ainsi les rôles que peut jouer un agent changent, et donc ses relations avec les autres agents du système. Comme nous le verrons dans la suite nous distinguons dynamique de l'organisation et dynamique de sa structure.

En parlant d'ouverture et dynamique de systèmes mutli-agents, on fait implicitement illusion à l'ouverture et dynamique des organisations et particulièrement des groupes d'agents. Les interrogations du concepteur en découlent : Que signifie ouverture d'un groupe ? Qu'est ce qu'un groupe d'agents dynamique ? Comment mettre en place les mécanismes assurant ouverture et dynamique des groupes ? Comment les agents doivent s'adapter à de tels groupes ?

Dans les systèmes classiques, le concepteur peut prétendre avoir toutes les latitudes pour maîtriser les types d'agents de son système ainsi que les interactions et l'évolution des groupes d'agents. Mais cette position est irréalisable dans un environnement ouvert, les types d'agents qui peuvent entrer et sortir des groupes, les interactions possibles entre les agents ne sont pas prévisibles. Sans des mécanismes pour maîtriser ces variantes et cadrer l'évolution du système en fonction des besoins et capacités des différents agents qui y entrent, on tombe dans l'anarchie et le système multi-agents ne peut plus fonctionner

Nous souhaitons donc nous situer à un niveau d'abstraction assez haut permettant d'exprimer les structures sociales indépendamment des systèmes et leurs agents. Toutefois, la contrainte d'opérationnalité, que nous nous sommes fixés, nous oblige à proposer une solution facilement implantable, indépendamment des architectures d'agents présentes et susceptible de prendre part au système. Notre proposition tente donc de réduire au maximum l'intervention du concepteur au niveau de la modélisation des organisations d'agents, et plus particulièrement celles déployées à grande échelle pour les applications ouvertes et dynamiques.

Nous sommes donc obligés de manipuler au niveau structurel des concepts partagés par tous les agents. Le concept de service est choisi comme point de raccord entre agents et rôles. La structure organisationnelle sera exprimée en fonction de ce concept qui décrit les fonctionnalités qu'un agent désire mettre au service de l'organisation à laquelle il appartient (c'est le concept de compétence sociale étudiée dans le chapitre précédent). Ces services seront rattachés aux rôles et les agents désirant offrir ces services le feront à travers les rôles. Le

concept de rôle doit être donc enrichi en décrivant, dès la phase de conception, les services que doivent offrir ses agents membres (joueurs du rôle). Cela permettra d'étudier le rôle du point de vue de l'agent qui le joue et de l'agent qui l'utilise. En effet, le premier saura quels services il devra offrir, le second quels services il pourra demander. Nous avons constaté, que dans tous les modèles agents orientés rôles on étudie le rôle soit du point de vue de l'agent qui le joue (centré agent) soit du point de vue de l'organisation (centré organisation) :

Vue centrée agent : Dans cette vision l'agent est vu comme une entité stable à laquelle des rôles sont temporairement rattachés [Cabri *et al.* , 2004a].

Vue centrée Organisation : Dans cette deuxième vision le rôle est l'entité stable et les agents lui sont rattachés [Colman & Han, 2003] [Odell *et al.* , 2003].

Même si on associe les deux visions [Odell *et al.* , 2004b], on omet de voir les rôles du coté des agents qui les sollicitent pour interagir avec leurs membres. Prenons par exemple le cas d'un restaurant. : Dans une telle organisation il y a un rôle serveur (dans un restaurant), il y a les agents serveurs (qui jouent le rôle serveur) et des agents qui interagissent avec ce rôle. Ces agents peuvent être les clients mais aussi les cuisiniers les caissiers etc. Ces agents interagissent avec un agent serveur pour une raison bien précise. Ce qu'ils cherchent c'est un service que tous les serveurs sont capables d'offrir du fait qu'ils sont justement des agents qui jouent le rôle de serveur. En décrivant les rôles à un niveau conceptuel, il est possible de décrire ce qu'on attend d'un agent qui joue ce rôle. Cela revient à définir le comportement des agents qui vont jouer le rôle. Le serveur, doit être capable de répondre à une des demandes : « apporter les menu », « prendre une commande », « apporter les plats », « apporter la facture », « débarrasser la table ». Chacun de ces services peut être exécuté en cas de besoin par tout serveur présent à l'instant t dans le restaurant.

Principe 1 : *Le rôle décrit les services que tous les agents, qui jouent ce rôle, sont capables de fournir. Ces services sont découverts (ou connus) puis exploités par les agents utilisateurs du rôle.*

Si un client a besoin du menu il saura à qui s'adresser : il appellera un serveur. En réalité, le client sait implicitement que ce sont les serveurs qui rapportent les menus. Mais s'il l'ignore il peut lancer un appel : « apporter le menu », l'agent client désire récupérer le menu. Un des serveurs (normalement le plus proche du client et disponible), se chargera de livrer le menu au client. Ce qui s'est passé c'est que l'agent serveur a reçu une demande à travers son rôle de serveur et a exécuté un des services que doit fournir tout agent serveur. Le client ne sait à priori pas qui va exécuter sa demande, ni dans le cadre de quel rôle il va le faire. Il suffit que pour une certaine raison (les serveurs sont saturés) ce ne soient plus les serveurs qui rapportent les menus mais les caissiers, pour que l'exécuteur de la requête « apporter menu » ne soit plus un agent jouant le rôle serveur mais plutôt un agent caissier. Un agent qui utilise un rôle pour obtenir un service ne connaît pas à l'avance l'agent qui le lui fournira.

Principe 2 : *Pour obtenir un service, un agent utilisateur s'adresse au rôle qui le mettra en relation avec un agent joueur du rôle qui fournira le service requis.*

Le rôle de serveur, précise qu'un serveur travaille six heures par jour, la rémunération d'une heure de travail, qu'il a droit à une heure de pause, que le nombre de serveurs présents en même temps ne doit pas dépasser un certain seuil et qu'il faut au moins un serveur en service à tout instant : ce sont les propriétés internes du rôle. Pour entamer son service le serveur doit être en tenu de travail et équiper de son carnet de commandes : ce sont les conditions pour jouer le rôle. Tout serveur doit fournir les services « apporter les menu », « prendre une commande », « apporter les plats », « apporter la facture », « débarrasser la table » :

ce sont les services offerts par le rôle serveur. Toutes ces propriétés sont internes au rôle et sont indépendantes des serveurs qui vont le jouer. Tout joueur du rôle se conformera à ces propriétés. Toutefois, certaines propriétés ne concerneront pas exclusivement le rôle serveur ou exclusivement les serveurs mais se situent dans la relation entre le rôle serveur et les serveurs.

En effet, le serveur saisit les commandes des clients et les mémorise dans un carnet (ou autre outil informatisé). Le carnet est la relation qui réunit le serveur (être humain) avec son rôle de serveur. Chaque serveur a son propre carnet de commandes correspondant à son activité en tant que serveur. Les notes qu'un serveur mémorise dans le carnet (heure de début de service, commandes des clients, heure de fin de service, incident ...) ne le concernent pas en tant qu'être humain et ne sont pas générales à tous les serveurs. Ces notes ne concernent que sa relation avec son rôle de serveur pendant la durée de son service. Une fois, son service terminé et qu'il quitte son rôle de serveur, le carnet, ou plutôt les feuilles de cette journée de travail, peuvent être archivées ou détruites. Le serveur n'en gardera aucune trace.

Principe3 : *Un agent s'enregistre comme joueur dans le rôle et exécute concrètement son rôle dans une entité le mettant en relation avec son rôle et mémorisant les informations concernant cette relation.*

C'est sur cette base de réflexion que nous allons proposer notre modèle, basé sur les concepts d'agent, groupe, rôle et service. Nous nous plaçons à un niveau d'abstraction assez élevé, permettant d'exprimer un modèle d'organisation suffisamment générique de systèmes multi-agents supportant différents types d'agents et adaptable à d'autres modèles et architectures d'interactions permettant son déploiement à très grande échelle. L'une des propriétés de base de ce système réside dans sa simplicité et sa conformité à la réalité. Ce compromis trouvé entre, simplicité descriptive et puissance conceptuelle, est une réelle satisfaction.

4.2 Concepts centraux

En guise de prélude au modèle AGRS que nous proposons, nous allons tout d'abord revenir sur l'analyse précédente. Les principes énoncés, précédemment permettent de dégager quelques recommandations, que nous devons respecter pour bâtir notre modèle.

- Il est important d'enrichir la définition du concept de rôle pour être utilisable, indépendamment de l'architecture des agents.
- Il faut préciser ce qu'on attend du rôle (coté utilisateur). Les agents doivent savoir pourquoi utiliser un rôle plutôt qu'un autre.
- Il faut préciser qui peut jouer ou utiliser un rôle donné et sous quelles conditions. L'agent qui joue le rôle doit satisfaire certaines conditions et les respecter tant qu'il est joueur du rôle.
- L'agent joueur doit aussi savoir ce qu'il gagne en exécutant les recommandations du rôle.
- Il faudra assurer la dynamique de la structure, à un niveau assez haut d'abstraction, et ce dès sa conception. Les rôles et groupes doivent se doter d'outils assurant leur dynamique.
- Distinguer agent, rôle et agent jouant un rôle est nécessaire puisqu'il y a des variables qui sont associées à la relation temporaire entre un agent et le rôle qu'il joue dans un groupe donné.

Notre modèle est l'association de trois concepts de base agent, group et rôle, liés par le concept de service. Le concept de service est la base de toute interaction entre agents, il est

aussi utilisé pour décrire la relation entre agents et rôles. Cette relation est représentée par le concept de AgentInRole qui cadre l'activité de l'agent pendant l'exécution du rôle qu'il joue.

La figure 4.1 ci-dessous présente le diagramme UML de ce modèle de base.

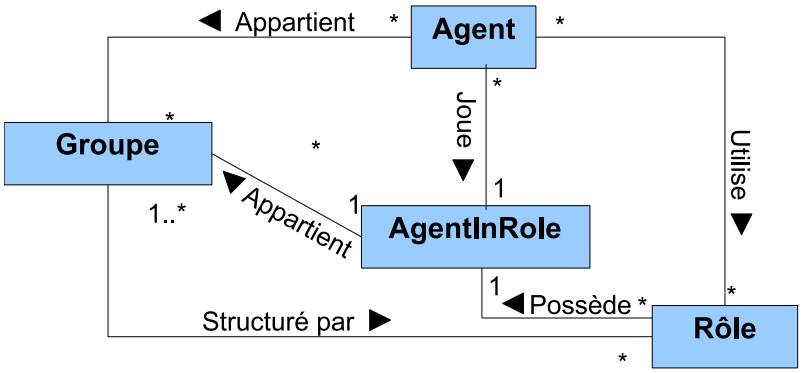

Fig. 4.1 – Modélisation UML simplifiée du modèle AGRS.

4.2.1 Le service

Les services constituent dans notre vision la moelle épinière du modèle. Il est la colle qui fait lier les agents aux rôles et les agents entre eux. Un service est la description d'un ensemble de fonctionnalités que doit proposer ou demander un agent. Nous voulons que tout soit exprimable en terme de service, même au niveau implémentation : les fonctions de bases du noyau (envoi/réception de messages, création d'un groupe, d'un rôle, exploration de l'accointance de l'agent ...). D'un point de vue plus concret, un service doit être vu comme un concept, faisant partie d'une ontologie de services d'un domaine donnée. Le résultat attendu par le fournisseur du service doit être communément convenu par les utilisateurs de l'ontologie.

Il faut faire la distinction entre service qui est une description et son implémentation. Le service peut être réellement implémenté (on le verra pour les service fournis par le rôle aux agents joueurs), il est alors un composant qui respecte sa description au niveau ontologique. Ainsi en plus du nom, les types de résultats attendus en retour et les types d'arguments passés en paramètre doivent être précisés dans la description du service au niveau de l'ontologie.

Nous pouvons émettre cette définition pour le concept de service :

« Un service est la description d'un ensemble de fonctionnalités que doit proposer l'agent qui se déclare pouvoir offrir le service ou que cherche l'agent qui souhaite l'obtenir. Il est le maillon qui lie les agents aux rôles et le sujet de toute interactions entre agents. »

Un service peut aussi être composé de sous services. Le service « communication » est la composition du service « sendMessage » et « receiveMessage ». La granularité et le niveau de composition des services restent tributaires du type d'application dans lequel le service est défini. Il est néanmoins le rôle du concepteur de définir un ensemble de services assez cohérent, afin de garantir un bon fonctionnement de l'ensemble de son application. La réalisation d'une tâche par un agent requiert l'exploitation d'un certain nombre de services. Si l'agent possède le service requis à un instant donné il en fait usage, sinon il doit trouver un rôle proposant ce service dans un des groupes auxquels il appartient ou ailleurs. Nous expliquerons dans la suite comment les services sont recherchés puis utilisés par les agents.

Il est intéressant de remarquer que l'utilisation du concept de service nous permet de réaliser une abstraction entre le « quoi » et le « comment », entre ce qui est sensé être échangé entre agents (fonctionnalités en terme de service offerts et obtenus) et la façon avec laquelle ces services seront réalisés. En effet, notons que ce ne sont pas les services eux-mêmes qui sont importants mais plutôt les fonctionnalités qu'ils représentent. Le service est décrit d'une façon unique dans une organisation donnée, mais peut être implémenté différemment selon les choix du concepteur.

L'idée est de faire du service un moyen d'abstraction par rapport au niveau applicatif. Chaque agent ou rôle peut implémenter le service à sa façon. Ainsi on peut avoir une équivalence entre un service est l'entité qui l'implémente. Autrement dit, au sens de la programmation à un service correspondent une ou plusieurs méthodes (au sens objet). Les résultats de chaque implémentation ne sont pas garantis, des mécanismes de certification du résultat de chaque service sont nécessaires.

4.2.2 Agent

Certes, nous souhaitons présenter un modèle organisationnel indépendant de l'architecture interne des agents, mais nous tenons aussi à ce qu'il soit opérationnel. La difficulté réside dans le moyen de les formaliser les comportements des agents dans un langage non ambiguë et indépendant de leur plateforme d'exécution. Une réponse à cette contrainte est d'exprimer les comportements d'un agent, abstraction faite de son architecture et de son type (cognitif ou réactif), en termes de services. Nous proposons de se focaliser sur ce que fait l'agent plutôt ce qu'il est. En effet, dans un environnement ouvert, il est relativement aisé de demander à chaque agent de décrire ses comportements, capacités et besoins plutôt que d'imposer une certaine architecture interne d'agents. Notre motivation première est surtout de s'affranchir d'un moyen d'expression permettant à l'agent de se positionner selon ses besoins et capacités dans le cadre de son organisation, quitte à perdre un peu de généricité.

L'agent est une entité autonome agissant dans un contexte social, par le biais de ses capacités, exprimés en terme de services offerts aux autres agents à travers l'organisation qui est constituée de groupes et de rôles. L'agent peut jouer un ou plusieurs rôles dans différents groupes. Les besoins de l'agent sont aussi décrits en terme de services demandés à d'autres agents par le biais des rôles dans le cadre des groupes. Toute l'activité de l'agent dans la société, indépendamment de la façon dont elle est réalisée, s'exprime en terme de services. A ce stade, même les opérations de créer, rejoindre ou quitter un groupe, de créer, jouer ou quitter un rôle sont exprimées en terme de demande de service **??**.

Un service peut être vu comme étant l'expression d'une capacité de l'agent qu'il désire mettre au service des autres agents de l'organisation. Il ne s'agit pas ici d'une contrainte que nous imposons sur l'architecture interne de l'agent. Les agents peuvent être conçus en terme de composants (comme c'est le cas de Magique ou Moca), mais cela n'est pas obligatoire. En effet, quelque soit l'architecture interne de l'agent, ce que nous imposons c'est que l'agent puisse exprimer ses capacités et ses besoins en terme de services. L'agent peut offrir un service (en l'exécutant), recevoir des requêtes d'exécution de services, de demander un service et recevoir une réponse à une requête. Au niveau de l'implémentation, on peut imaginer une couche qui permet de faire la traduction des capacités et besoins de l'agent en terme de services exploitables au niveau de l'organisation. Nous pouvons modéliser un agent comme sur la figure 4.2 ci-dessous. Le modèle se base sur la notion de borne (d'entrée et sortie), utilisée par le langage de modélisation BRIC [Ferber, 1999].

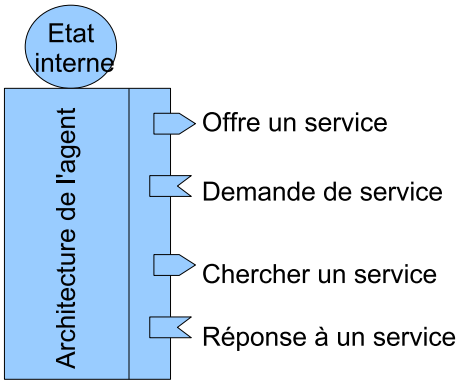

FIG. 4.2 – Modèle générique d'un agent AGRS.

Comme l'indique le modèle générique de l'agent, nous faisons abstraction de l'architecture et l'état interne de l'agent. Ce qui nous intéresse est l'activité de l'agent : ces besoins et capacités exprimés en terme de service à fournir et service à obtenir.

Il est intéressant de noter qu'à ce stade on n'impose pas non plus la représentation de la structure sociale. Celle-ci peut avoir une existence interne au niveau de l'agent comme l'agent peut initialement être plongé et laissé à son propre destin dans la découverte de son environnement social. Libre au concepteur d'internaliser, par des connaissances de l'agent, une description de la structure sociale. De toutes les façons, vu le contexte d'ouverture et dynamique, dans lequel nous situons notre travail, l'agent devra à tout instant remettre en cause ses connaissances sur la structure sociale puisque celle-ci est évolutive.

4.2.3 Role

Le rôle dans AGR [Gutknecht, 2001] est une représentation abstraite d'une fonction, d'un point d'interaction ou d'une identification d'un agent au sein d'un groupe. Le rôle est supposé pouvoir contraindre le comportement de l'agent sans pour autant pouvoir l'exprimer au niveau du modèle, ni d'ailleurs plus tard au niveau de l'implémentation [MadKit, 1998]. Cela sous entend que les agents doivent savoir à l'avance ce qu'on attend d'eux en jouant le rôle, quelle position ils auront et quels droits et obligations ils obtiendront. Notre contribution vise à réduire cette abstraction, dissiper ce flou et fournir au rôle les moyens d'exprimer d'une façon plus claire et explicite ce qu'il désigne dans l'organisation. Tout en gardant l'aspect générique et permettre au rôle de se détacher du niveau applicatif, nous proposons une définition du concept de rôle qui regroupe dynamique et ouverture.

Le rôle est la représentation abstraite, d'un ou plusieurs services offerts par les agents qui le jouent. Il cadre l'interaction entre agents : joueurs offrant les services et agents utilisateurs les requérant. Plus simplement il est la frontière entre agents joueurs et agents utilisateurs. Les agents joueurs proposent des services et les agents utilisateurs les contactent à travers le rôle pour les utiliser. Un rôle peut être une coquille vide et ne contenir aucune description de service, on se retrouve dans le cadre d'un rôle classique au sens AGR. Pour assurer la dynamique du rôle, on propose l'utilisation de règles modifiant les conditions de jouer ou d'utiliser le rôle et qui peuvent modifier les services qu'il propose.

Nous proposons alors de définir un rôle par le sextuplet <Gr, Nm, Cd, Sv, At, Rl> où :

1) Gr : l'identifiant du groupe dans lequel le rôle appartient.

2) Nm : le nom du rôle. Un rôle ayant une seule instanciation dans un groupe donné.

3) Cd : l'ensemble de conditions du rôle nous distinguons deux types de conditions :

3.1) CdJ : Les conditions que doit satisfaire un agent désirant jouer le rôle. 3.2) CdU : Les conditions que doit satisfaire un agent utilisateur du rôle.

4) Sv : L'ensemble des services définis au niveau du rôle. Nous distinguons :

4.1) Les services offerts qu'utiliseront les agents utilisateurs du rôle.

4.2) Services fournis aux agents qui jouent le rôle. Tant que l'agent joue ce rôle il peut utiliser ces services.

4.3) L'ensemble des services transférables que peut acquérir les agents joueurs et qui les garderont même quand ils lâchent le rôle.

5) At : Les attributs internes du rôle. A distinguer des attributs concernant l'association du rôle à un agent : (Agent dans le rôle) que nous étudierons plus tard.

6) Rl : L'ensemble de règles qui gèrent le rôle et sa dynamique elles utilisent et agissent sur les conditions et les attributs pour modifier l'état du rôle.

La figure 4.3 donne une modélisation du rôle selon le modèle AGRS :

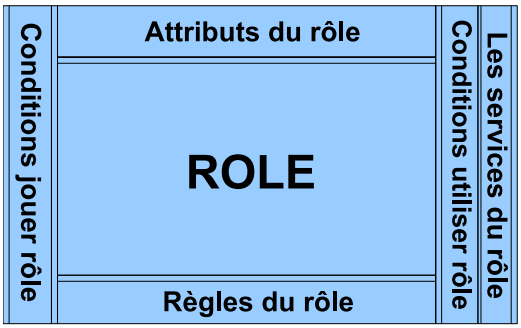

FIG. 4.3 – Architecture du rôle.

a) Les conditions d'un rôle

Jouer un rôle tout comme l'utiliser peut être soumis à des conditions. Jouer le rôle de serveur sans être en uniforme n'est pas possible. Demander un menu à un serveur ne peut se faire qu'une fois le client à table. Certaines conditions peuvent être catégorisées en classes comme le propose (Odell) [Odell *et al.* , 2003] pour les agents en spécifiant pour chaque classe d'agents quels types de rôles ils peuvent jouer. Préciser pour un agent la liste de rôles qu'il peut jouer à l'avance n'a pas de sens dans des systèmes ouverts à moins de lancer en continue des recherches de nouveaux rôles que l'agent peut jouer selon son état.

Les conditions pour jouer un rôle doivent se situer du côte du rôle et pas de celui des agents. En effet, le rôle évolue et fixer des conditions pour jouer un rôle du côté des agents revient à mettre à jour ces conditions au niveau de tous les agents à chaque modification pour assurer la cohérence du système. C'est donc au rôle de vérifier en cas de besoin si un agent peut ou non le jouer. Lorsqu'un agent désire ou requiert jouer un rôle il formule une demande qui, selon son état et les conditions du rôle cible, sera approuvée ou refusée. Mais satisfaire les conditions à un instant donné ne peut pas être garanti dans la durée. En effet, les agents évoluent et les rôles aussi, il est donc nécessaire de prévoir des mécanismes qui permettent de recalculer la satisfaction des conditions d'un rôle, à chaque fois que l'agent agira dans

l'organisation en utilisant ce rôle. De la même façon, l'utilisation d'un rôle peut être sujet à des conditions qui se situent au niveau du rôle.

Au niveau du modèle nous ne posons aucune contrainte sur le moyen de vérification des contraintes. Pour ce faire, plusieurs mécanismes peuvent être implémentés. Nous pensons plus particulièrement aux mécanismes de certifications comme celui utilisé dans RoleSystem (cabri&al). Dans RoleSystem, c'est au noyau de délivrer des certificats aux agents en vérifiant qu'ils répondent bien aux contraintes imposées par le rôle qu'ils désirent jouer.

b) Les services d'un rôle

Au niveau du rôle nous distinguons trois types de service : offerts, fournis et transférables. Nous désignons par service, la description d'une fonction qui peut être ou non implémenté au niveau du rôle (les services fournis sont implémentés). Afin de mieux cerner la signification de chacun de ces concepts, traitons le rôle « caissier ». Imaginons que la caisse s'ouvre avec une clé, qu'elle contient une calculatrice pour calculer et que le caissier doit avoir des connaissances en comptabilité pour effectuer les comptes de la journée. « Encaisser, comptabiliser, calculer » sont les services offerts par le rôle caissier. Pour payer sa consommation, un agent client s'adressera à un agent jouant le rôle caissier. De même tout agent désirant effectuer une opération de calcul de comptabilité s'adressera à un agent jouant le rôle caissier. Tout agent caissier doit pouvoir répondre à chaque demande concernant un des services offerts par le rôle caissier. Les services « ouvrir caisse, calculer » sont fournis par le rôle à ses agents joueurs. Du point de vue programmation ces services sont implémentés par le rôle caissier. Ouvrir la caisse est possible car la clé est fournie par le rôle caissier. De même calculer est possible grâce à la calculatrice. Notons que les services fournis ne sont pas nécessairement offerts. On ne propose pas aux agents clients le service « Ouvrir la caisse ». S'il est possible de faire des copies de la clé et de la transférer au caissier, ce dernier aura la possibilité d'ouvrir la caisse même s'il ne joue plus le rôle caissier. Le service ouvrir caisse est donc transférable du rôle aux agents joueurs. Remarquons qu'un agent joueur du rôle « caissier », doit avoir les compétences requises en comptabilité. Les agents joueurs du rôle doivent être capable d'offrir le service « comptabiliser », c'est une condition pour pouvoir jouer le rôle qui est vérifié avant d'autoriser à un agent de jouer ce rôle.

Nous proposons les définitions suivantes pour les différents types de services du modèle AGRS :

- Les services offerts : Regroupent la description de l'ensemble de services que peut solliciter un agent utilisateur du rôle. Ces services ne sont pas nécessairement implémentés par le rôle. Quand un agent désire utiliser le rôle et obtenir un des services offerts, il sera redirigé vers un agent joueur du rôle qui implémente le service et a l'obligation d'exécuter la requête.
- Les services fournis : Décrivent les services qu'implémente le rôle et met à la disposition des agents joueurs du rôle. Uns service fourni peut être parmi les services offerts, mais ce n'est pas toujours le cas. A la description d'un service fourni, doit correspondre une implémentation au niveau du rôle. Nous pouvons voir cette implémentation comme un composant. L'agent joueur du rôle, qui invoquera un service fourni, n'a pas à se soucier de la façon avec laquelle il sera implémenté par le rôle. Du point de vue des agents, l'implémentation d'un services fourni est une boite noire.
- Les services transférables : Décrivent l'ensemble des services fournis par le rôle aux agents joueurs et qui peuvent être transférés du rôle vers les agents joueurs, s'ils le souhaitent. Si l'agent quitte le rôle il pourra toujours utiliser les services qu'il a importé des rôles

qu'il a joué. Nous pouvons voir ça comme un enrichissement des capacités de l'agent. Ce mécanisme de transfert de service permet la propagation des capacités en dehors d'une seule organisation. Comme on le verra plus tard, un agent peut appartenir à plusieurs organisations, et il peut créer des rôles (s'il a l'autorisation nécessaire). En créant des rôles basés sur un ou plusieurs services transférés d'un rôle d'une organisation externe, l'agent joue le rôle de médium entre organisations à priori indépendantes.

Remarque : : *Par soucis de simplicité, on confondra dans la suite de ce travail les concepts de service fourni et son implémentation de même pour les services transférables. Cela se justifie par le fait que le service en soi n'est qu'une description et qu'à tout service fourni correspond une implémentation d'où cette fusion des deux concepts.*

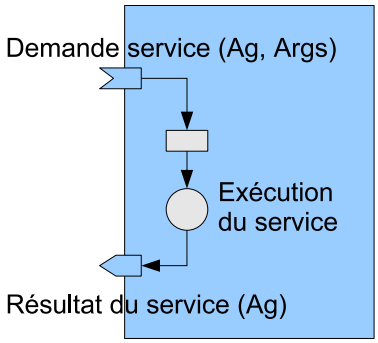

FIG. 4.4 – Le comportement d'un service fourni par le rôle.

c) Les attributs d'un rôle

Nous désignons par attribut du rôle toute information qui peut être utilisée durant le cycle de vie d'un rôle dans un groupe et qui ne concerne que le rôle, indépendamment des agents qui le jouent ou l'utilisent. Certains de ces attributs sont partagés par tous les rôles et font donc partie de la classe abstraite Rôle. Ainsi les variables de cardinalité : nombre d'agents joueurs, nombre d'agents utilisateurs, nombre d'agents joueurs (et utilisateurs) maximum et minimum autorisés sont des attributs qu'on peut considérer comme partagés par tous les rôles. Nous n'imposons pas leurs existences au niveau du modèle bien que nous les proposons au niveau de l'implémentation.

Le concepteur du rôle créé ses attributs selon les besoins de l'application. Les attributs peuvent varier durant le cycle de vie du rôle et c'est aux règles de guider ces modifications. Si on reprend le cas du serveur de restaurant, le nombre maximum d'agents qui peuvent jouer le rôle dans un groupe donné est un exemple d'attributs. Ce nombre peut être changé avec les règles du rôle ou celles du groupe.

Les variables qui ont trait à l'évolution de l'agent joueur dans son rôle ne sont pas rattachés au rôle, mais à ce que nous appellerons dans la suite « AgentInRole ».

d) Les Règles d'un rôle

Bien que certains considèrent que les rôles sociaux sont des entités assez stables et invariantes dans toute organisation multi-agents [Boella *et al.* , 2006], il n'en demeure pas moins que cette idée est très loin de la réalité dans différents types d'applications. La vérité est que la position du rôle dans l'organisation sociale est assez stable et les relations entre les rôles varient rarement. Par contre les services que doit fournir le rôle, les conditions requises pour jouer ou utiliser un rôle ainsi que les attributs du rôle sont dans la plupart des cas des proprié-

tés dynamiques. Un agent peut dans deux instants différents avec le même état et les mêmes capacités recevoir deux réponses différentes à une requête de jouer ou utiliser un même rôle. Si les serveurs étaient saturés le client désirant récupérer un menu n'aurait eu aucune réponse à sa demande. Ce qui s'est passé c'est que les conditions d'utilisation du rôle serveur limitent le nombre d'utilisateurs à un certain seuil, dépassé lors de la première demande du client. Les conditions pour jouer le rôle peuvent aussi varier selon son état interne et l'état global du groupe auquel il est rattaché. Ainsi si les serveurs du restaurant sont débordés les conditions pour jouer le rôle serveur, peuvent être plus souples et d'autres agents pourront jouer ce rôle pour pallier un besoin du système (par exemple : jouer le rôle serveur sans tablier). L'état interne du rôle doit être géré par un ensemble de règles qui assurent sa dynamique en fonction des besoins du système. Ces règles manipulent les attributs du rôle, ses conditions et ses services et peuvent elles même être modifiées.

Dans notre modèle nous distinguons trois types de règles :

- Les règles intrinsèques : sont celles qui concernent le rôle proprement dit et pas l'implication des agents dans les rôles (AgentInRole), c'est de ces règles là qu'il est question dans cette section. Elles prennent en paramètres, les attributs du rôles, l'ensemble des services (offerts, fournis et transférables) et les modifient en sortie.
- Les règles extrinsèques : Comme son nom l'indique, une règle extrinsèque est externe au rôle. En fait, ces règles gèrent l'évolution et la dynamique de l'agent dans son rôle. Elles ne modifient pas les attributs ou services du rôle mais plutôt celle de l'instance AgentInRole.
- Les règles globales : se situent au niveau du groupe et peuvent faire intervenir plusieurs rôles ou manipuler des attributs généraux qui ne concernent pas un rôle particulier.

Notons pour finir que nous n'imposons aucune contrainte à ce niveau sur la façon d'implémenter ou interpréter les règles. C'est au concepteur de mettre en oeuvre les règles de son organisation.

4.2.4 Jouer et utiliser un rôle : AgentInRole

Nous considérons que jouer un rôle n'est pas une simple association entre l'agent et le rôle, l'état de ce dernier et ses conditions ainsi que ses règles internes peuvent être altérées par cet événement. Cependant, il est intéressant de voir comment se comporte l'agent en jouant son rôle. Comme nous l'avons expliqué précédemment, tout rôle possède un ensemble d'attributs internes qui varient selon l'état global du système et l'état des agents qui le jouent et l'utilisent. De même, tout agent en jouant un rôle possède des attributs qui concernent sa relation avec le rôle. Un agent qui joue le rôle de serveur doit mémoriser les commandes qui lui sont demandées, le nombre d'heures qu'il a passé en service... Ces informations ne concernent pas l'agent et ne peuvent pas être stockées dans le code de l'agent, mais dans une entité le mettant en relation avec le rôle qu'il joue. Nous appelons cette entité : « AgentInRole ». Cette entité devra contenir les règles, attributs et services du rôle et aura en plus des attributs propres à l'activité de l'agent joueur dans son rôle. Le rôle est une sorte de définition de haut niveau qui sera mise en oeuvre concrètement par les AgentInRole.

En essayant de modéliser cette relation on s'est très vite rendu compte qu'on se trouvait dans le cas du patron de conception Abstraction occurrence [Lethbridge, 2004]. Avant de modéliser la relation entre rôle, agent et AgentInRole, sous forme de diagramme de classe, essayons d'expliquer en quoi consiste ce patron de conception.

Le principe d'abstraction-occurrence est pertinent dans le cas où il existe une entité qui est conceptuellement plus haute et représente une abstraction d'une autre entité qui applique cette abstraction. Prenons un cas d'école simplifié, extérieur au contexte des SMA : Les série télévisée. Il est possible de modéliser un épisode de série TV par une seule entité qui contient le nom et le producteur de la série, le numéro, titre et synthèse de l'épisode. Nous remarquons que de ces attributs se dégagent deux catégories : une partie fixe et qui varie pas selon les épisodes c'est la définition d'une série TV il s'agit du (nom de la Série, Producteur). La deuxième catégorie concerne un épisode en particulier (le numéro, titre, synthèse). Le principe du patron de conception consiste à séparer ces deux catégories en deux classes différentes, l'une est l'abstraction et concerne dans ce cas la série, la deuxième est l'occurrence qui concerne les épisodes de la série. De cette façon, on évite de répéter les informations fixes qui se répètent dans chaque occurrence.

Dans notre modèle, c'est typiquement le cas des Rôles et les relations AgentInRole. En effet, la définition des comportements qu'un agent joueur doit avoir est présente dans l'entité Rôle, le respect de ces comportement (jouer le rôle) de façon effective se fait dans l'entité AgentInRole.

L'agent est donc en association avec un rôle et le fait de jouer le rôle se manifeste dans « AgentInRole ». En d'autres termes l'agent s'enregistre comme joueur dans le rôle et exécute concrètement son rôle dans AgentInRôle.

Ci-dessous le schéma UML modélisant la relation entre Rôle, Agent et AgentInRole au niveau classes du modèle et de l'exemple. Notons la relation design-pattern abstraction-occurrence entre Rôle et AgentInRôle. Comme on l'a expliqué cette relation est justifiée par le fait qu'un rôle est à la fois une abstraction du concept AgentInRole qui constitue son exécution réelle et qu'à un rôle sont associés plusieurs occurrence AgentInRôles. D'autre part, une relation ternaire entre Agent Rôle et AgentInRôle pour modéliser le fait qu'un agent joue un rôle dans l'entité AgentInRôle.

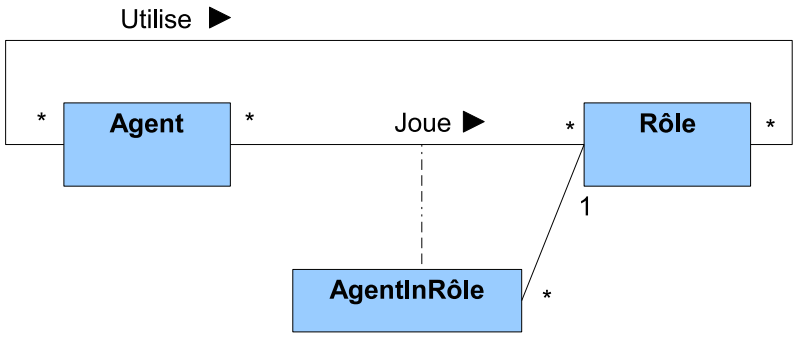

FIG. 4.5 – Diagramme des classes Agent Rôle AgentInRôle.

La figure 4.6 illustre un exemple d'utilisation du modèle. La classe Personnel hérite de la classe agent et la classe PersonnelInServeur est une extension de AgentInRole. Les agents de type personnel jouent le rôle Serveur dans des entités de type PesonnelInServeur. L'utilisation d'une classe de type prédéfini (Personnel) pour jouer le rôle serveur n'est que pour la modélisation de l'exemple, il est évident que jouer le rôle serveur ne dépend pas d'une classe précise, à moins que ça soit spécifié par les règles du rôle imposées par le concepteur. Nous n'avons pas modélisé l'utilisation du rôle Serveur, mais tout agent du système peut prétendre utiliser le rôle. Après, selon les choix du concepteur, il pourra ajouter des restrictions au niveau du rôle pour accepter certains agents et en refuser d'autres.

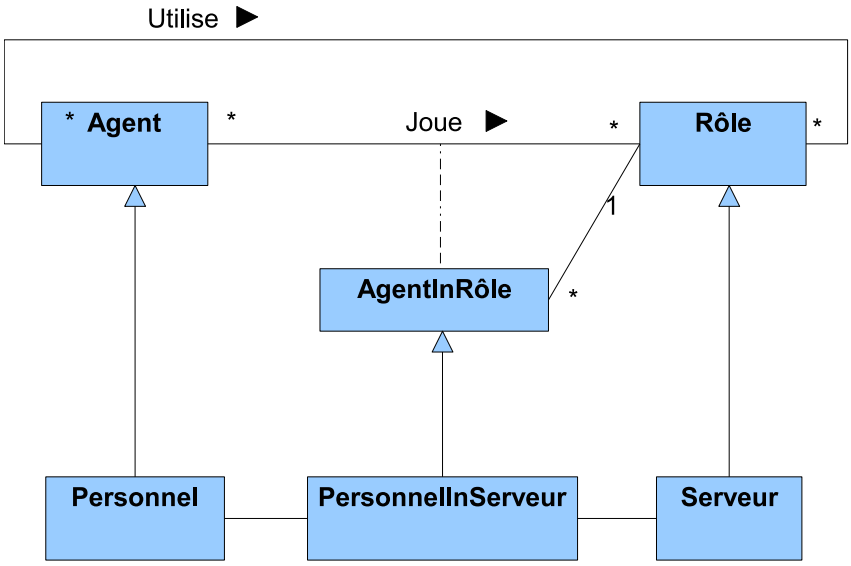

FIG. 4.6 – Exemple d'utilisation du modèle AGRS.

Essayons maintenant de modéliser le comportement de l'entité Rôle en réponse à une demande de le jouer, puis à celle de l'utiliser. L'agent A s'adresse au rôle et demande de devenir joueur en présentant un certificat garantissant qu'il vérifie les conditions du rôle. Le rôle effectue le vérification de validité du certificat, vérifie que les règles autorisent cet évènement (par exemple : le nombre maximum de joueurs n'est pas dépassé). Une fois les vérifications terminées, le rôle crée une instance de l'entité AgentInRole et renvoie sa référence à l'agent demandeur. A partir de cet instant, l'agent joueur n'a plus d'interaction directe avec le rôle mais plutôt avec l'entité AgentInRôle. A contrario, les entités AgentInRôle restent en relation directe avec l'entité Rôle qu'elles concrétisent. Ceci s'explique par le fait que les rôles sont dynamiques, grâce à leurs règles, ainsi les modifications qui peuvent avoir lieu au niveau du rôle, doivent se refléter immédiatement sur les entités AgentInRole pour garantir la synchronisation de l'organisation. Si par exemple une règle modifie le rôle, en supprimant un des services offerts, il sera dangereux pour l'intégrité du système de ne pas reproduire cette modification majeure sur l'ensemble des entités AgentInRole. Les agents ne devront plus répondre à des demandes de service concernant le service supprimé. A la réception d'une demande de l'utiliser, le rôle effectue les vérifications nécessaires (certificat conforme aux conditions, et règles autorisant un nouveau joueur...) et sélectionne un agent joueur, parmi sa liste d'agents joueurs, pour le mettre en relation avec le nouvel agent utilisateur. L'agent utilisateur, reçoit une référence de l'entité AgentInRôle et n'aura plus aucune interaction avec le rôle pour ce qui est de la demande d'exécution des services. En réalité, l'agent utilisateur n'a même pas de relation directe, sauf exception imposée par le concepteur, avec l'agent joueur. Toutes les interactions entre agent joueur et utilisateur doivent s'effectuer dans le cadre de l'entité AgentInRôle.

Quand l'agent utilisateur du rôle invoque l'exécution d'un service, il le fait à travers l'entité AgentInRôle. Celle-ci peut effectuer certains traitements (par exemple mémoriser la demande), puis transmet la requête à son agent joueur du rôle. Ce dernier, décide de répondre ou d'ignorer la requête. Refuser une demande peut être sujette à une sanction. La réponse au service exécuté passe par l'entité AgentInRôle qui la transmet à l'agent utilisateur qui l'a demandé.

Des règles qui concernent la réalité de jouer un rôle pour un agent sont nécessaires. Le salaire journalier d'un agent jouant le rôle serveur est calculé en fonction des heures de services. Une règle récompensant ou pénalisant l'agent qui joue le rôle, se situe dans la relation AgentInRole. Nous appelons règles extrinsèques d'un rôle toute règle qui gère l'évolution et la dynamique de l'agent dans son rôle. Pour un concepteur, il est facile de distinguer une règle intrinsèque d'une règle extrinsèque, il suffit de voir si la règle s'applique uniquement aux attributs du rôle auquel cas la règle est intrinsèque. Si par contre la règle fait appel à un attribut de l'entité AgentInRole elle ne peut qu'être intrinsèque. Nous n'imposons aucune méthode pour écrire les règles à ce niveau. Par contre, ce qu'il faut rappeler c'est que toutes les instances de l'entité AgentInRôle d'un même rôle doivent être identiques à leurs créations et donc avoir les même règles.

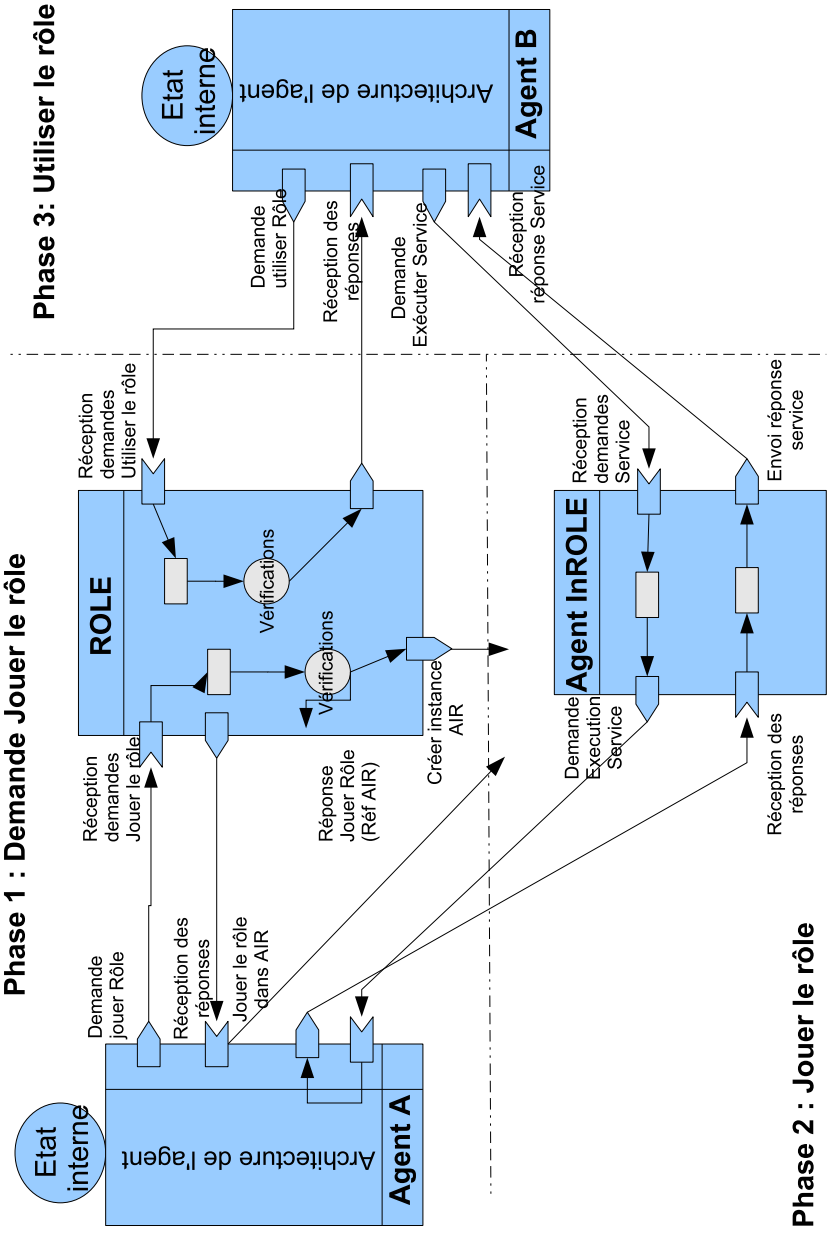

FIG. 4.7 – Le cycle d'interactions entre agent joueur et agent utilisateur d'un rôle.

La figure 4.8 ci-dessous présente le niveau applicatif de l'exemple de restauration. Le rôle serveur hérite de la classe Rôle du modèle. L'agent "Eric" demande de jouer le rôle "ServeurMacDo". Après vérification des conditions, l'agent "Eric" devient joueur du rôle "ServeurMacDo". Le rôle serveur crée la relation "EricServeur" instance de la classe "PersonnelServeur" et qui sera le contexte d'exécution du rôle "ServeurMacDo" par l'agent "Eric". Un agent client "Jhon" désire utiliser le rôle Serveur pour avoir le service "Menu". Le rôle "ServeurMacDo" vérifie qu'il peut utiliser le rôle et lui indique la relation "EricServeur" dans laquelle il pourra l'obtenir. L'agent "Jhon" demande à travers "EricServeur" à l'agent "Eric" de lui fournir le menu. L'agent répond à la requête de "Jhon" toujours dans le cadre de la relation "AgentInRole" : "EricServeur". Notons que toute demande d'utilisation d'un service, ainsi que la réponse à cette requête, peuvent être sauvegardées dans l'entité AgentInRôle.

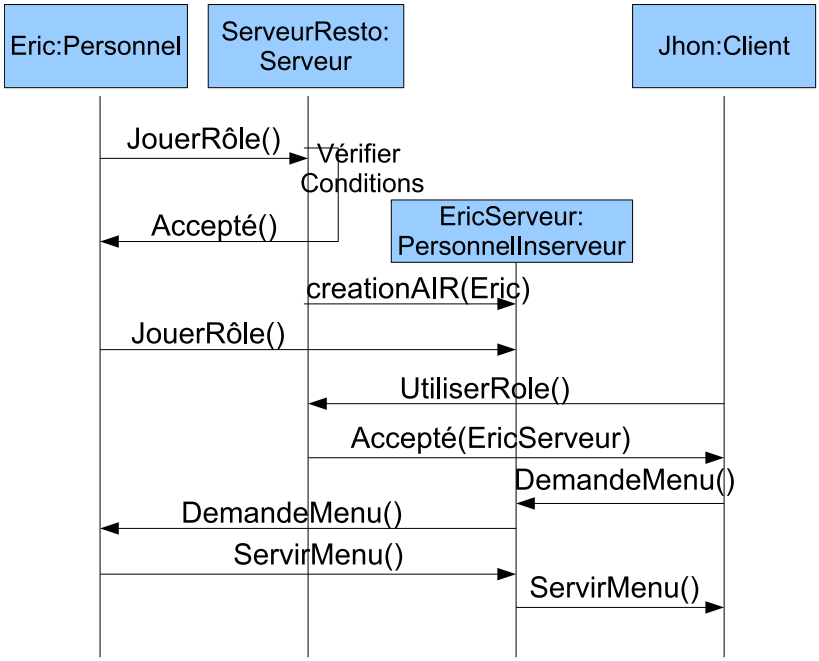

FIG. 4.8 – Diagramme de séquences : Jouer et utiliser le rôle serveur.

Une question qui a plus trait à l'implémentation mais qu'on peut légitimement se poser à ce stade, il s'agit de l'emplacement physique de l'entité AgentInRole. Si l'agent joueur et agent utilisateur du rôle se trouvent normalement dans deux sites différents, où se positionne l'entité AgentInRole. Comme la plupart des choix de l'ordre de l'implémentation, nous n'imposons pas à ce niveau une solution unique. L'entité AgentInRole peut se trouver sur le même noyau que l'agent utilisateur, sur un noyau autre que les deux interlocuteurs ou sur le même noyaux que l'agent joueur. Nous opterons pour cette dernière solution lors de la mise en oeuvre de notre modèle pour des raisons d'optimisation au niveau des échanges de messages.

4.2.5 Groupe

Un groupe est la structure sociale qui conceptualise un rassemblement dynamique d'agents. Le groupe se défini par des agrégats de rôles. Chaque agent fait partie d'un ou plusieurs groupes et joue et utilise un ou plusieurs rôles dans ces groupes. La structure du groupe est une description abstraite du groupe et qui décrit l'ensemble des rôles qui le forment.

Nous souhaitons avoir des groupes ouverts favorisant la dynamique mais rien ne met ça en oeuvre dans le modèle AGR. Dans cette section nous expliquons les modifications apportées au concept de groupe dans le modèle AGR, afin de supporter les modifications apportées au concept de rôle. Le groupe doit tenir compte de ces contraintes : ouverture et dynamique.

a) Problèmes et contexte

L'ouverture et dynamique des groupes changent complètement la façon de concevoir et d'implémenter les systèmes multi-agents. Ces deux propriétés introduisent donc des contraintes supplémentaires sur la conception, création, gestion et maintenance des groupes d'agents. Nous essayons de mettre en lumière les points qu'il faut traiter afin de tenir compte des propriétés d'ouverture et dynamique dans les groupes ouverts avant d'exposer les solutions que nous proposons pour répondre à ces besoins.

Groupes ouverts :

Hétérogénéité : Les groupes peuvent accueillir de nouveaux membres. Ces membres peuvent provenir de différents environnements. Bâtir la structure du groupe en se basant sur des présuppositions, concernant le raisonnement interne des agents ou sur leurs protocoles d'interactions, n'est pas possible dans un tel contexte. La structure du groupe doit faire abstraction des types d'agents. Elle doit aussi produire des concepts génériques, cadrant l'activité des agents et les interactions qui guident ces activités.

Visibilité externe : Les groupes doivent être visibles afin d'être découverts et donc accessibles par les agents. Un groupe invisible des agents externes ne peut pas être qualifié d'ouvert. La visibilité d'un groupe doit inclure les moyens d'y accéder tout comme la description des services proposés par chacun de ses rôles. Si un agent découvre un service contenu dans un groupe, il est logique qu'il puisse entrer dans le groupe et utiliser le service trouvé. Il faut que les moyens et conditions pour accéder à un groupe et utiliser un service donné soient visibles pour les agents externes.

Groupes dynamiques :

Dynamique organisationnelle :

Les relations inter-agents ainsi que celle entre agents et rôles peuvent changer après le lancement du système. Un agent doit avoir la possibilité d'obtenir un rôle et le quitter quand il le souhaite. Il doit pouvoir interagir avec un agent et rompre l'interaction selon ses besoins et contraintes.

Dynamique structurelle :

La structure du groupe doit être modifiable à tout instant. Si le groupe est structuré via ses rôles, ces derniers peuvent alors être supprimés, d'autres ajoutés selon l'état du système. Le groupe structure l'activité des agents, si ces agents expriment leurs activités par le biais de leurs services, il faut doter la structure du groupe de moyens assurant la prise en compte des changements qui peuvent subvenir. Certains services peuvent ne plus exister, d'autres peuvent apparaître, la structure du groupe se trouve affectée par ces changements et doit s'adapter en conséquence.

b) Solutions apportées

Nous nous proposons dans cette partie de résoudre les deux problèmes de dynamique et d'ouverture. Ce qui va nous intéresser dans ce paragraphe c'est de trouver un compromis entre la résolution du problème et la contrainte de généricité que nous nous sommes imposés comme propriété de base de notre modèle. Nous avons jusqu'ici présenté un jeu de concepts

de base (Rôle et Service) pour la description des interactions et des activités des agents, tout en évitant d'imposer une contrainte quelconque sur l'architecture interne des agents ou les types d'applications. Néanmoins, nous voudrions également proposer des solutions permettant de répondre aux besoins d'hétérogénéité, visibilité, dynamique structurelle et dynamique organisationnelle.

Groupes ouverts :

Si le rôle est décrit à travers ses services et contraintes, la description du groupe s'effectue à partir de ses rôles. Le groupe contient un ensemble de rôles offrant des services. Les agents joignent le groupe pour jouer ou utiliser ses rôles. Il faut que les agents puissent connaître les services offerts par les rôles ainsi que les conditions d'entrer dans le groupe. En même temps, un groupe n'a pas à dévoiler sa structure interne aux agents ne lui appartenant pas. Nous proposons donc, de rendre opaque la structure interne du groupe pour un agent externe. La structure du groupe est invisible de l'extérieur. Les agents externes n'aperçoivent du groupe que sa description. Cette description contient les conditions d'entrée au groupe et les services offerts par l'ensemble de rôles existants. Les agents désirant obtenir ou offrir un service doivent joindre un groupe dont un de ses rôles propose ce service. Le groupe reste donc le cadre de toute activité sociale. La suite des activités de l'agent se fera par le biais du rôle comme expliqué précédemment. Tout agent désirant obtenir un service, demande à utiliser un rôle le proposant, ce dernier le met en relation par le biais de la relation AgentInRole avec un agent joueur du rôle qui lui fournira le service recherché.

Groupes dynamiques :

Les agents découvrent les services offerts par l'ensemble des groupes qui leurs sont visibles et selon leurs besoins, joignent le (ou les) meilleur(s) groupe(s). Chaque groupe défini ses conditions d'entrée. En plus des conditions propres à chaque rôle, les agents externes sont donc amenés à satisfaire les conditions du groupe avant de jouer ou utiliser un rôle.

Comme pour les rôles, la dynamique de la structure d'un groupe est assurée grâce à l'utilisation de règles. Le concepteur du groupe établit selon ses besoins les règles du groupe. Nous distinguons règles fonctionnelles et méta-règles.

Les règles fonctionnelles : Le premier type de règle, concerne la gestion du groupe d'agents. Ces règles précisent les conditions d'accès, les cas de sanctions, les événements de mise à jour du groupe (gestion distribuée, on y reviendra ultérieurement). Ces règles ne peuvent être ni modifiées ni supprimées par les agents, et elles sont prioritaires par rapports aux règles non fonctionnelles.

$$G^C = \{g_i, 1 <= i <= n\},\ \textit{Ensemble de groupes d'une communauté C donnée.} \tag{4.1}$$

$$R^g = \{r_i, 1 <= j <= n\},\ \textit{Ensemble de règles d'un groupe g donné.} \tag{4.2}$$

$$P^g = \{p_k, 1 <= k <= p\},\ \textit{Ensemble de propriètés d'un groupe g donné.} \tag{4.3}$$

Un point délicat demeure malgré tout dans l'expression de ces règles : comment assurer la dynamique de ces règles fonctionnelles si les agents ne peuvent pas les modifier ? Il serait bien évidement inacceptable qu'un agent puisse modifier une des règles du groupe, les règles n'auront plus de raison d'être, tout agent pourra les modifier selon ses besoins et les contraintes

n'auront plus aucun effet. Cependant, il serait dommage de reporter complètement cet aspect à la phase de l'implémentation. Nous allons chercher à exprimer cela au niveau du modèle organisationnel, via les méta-règles.

Les méta-règles : La structure organisationnelle qu nous proposons étant évolutive, la structure du groupe se trouver instable et dynamique, il faut proposer des solutions permettant de maintenir la cohérence et la stabilité tout en assurant la dynamique des structures, c'est la fonction des méta-règles. Ce sont des règles conçues pour gérer les règles fonctionnelles. En effet, afin d'offrir une dynamique de la structure du groupe, il est primordial que les règles qui l'organisent soient évolutives et souples : c'est une des fonctionnalités des méta-règles. Une règle fonctionnelle qui précise une des conditions d'accès au groupe, peut être mise à jour en fonction de l'état des autres membres du groupe. Etant donné que les modifications sur les règles fonctionnelles sont interdites pour les agents, pour des raisons de sécurité, c'est aux méta-règles de le faire. Comme on le verra ultérieurement, la mise à jour effectuée par ces règles se fait automatiquement dans les différents noyaux accueillant le groupe, et il n'est donc pas nécessaire de déclencher une mise à jour en mode "broadcast".

c) Définition du concept de groupe

Le concept de groupe est modifié afin de répondre aux besoins décelés précédemment. On propose alors de définir le groupe par le tuplet <ID, Cds, Rol, Ser,Ats, Rls> où : 1) ID : L'identifiant du groupe. Chaque groupe a un identifiant unique.

2) Cds : Les conditions d'appartenir au groupe.

3) Rol : La liste des rôles que contient le groupe à un instant donné.

4) Ser : L'ensemble de services offerts par les différents rôles du groupe.

5) At : Les attributs internes du groupe.

6) Rls : L'ensemble de règles qui gèrent le groupe et sa dynamique elles utilisent et agissent sur les conditions et les attributs pour modifier l'état du groupe.

A chaque rôle correspond un type de relation « AgentInRole ». Chaque agent jouant un rôle est lié à une instance du type de relation Agent InRole correspondant à son rôle. La structure du groupe contient les rôles, les services qu'ils offrent mais aussi les relations « AgentInRôle ». C'est le groupe qui cadre l'activité des agents et leurs interactions qui se font au travers de la relation « AgentInRôle ».

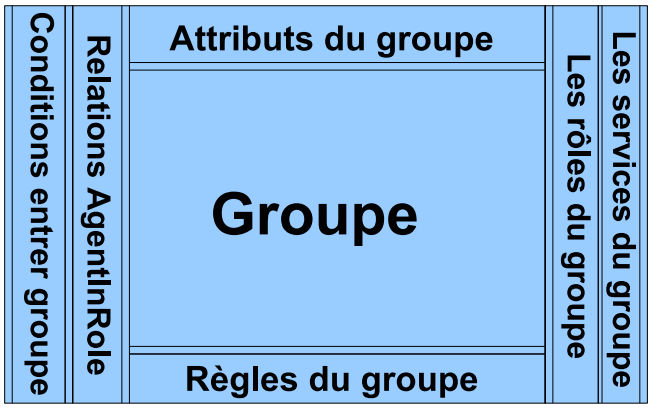

FIG. 4.9 – Architecture du groupe .

4.3 Expression des interactions

Jusqu'ici, nous n'avons réellement évoqué le problème de communication entre agents que du point de vue de l'utilisation des services. Afin de respecter la contrainte de généricité par rapport aux architectures internes des agents, il est primordial de considérer l'interaction indépendamment des langages de communication ou d'imposer un support de communication particulier. Les différentes techniques implémentées sur les systèmes d'agents ou utilisées par les architectures d'agents ne nous intéressent pas à ce stade. Elles peuvent toujours être utilisées en complément à ce que nous proposons au niveau du modèle. Il reste néanmoins important de préciser que toutes les interactions, quelles que soient leurs natures, se font dans le cadre de la relation AgentInRole. Par défaut, tout agent joue le rôle « membre » dans chacun des groupes auxquels il appartient. De cette façon, si un agent reçoit un message, même si ce dernier n'est pas pour demander un service particulier proposé par un rôle précis, l'agent récepteur traitera ce message dans le cadre de sa relation de type AgentInRole qui est (AgentInMembre) correspondant au rôle « membre ».

Nous distinguons deux types de communication : la communication pair à pair et la communication par diffusion.

Communication pair à pair

Dans ce type de communication l'agent émetteur connaît à l'avance l'agent destinataire ou au moins la position qu'il occupe dans l'organisation ou la fonction qu'il doit offrir.

Si l'émetteur connaît l'adresse de l'agent destinataire il le contacte directement sans passer par un rôle précis. Cependant, l'agent récepteur du message, l'exécutera obligatoirement dans l'instance AgentInRole correspondant au rôle qui encapsule le service proposé. Reste à préciser comment régler les interactions entre deux agents jouant un même rôle dans l'objectif de coopérer dans le cadre du rôle joué. Là encore, nous ne sortons pas du cadre de la relation AgentInRole. Si de telles interactions sont possibles, c'est au concepteur d'ajouter les services correspondants aux différentes possibilités d'échanges de messages entre agents jouant le rôle. L'utilisation de ces services peut être restreinte à la condition d'être joueur du rôle. Ces services peuvent aussi ne pas être publiés du tout. Le concepteur du système peut interdire ce genre d'interactions s'il a besoin de contrôler les demandes de services.

Un deuxième cas se présente si l'émetteur ne connaît pas au préalable le destinataire de l'interaction qu'il a initié. Il doit alors passer par une recherche par service, qui lui permettra de déterminer un certain nombre d'agents répondants à son besoin. Il choisira ensuite, selon ses préférences, le rôle qu'il utilisera pour atteindre le but de son interaction. C'est au rôle de mettre l'agent initiateur de la requête en relation avec un agent joueur du rôle et fournissant le (ou les) service qu'il désire avoir. Il est intéressant de noter qu'à ce stade l'adresse de l'agent joueur du rôle n'est pas nécessairement dévoilée. Libre au concepteur d'autoriser ou non l'échange d'adresse entre agent émetteur et agent récepteur de l'interaction.

Communication par diffusion

Il s'agit ici de messages émis par un seul agent à destination d'un ensemble d'agents. Un message peut être envoyé à tout un groupe ou à un ensemble d'agents jouant un certain rôle. L'utilisation de ce type de communication est peu recommandée dans des systèmes ouverts. En effet, le nombre d'agents étant incontrôlable, il peut en résulter des problèmes réseaux (goulots d'étranglements, problème de connexion de certains agents, messages perdus ...). Nous verrons par la suite comment, nous utilisons, avec beaucoup de précautions, ce genre d'interactions au niveau de la gestion des groupes.

4.4 Conclusion

Nous avons présenté un modèle organisationnel de systèmes multi-agents, ouverts et dynamiques, basé sur les concepts d'agent, groupe, rôle et service. Nous avons exposé la problématique en raffinant les besoins et étudiant les contraintes auxquels nous sommes confrontées. Ainsi, deux contraintes se sont imposées. La première contrainte est l'abstraction applicative pour garantir la généricité prônée par notre modèle. La deuxième contrainte est l'opérationnalité. Le modèle proposé doit pouvoir être facilement implémentable avec différentes architectures internes d'agents. Cette étude des contraintes nous a guidé dans notre recherche des principaux axes de notre travail. On l'a bien vu, les agents n'interagissent que pour une raison bien déterminée. Les agents interagissent pour échanger des services et occuper des positions dans l'organisation en fonction de leurs besoins et capacités. Le concept de service devient alors une frontière entre agents mais aussi entre agents et rôles. Nous avons prouvé qu'en utilisant le concept de service, pour exprimer les fonctionnalités d'un rôle, nous pouvons factoriser les agents d'un SMA ouvert tout en cadrant, au niveau organisationnel, les interactions entre agents. Exprimer les fonctionnalités d'un agent et ses besoins en terme de services permet d'expliciter les fonctions d'un rôle, mais aussi de garantir un modèle opératoire indépendamment des architectures internes des agents. Nous avons étudié plus en profondeur la propriété de dynamique, cela nous a permis de distinguer dynamique organisationnelle de la dynamique structurelle. En associant des règles aux groupes et rôles, et en ajoutant des méta-règles, permettant de modifier les règles, nous proposons une solution pour offrir une dynamique au système exprimable à un haut niveau d'abstraction.

On peut finalement voir notre modèle organisationnel comme une vision du monde (au sens informatique), où les agents s'expriment en terme de services. Les agents proposent et cherchent des services ils sont à la fois fournisseurs et consommateurs de services. Les services sont structurés par les rôles qui deviennent alors la frontière entre agents utilisateurs et fournisseurs de services. Les rôles appartiennent à un groupe donné qui fait partie d'une communauté. L'ensemble des communautés fait le monde dans lequel les agents vivent. Inversement un agent cherchant un service dans le monde, se trouve en face d'un ensemble de communautés. Il devra pouvoir chercher quelle communauté englobera le groupe qui contiendra le rôle offrant le service recherché, et qui sera exécuté par un agent jouant ce rôle.

Cette vision du monde des agents que nous appellerons « La vision globale », s'applique au mieux à un environnement comme Internet où les concepts de communauté, groupe, rôle et service trouvent tout leur sens. La figure 4.10 qui suit modélise cette vision.

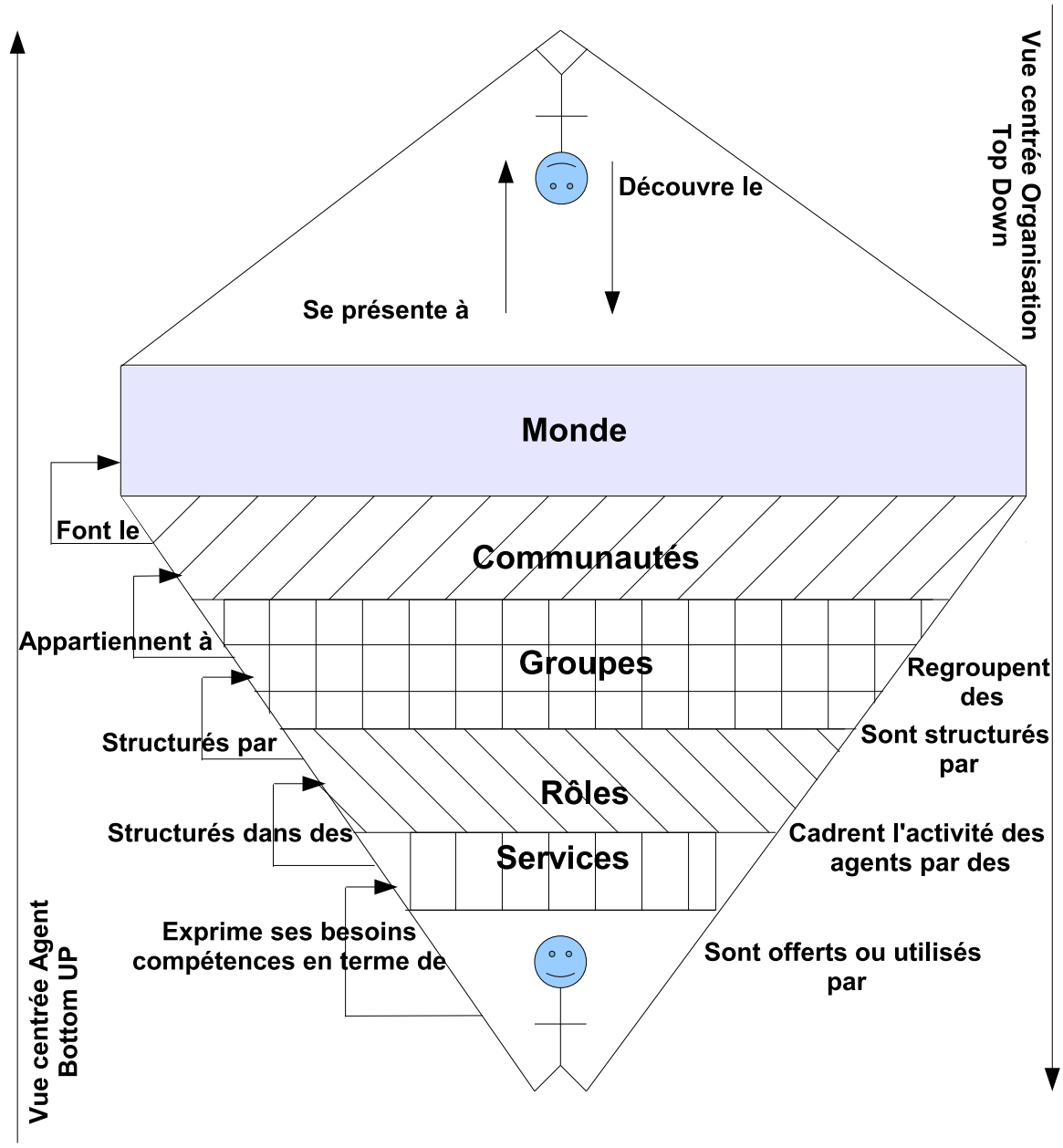

FIG. 4.10 – La vision globale.

Chapitre 5

Gestion décentralisée de groupes distribués

NOUS avons expliqué que la structure et l'activité des agents sont deux faces d'une même médaille. C'est la base de la théorie de structuration selon Higgens [].Les notions d'action et de structure se supposent l'une et l'autre dans une relation dialectique. Les relations des acteurs en co-présence et les structures sociales sont indissociables. Les structures cadrent l'activité des agents et ces derniers créent, modifient et font évoluer la structure. Ce processus est guidé par trois sources. Le contrôle réflexif (conscience pratique), conscience discursive et l'inconscience des agents. Les deux derniers concepts ont trait à la conception interne de l'agent et dépendent donc de l'architecture interne de l'agent.

Notre intérêt porte donc plus sur le premier point : le contrôle réflexif. Nous essayerons de comprendre de quoi il s'agit concrètement dans une organisation d'agents. Ensuite nous prouverons que cette conscience pratique peut être assimilé, dans le cadre des groupes d'agents ouverts, à une mémoire partagée entre agents. Nous définirons alors qu'est ce qu'une mémoire partagée, en présentant les différents mécanismes de gestion d'une telle mémoire. Nous défendrons enfin notre choix de type de gestion, ainsi que les moyens de les modéliser au niveau de l'organisation et leur mise en oeuvre dans la phase de l'implémentation.

5.1 Le contrôle réflexif

L'agent doit contrôler réflexivement son action. Cela revient, en plus de la capacité de pouvoir explique à soi pourquoi il agit, à avoir la capacité de connaître de surveiller, et contrôler le flot continu de la vie sociale et de pouvoir s'y situer continuellement. Le contrôle réflexif n'est donc réalisable que si les agents puissent avoir accès à la structure sociale de l'organisation à laquelle ils appartiennent et être au courant des changements qui la modifient. C'est ce que Giddens [Giddens 84] appelle savoir mutuel. Il s'agit des connaissances partagées par tous les membres et qui est nécessaire au maintien de la cohérence de l'organisation. L'action d'un agent ne peut être correctement interprétée, si les conditions qui la cadrent et le contenu de l'organisation ne sont pas connus par l'agent observant l'action ou la subissant.

Comme Giddens [Giddens84], nous défendons l'idée que la structure existe concrètement indépendamment des agents mais aussi da la mémoire des agents. La structure de l'organisation, ainsi que l'organisation concrète, sont vues et interprétées par chaque agent selon sa position, ses droits, ses capacités et ses besoins. Dans un système ouvert il est prétentieux

de postuler que tous les agents ont la même perception du monde social dans lequel ils sont plongés. Le problème qui se pose est comment assurer un mécanisme permettant à tout agent d'effectuer un contrôle réflexif dans une organisation ouverte. Où est mémorisée l'organisation en dehors de la conscience des agents ? Comment permettre à chaque agent de percevoir son environnement social afin d'effectuer correctement son contrôle réflexif nécessaire pour réussir ses actions dans la société et pour en même temps comprendre l'activité des autres. Dans notre solution le groupe est le cadre de toute activité sociale. Le contrôle réflexif des agents doit se faire sur l'activité au sein du groupe auxquels ils appartiennent. L'agent doit pouvoir observer et rafraîchir continuellement l'activité du groupe et sa composition. Le problème revient donc à assurer une gestion et mise à jour de la structure du groupe et son contenu permettant à tout agent d'avoir une vision correcte du groupe.

Le problème d'assurer le contrôle réflexif, revient à assurer une mise à jour de ce savoir mutuel qu'est la structure du groupe et son contenu chez tous les agents situés dans différents endroits. Cette gestion doit se conformer à la contrainte forte de distribution que nous fixons pour notre modèle.

5.2 Problématique et proposition

5.2.1 Problématique

Revenons à une remarque qui a conclu notre réflexion sur le problème du contrôle réflexif, quand nous résumions ce problème à la mise à jour du savoir mutuel que constituent le groupe et sa structure chez les différents agents membres. Si un changement a lieu dans le groupe les agents membres de ce groupe doivent pouvoir être au courant de ces changements. Certains de ces changements ne sont pas très importants et peuvent être transmis avec un délai de retard, d'autres sont vitaux pour la cohérence du groupe et doivent être transmis en temps réel. Par exemple l'ajout ou la suppression d'un rôle est un évènement majeur et tous les agents doivent le savoir dès qu'il survienne. Les critères déterminants l'importance d'un évènement dépendent des besoins de l'application et des choix des concepteurs.

5.2.2 Les groupes comme mémoire partagée

Au premier coup d'oeil, le problème de gestion des mises à jour des groupes distribués peut passer pour un cas assez complexe et original. Ceci est d'autant plus compliqué qu'il s'agit de mettre à jour autant le contenu du groupe que sa structure. Une autre manière d'approcher la notion de mise à jour du groupe est de voir les règles et la structure comme des simples données faisant partie du contenu du groupe. Le problème se réduit donc à gérer un ensemble de données à partager entre différents membres. On aboutit donc à un problème de gestion de mémoire partagée. Cette mémoire contiendra la structure du groupe et son contenu.

L'intérêt de ce point de vue, est de tirer profil des travaux qui ont été réalisés dans la gestion des connaissances partagées et plus particulièrement des mémoires coopératives. Nous adopterons ensuite la meilleure solution en l'adaptant à notre problématique des groupes ouverts, dynamiques et distribués. Ce problème de gestion des connaissances est un problème complexe et a été traité par plusieurs disciplines : Sociologie, économie, finance... Nous définirons, dans le prochain paragraphe, les objectifs de la gestion des connaissances partagées dans un cadre informatique. Nous focalisons notre étude surtout sur la gestion organisationnelle des connaissances d'un groupe.

5.3 La distribution des mémoires partagées

Les objectifs de la gestion de connaissances : K.M (ou (Knowledge Management)), est de promouvoir l'évolution, la communication et la préservation des connaissances au sein de l'organisation [Steels, 1993]. Il faut faire parvenir les bonnes connaissances au bon membre au bon moment.

La réalisation d'une mémoire organisationnelle se base sur six étapes selon [R.Dieng & al 98] : 1) Détection des besoins 2) Construction de la mémoire organisationnelle 3) Diffusion de la mémoire organisationnelle 4) L'utilisation de le mémoire organisationnelle 5) L'évaluation de la mémoire organisationnelle 6) Maintenance et évolution de la mémoire organisationnelle.

C'est la phase de diffusion qui nous intéresse le plus dans ce paragraphe. Les autres parties processus font partie de la spécification de la structure de groupe que nous avons décrit ultérieurement (étape 1-2-6), ou de la compétence de chaque agent (étape 4 et 5).

Nous distinguons dans la littérature quatre types de gestion de mémoire partagée, référencées selon deux axes : Le maintien de l'information partagée et le mode de distribution. Ces deux axes sont divisés selon deux modes passifs et actifs. Si c'est la mémoire qui effectue le maintien et l'alimentation des connaissances qu'elle contient, on parle de gestion active des données. Si c'est aux agents de l'alimenter et d'assurer son maintien on parle de gestion passive. Quand les agents consultent la mémoire pour récupérer les nouvelles connaissances, et rafraîchir ainsi leurs perceptions des connaissances de l'organisation dans laquelle ils appartiennent, la diffusion est passive. Si la mémoire informe les membres concernés (ou tous les membres) des nouvelles connaissances la constituant, on dit que la distribution est active. Les actions de la mémoire, quand elle est active, peuvent être assurées par des processus internes à la mémoire. La mémoire n'est plus vue comme simple dépôt de connaissances mais comme un vrai processus dynamique. Une autre approche pour réaliser l'activité est de déléguer à des responsables l'ensemble des activités (mise à jour ou alimentation).

Voyons comment [Van Heijst et al., 1996] les catégorisent à partir de ces deux axes (distribution/gestion) et ces deux modes (actif/passif) :

Tab. 5.1 – Les différents types de gestion de mémoire partagée

	Gestion passive des données	Gestion active des données
Distribution passive	Knowledge attic	Knowledge sponge
Distribution active	Knowledge publisher	Knowledge pump

Le knowledge attic : C'est une mémoire utilisée comme un archive qui peut être consulté quand les membres de l'organisation le souhaitent et quand ils en ont besoin. Elle n'est pas intrusive mais nécessite une très grande implication et discipline des membres pour éviter de la rendre obsolète. En effet, sans les demandes et sollicitations des membres pour s'informer de l'état de la mémoire, son fonctionnement n'aura pas de sens. Cela implique des mécanismes spécifiques de synchronisations afin d'assurer le maintien d'une vue d'ensemble correcte. D'autre part, des mécanismes de mis à jours adéquats doivent être mis en place afin d'assurer un minimum de cohérence et de crédibilité de l'état global de la mémoire pour les membres.

Le knowledge sponge : C'est une mémoire organisationnelle régulièrement alimentée pour la maintenir assez complète. Son utilisation est laissée à la charge des membres de l'organisation. L'organisation tente de construire une mémoire plus au moins complète. Par

contre, ça reste aux utilisateurs de consulter et récupérer les connaissances mises à jours dans la mémoire et les utiliser après selon leurs besoins.

Le knowledge publisher : C'est une mémoire où la contribution est laissée à la charge des différents membres de l'organisation. En revanche, les processus de la mémoire analysent la connaissance entrante, la combinent avec celle déjà stockée et diffusent les informations intéressantes aux membres potentiellement intéressés par ces modifications.

Le knowledge pump : C'est une mémoire de corporation qui garantie que la connaissance développée dans l'organisation soit correctement reçue et utilisée par les différents membres de l'organisation. L'alimentation de la mémoire avec les nouvelles connaissances n'est pas à la charge des membres. C'est la mémoire, par le biais de ses processus ou à travers ses responsables, qui garantit la mise à jour de la mémoire en ajoutant de nouvelles connaissances. La diffusion des connaissances fraîchement acquises est aussi la responsabilité de la mémoire. Les membres, reçoivent périodiquement des informations mettant à jour la perception qu'ils ont de la mémoire organisationnelle.

Nous situons la gestion des groupes dans notre modèle dans le cadre du (knowledge pump), nous souhaitons que tous les agents soient informés continuellement de l'évolution du groupe dans lequel ils agissent. D'autre part, nous ne souhaitons pas une intervention des agents dans l'activité d'enrichissement de la composition du groupe. Cela impliquera des suppositions sur le raisonnement interne des agents. Nous devons donc, doter le groupe de mécanismes permettant une gestion et distribution active. Ce qui nous intéresse particulièrement dans cette approche est avant tout l'idée d'autogestion de la mémoire que nous confondons avec le groupe (structure et contenu). Nous restons fidèles à notre abstraction par rapport aux architectures internes des agents tout en leur assurant un contrôle réflexif nécessaire pour la réalisation de leurs activités.

5.4 La gestion des groupes au niveau de l'organisation

Bien qu'il soit à cheval entre la phase de conception et celle de l'implémentation, le problème de la gestion des systèmes multi-agents et des groupes d'agents en particulier est souvent traité au niveau de l'implémentation. Considérer ce problème de gestion uniquement de cet angle purement implémentatoire, complique la tâche du concepteur et permet d'avoir un panaché de solutions aussi différentes les unes que les autres et souvent peu efficaces. Nous proposons ici une solution, située au niveau du modèle, permettant de gérer les groupes ouverts et dynamiques de façon décentralisée. Nous tenons à rester cohérent avec les hypothèses de ce modèle, nous ferons abstraction des architectures d'agents ainsi que des types d'applications qui peuvent se baser sur notre modèle.

Après avoir présenté brièvement les différents types de mémoires et situé la gestion du groupe d'agents dans notre modèle dans le cadre du « Knowledge Pump », nous allons proposer quelques solutions pour prendre en compte les deux axes du « Knowledge Pump » : gestion active et distribution active. En premier temps nous expliquons comment exprimer ces deux aspects au niveau organisationnel, avant de présenter quelques pistes d'implémentation.

5.4.1 Représentation

Nous définissons le groupe comme une sorte particulière de mémoire partagée de type « Knowledg Pump ». Au lieu d'avoir une seule représentation concrète du groupe dans laquelle

se font tous les enrichissement et le maintien des connaissances de l'organisation, on définit un groupe comme un ensemble de copies. Chaque copie du groupe est une représentation de la structure et contenu du groupe dans une plateforme contenant un ou plusieurs agents membres de ce groupe. La solution consiste en fait à se reposer sur le groupe lui-même : le groupe est vu à la fois comme le contexte d'activité des agents et le miroir reflétant cette activité à tous ses membres.

Notre hypothèse sera donc de supposer que chaque copie de groupe mémorise les actions des agents, qui lui sont rattachés, et les effets de ces actions sur la structure. Elle évalue ensuite l'impacte de ces effets et juge leurs importances. Si un fait, ou l'accumulation de faits, sont jugés importants la copie diffuse la nouvelle version vers les autres copies du groupe. L'ensemble des agents peuvent à cet instant avoir la même vue du groupe quelque soit leur emplacement physiquement. Un groupe k est défini par l'ensemble de copies qui le forment : Gk = Gk1,...,Gkn.

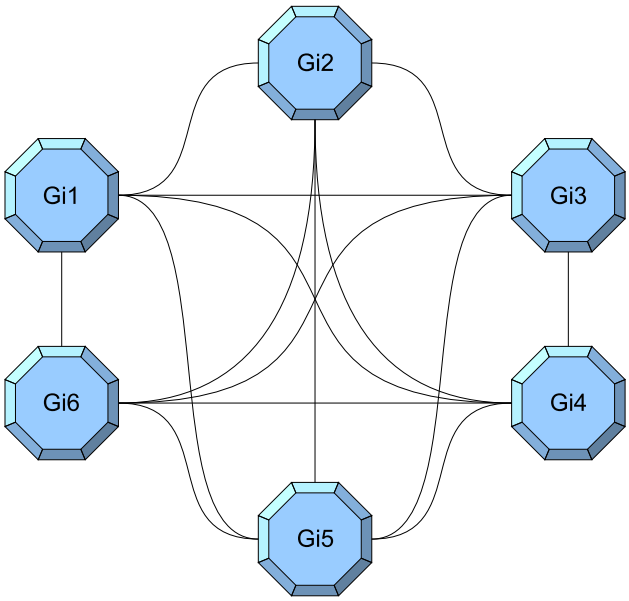

FIG. 5.1 – Exemple de groupe distribué sur six noyaux différents .

On expliquera, dans la partie implémentation, comment les copies sont échangées entre noyaux contenant une copie du groupe et les nouveaux noyaux qui possèdent des agents désirant appartenir au groupe.

On définit également l'état E d'un groupe comme le dernier état mis à jour et partagé par tous les membres. En effet, chaque modification au niveau du contenu du groupe ou sa structure est enregistrée localement. Par conséquent, les copies Gi ne sont pas forcément identiques à un instant t.

5.4.2 Relation d'équivalence

Nous distinguons le concept « copie correcte » de celui de « copie exacte » par rapport à l'état E. Le processus de distribution ne se déclenche pas à chaque ajout d'une information au groupe. Cela signifie que les copies sont supposées être identiques, dès lors qu'il n'y a pas eu une modification pertinente qui nécessite une mise à jour de la vue globale du groupe. Deux agents situés dans deux plateformes différentes peuvent ne pas avoir la même valeur d'un

attribut d'une entité donnée. Il s'agit sans doute d'une modification qui a eu lieu sur l'entité sur l'un des deux sites, et que cette modification n'a pas été propagée, vu qu'elle n'est pas considérée comme majeure. La copie recevant une mise à jour, informe ses agents membres de l'actualisation de son état. Après, c'est aux agents de redéfinir leurs perceptions du groupe.

Une copie Gi est équivalente à l'état E si et seulement si, aucun changement sur E, mémorisé dans Gi, ne nécessite une mise à jour, selon les règles du groupe G. On peut dire que la copie Gi est correcte vis-à-vis de E. Cela signifie qu'à tout instant toutes les copies sont correctes par rapport à l'état E et sont donc correctes entre eux.

$$\textit{A tout instant } t, \textit{ pour un groupe } G^k \textit{ donné}, \ \forall i, \ G_i^k \equiv E \tag{5.1}$$

$$\textit{A tout instant } t, \textit{ pour un groupe } G^k \textit{ donné}, \ \forall i, \ \forall j, \ G_i^k \equiv G_j^k \tag{5.2}$$

La relation d'équivalence entre copies n'a pas de sens en dehors d'un groupe donné. L'équivalence n'est vérifiable qu'à partir des règles du groupe. Deux groupes différents n'ont pas nécessairement les mêmes règles, il est donc impossible d'appliquer la relation d'équivalence entre les copies de deux groupes différents.

5.4.3 Distribution par évènements

Nous avons déjà définit la notion de règles d'un groupe. Nous avons expliqué que c'est aux règles d'assurer la dynamique du groupe. Ce que nous cherchons ici, c'est de permettre au groupe de juger de l'importance d'une action survenue à un instant donné. Chaque comportement des agents est donc mémorisé et évalué par le groupe. Les règles décident, selon l'état général de la copie du groupe, de diffuser les changements qui ont eu lieu depuis la dernière mise à jour reçu par la copie (l'état E).

Ev(G) = L'ensemble des événements qui nécessitent une mise à jour de la copie du groupe

La diffusion peut se faire, selon le type de mise à jour, par deux modes. Soit la diffusion se fait sur toute la copie du groupe, soit cette dernière effectue la différence entre l'état E et la nouvelle copie. Tout dépend donc, du choix du concepteur dans la mise en oeuvre des règles qu'il précise pour la mise à jour.

Une règle de diffusion peut donc prendre, selon le type de mise à jour, une de ces deux formes :

$$Si\ Gi \neq E \Rightarrow (Diffuser(Gi \ominus E)) \wedge (Affect(E, Gi)) \tag{5.3}$$

$$Si\ Gi \neq E \Rightarrow (Diffuser(Gi)) \wedge (Affecter(E, Gi)) \tag{5.4}$$

5.4.4 Conclusion

On a vu qu'il est possible d'exprimer la gestion décentralisée des groupes distribués au niveau du modèle organisationnel. De plus on a pu situer notre solution vis-à-vis des différents modes de gestion de connaissances. Le mode de gestion proposée se repose sur deux axes : une alimentation active du groupe, et une distribution active de l'état du groupe aux différents

membres. Notre objectif est de trouver un point médian entre la gestion déléguée totalement à l'implémentation et une gestion exprimée au niveau organisationnel et facilement mise en oeuvre au niveau de l'implémentation. Pour cela, nous reposons sur le groupe lui-même pour exprimer, à partir de sa description au niveau organisationnel, les mécanismes de gestion et de distribution.

Notre modèle s'inspire du modèle « Knowledge Pump » où la gestion de la mémoire est active et la distribution est aussi active. Le groupe est considéré comme un cas particulier de mémoire partagée. Nous définissons le groupe comme un ensemble de copies distribuées sur les différents plateformes accueillant des agents appartenant à ce groupe. La structure du groupe (ensemble de rôles) ainsi que son contenu (règles instances des rôles, relation AgentInRole...) sont dupliqués.

Nous avons évoqué la distribution et la diffusion pour la mise à jour sans préciser comment cela est possible au niveau de l'implémentation. Nous avons décrit dans le chapitre 3, le concept de service et les concepts de recherche et publications de service et de la description d'un groupe ou d'un rôle, sans proposer une solution pour mettre en oeuvre ces principes. Nous essayons dans la partie deux de proposer une solution basée sur la technologie P2P pour réaliser la communication entre copies d'un groupe, les communications entre agents, la publication et recherche de services... Notre solution sera une architecture hybride : Multi-agent/P2P qui sera présenté au chapitre 7 avec une proposition d'implémentation du modèle AGRS. Le chapitre 6 sera consacré à la définition des concepts P2P et en particulier sur la plateforme JXTA.

Chapitre 6

Les technologies P2P : JXTA une solution pour le déploiement à grande échelle

L'idée L'idée d'utiliser les principes et les technologies P2P [Wai-Sing Loo, 2007] pour résoudre certains problèmes des systèmes mulit-agents ouverts n'est pas récente. Déjà en 1998, Sycara dans [Sycara, 1998] décelait le besoin de l'utilisation de la technologie P2P en association avec les systèmes multi-agents pour pallier aux problèmes dus à l'hétérogénéité, l'ouverture et la dynamique. Elle énonçait dans ce sens : « Actuellement, la grande majorité des systèmes multi-agents consistent à un seul agent. Cependant, comme la technologie agent devient mature et traite des applications de plus en plus complexes, le besoin d'un système composé d'agents qui communiquent dans un mode P2P devient apparent. » Dans ce chapitre nous allons tout d'abord présenter le paradigme Peer-to-Peer (P2P, pair-à-pair ou égal-à-égal) et les différents modèles existants. Notre classification se portera sur les modèles purs et les modèles hybrides. Les modèles hybrides se répartissent selon le type d'indexation : centralisée ou décentralisée. Nous exposerons ensuite deux systèmes P2P qui ont connu un très large succès et qui ont fait connaître la technologie P2P au grand public : Genutella [Ganutella, n.d.] qui se conforme aux spécifications des architectures pures et Napster qui est l'exemple de ce qu'on peut qualifier de système P2P hybride. Enfin, nous présenterons en détail la plateforme JXTA [JXTA, n.d.] que nous utiliserons, dans la suite de notre travail, pour combiner les concepts P2P avec les propriétés des agents et proposer ainsi l'architecture hybride Madkit-JXTA implémentant notre modèle AGRS.

6.1 Les différents modèles P2P existants

6.1.1 Définition

Une architecture réseau distribuée est dite P2P, ou égal-à-égal comme certains préfèrent l'appeler, si tous les participants sont égaux ou plus précisément s'ils ont des responsabilités et capacités équivalentes et partagent une partie de leurs ressources (ressources matérielle, logiciels, physique ou virtuelle). Ces ressources partagées sont directement accessibles par les autres pairs, sans besoin de passer par des entités intermédiaires. Le principe d'égalité entre les participants à de tels systèmes se justifie par le fait que ces pairs sont à la fois

demandeurs et fournisseurs de ces ressources. L'échange de ces ressources se fait sous forme de services [Schollmeier, 2002]. On retrouve ici le concept de service et son utilisation dans le modèle AGRS que nous proposons. Il faut avouer que nous nous sommes beaucoup inspiré des travaux du domaine des P2P pour concevoir et bâtir notre modèle.

A l'opposé du paradigme Client-Serveur, le paradigme P2P est basé sur le concept que chaque composant du système peut à la fois être client et serveur. Les systèmes P2P se distinguent par les protocoles de connexion utilisés, la manière d'organiser les pairs, les mécanismes de recherche et de publication et les relations entre les différents pairs [Nejdl *et al.* , 2002]. Nous distinguons essentiellement deux différents types de classification de catégories mettant en oeuvre ce principe [Schollmeier, 1998]. On peut qualifier une architecture P2P de pure ou hybride.

Remarque *On peut aussi catégoriser les architectures P2P en systèmes structurés ou non structurés.*

6.1.2 Les architectures P2P pures

Un réseau P2P est qualifié de pur si c'est un réseau obéissant à la définition d'une architecture P2P, et qui en plus permet d'enlever arbitrairement n'importe quelle entité, faisant partie du réseau, sans pour autant affecter la qualité ni les services offerts par le réseau [Schollmeier, 2002]. En d'autres termes, aucune entité n'est indispensable au bon fonctionnement du système. Cela revient à dire que le système doit être totalement décentralisé que ce soit pour la publication et recherche des services ou pour l'acheminement des messages d'un pair à un autre. Cela produit des systèmes très autonomes, peu liés et dans la plupart du temps sans aucune structure hiérarchique pour éviter la centralisation ou la suprématie de certains noeuds sur d'autres. Cette équivalence parfaite des pairs offre une grande scalabilité au système et garantit une tolérance aux pannes. Cependant, de tels systèmes ne permettent pas une recherche et découverte rapide des ressources disponibles sur le réseau. L'absence d'une vue globale du système rend difficile la prédiction de l'attitude globale du système et son contrôle. De même, il n'y a aucune garantie concernant la qualité des services offerts par l'ensemble des pairs. Afin de pallier à ces défauts des architectures pures, de nouvelles architectures dites « hybrides » ont vu le jour.

6.1.3 Les architectures P2P hybrides

Dans les architectures hybrides, il y a un serveur central qui maintient, sous formes de meta-données, des répertoires d'informations concernant les pairs enregistrés sur le réseau [Pourebrahimi *et al.* , 2005]. Deux approches majeures existent pour mettre en oeuvre une telle architecture hybride : Les approches d'indexation centralisées et les approches d'indexation décentralisées.

Indexation centralisée :

Dans une telle technique, un serveur central maintien un indexe des ressources partagées par les pairs actifs du réseau. Chaque pair maintient une connexion directe avec le serveur central qui se charge de l'envoi des requêtes. Une telle approche assure une rapidité d'exécution des recherches (le serveur indexe efficacement les ressources), et garantie une qualité des réponses fournies aux requêtes des pairs demandeurs.

Cependant, on se retrouve avec les défauts des techniques centralisées. Le système est vulnérable et le risque de panne du serveur central est élevé et peut mettre la viabilité du système en danger. D'autre part, les tableaux d'indexes ne peuvent pas être de tailles illimitées, la propriété de scalabilité n'est plus assurée. Les systèmes à indexation décentralisée sont créés pour combler ces manques.

Indexation décentralisée :

Dans les approches d'indexation décentralisée, certains pairs ont plus d'importance que d'autres, on les appelle les Super-Peers [Nejdl *et al.* , 2004]. Ils maintiennent les indexes centralisés des ressources partagées par les différents pairs connectés. Ils reçoivent les requêtes des pairs et les dispatchent vers les différents fournisseurs potentiels. Les Super-Peer forment un réseau parallèle au réseau global et échangent des messages pour synchroniser leurs tables d'indexes ou faciliter les recherches. Si un de ces Super-Peer disparaît du réseau ou tombe en panne, les pairs qui lui sont connectés pourront se connecter à un des autres Super-Peer présents et le système demeure opérationnel. Souvent le choix des Super-Peer se fait selon leur capacités de stockage, de calcul et la largeur de bande de leurs connexions. Un mécanisme permettant de désigner de nouveaux Super-Peer si leur nombre devient inférieur à un certain seuil critique, peut être mis en place afin de garantir la fiabilité et l'opérationnalité du système.

Par rapport aux architectures pures, le temps de découverte est réduit grâce à la communication et l'indexation entre les Super-Peers. Le nombre de messages échangés, pour la recherche et la découverte, est aussi réduit puisqu'il ne concerne que les Super-Peers et pas l'ensemble de pairs du réseau. Comparé à la solution d'indexation centralisée, ces modèles évitent la dépendance à un serveur central et réduisent par conséquence donc la vulnérabilité du réseau. Cependant, la vitesse de recherche est inférieure à celle des solutions d'indexation centralisée.

Dans le paragraphe suivant nous présentons Genutella [Ganutella, n.d.] et Napster [Napster, n.d.]qui implémentent l'un le modèle pur et l'autre le modèle hybride.

6.2 Etude d'architectures Peer-to-Peer existantes

Il existe plusieurs modèles d'applications Peer-to-Peer qui sont plus au moins conformes aux règles de base du concept. Nous exposerons en premier lieu les applications les plus connus et qui ont fait le succès de la technologie pour le grand public à travers d'Internet, avant de présenter dans le prochain paragraphe plus en détails la plateforme JXTA [Gradecki, 2002] et les différents protocoles qui la constituent.

6.2.1 Gnutella : Un modèle d'architecture P2P pure

Cette application permet l'échange de fichiers sur le réseau Internet en mode peer-to-peer. Chaque utilisateur étant représenté par un acteur qui aura la tâche de lui trouver les ressources qu'il désire. L'acteur rejoint une communauté pour pouvoir atteindre son objectif de recherche du fichier que l'utilisateur désire. L'entrée de l'acteur à la communauté implique qu'il devra respecter les règles qui la régissent, il s'agit de se comporter comme tous les autres acteurs : être client et serveur en même temps. Pour chercher une ressource, un pair s'adresse à ses pairs voisins qui à leur tour, s'ils n'ont pas de réponse, diffuseront la requête à leurs voisins et ainsi

de suite. Le réseau n'a donc aucune topologie prédéfinie à l'avance : on parle de système P2P non structuré. Les mécanismes de recherche produisent des lots de messages incontrôlables et peuvent bloquer le système en faisant chuter la bande passante existante. Tous les acteurs de la communauté sont symétriquement dépendants : il n'y a pas d'acteurs favorisés par rapport à d'autres et tous s'offrent mutuellement le service d'exploration et de recherche de fichiers. En plus, chaque acteur peut à tout moment quitter la communauté, sans pour autant que cela n'affecte le fonctionnement des autres acteurs pairs. La communauté est donc indépendante de ses acteurs adhérents. Plusieurs modifications ont été apportées à la version initiale de Gnutella, pour combler certaines lacunes, en introduisant le concept de SuperPeer. A partir de cet exemple nous pouvons dégager deux principes importants. La notion de communauté et celle d'égalité entre pairs.

6.2.2 Le modèle Napster : Un modèle d'architecture hybride

Napster s'appuie sur les mêmes principes de base de Gnutella en ce qui concerne la représentation des utilisateurs par des pairs, qui auront pour buts la recherche et le téléchargement de fichiers. La seule différence réside dans le fait que les acteurs doivent consulter un acteur particulier, afin de connaître les services offerts par les autres pairs. Cet acteur est le serveur Napster qui a besoin des autres acteurs pour publier leurs services. La dépendance symétrique entre acteurs est maintenue dans la mesure où tous les acteurs, à part le serveur, doivent publier la liste de leurs ressources chez le serveur, et que tous peuvent consulter cette liste en le contactant. La notion de communauté est aussi maintenue puisque tous les acteurs sont des clients Napster, qui est considéré comme une communauté d'acteurs en peer-to-peer. L'inconvénient de la centralisation reste le plus grand défaut et la panne du serveur Napster produit à la chute de tout le système. Nous constatons que ce modèle respecte les deux principes que nous avons dégagés préalablement à savoir, l'existence d'une communauté de pairs d'acteurs qui sont symétriquement dépendants et sont tous au même pied d'égalité. Le seul inconvénient c'est l'existence d'un serveur central. En effet, le fonctionnement de toute la communauté dépendra de l'existence et la fiabilité de ce serveur ce qui fait la différence avec le modèle Gnutella. C'est certainement pour cette raison que Gnutella est qualifié d'application P2P pure ce qui n'est pas le cas de Napster, qualifié de système hybride.

6.2.3 Conclusion

Essayons maintenant de fixer quelques critères et règles pour que des acteurs soient qualifiés de pairs (cad en situation de peer-to-peer). En premier lieu, chaque acteur doit avoir des buts à atteindre. Dans le cas du partage de fichier, ces objectifs sont la recherche et le téléchargement de fichiers. L'acteur n'est pas capable d'atteindre ses objectifs tout seul et il a besoin de l'aide d'autres acteurs qui ont globalement les mêmes buts. Chaque acteur doit accepter que les autres obtiennent de lui ce qu'il leurs demande comme services et vice versa. Dans le cas de partage de fichier, les services d'exploration sont offerts par chaque acteur. L'acteur est autonome, dans la mesure où il spécifie ses préférences et ses contraintes quant à l'accès à ses ressources et ses services offerts. Les règles que doivent obéir les acteurs sont celles d'une certaine communauté, à laquelle ils doivent appartenir. Ainsi, une communauté d'acteurs en peer-to-peer n'est rien d'autres qu'une organisation virtuelle, régie par des lois et des règles, et qui a pour objectif de permettre à ses membres (les acteurs pairs) de partager certains services, en respectant l'égalité entre eux. La communauté doit être dynamique dans le sens où ses membres et leurs services changent, apparaissent et disparaissent continuelle-

ment. Pour être qualifié de pur, le système doit être capable de fonctionner convenablement indépendamment des membres qui le forment (ne dépend pas d'un acteurs précis). Les notions d'autonomie, d'organisation régie par des règles, d'égalité entre membres (fournisseur et demandeur), recherche et publication de services et surtout l'opérationnalité du système indépendamment des membres du système, sont des concepts qui font le modèle AGRS et qui sont le coeur des modèles P2P. Là encore, on constate l'influence qu'ont eu sur notre travail, les grandes lignes et concepts des modèles P2P. C'est de cette manière que nous avons choisi d'aller jusqu'au bout, c'est-à-dire jusqu'au stade de l'implémentation de notre modèle. Nous avons donc fait le choix de composition entre les concepts de deux technologies que nous considérons complémentaires : les systèmes multi-agents et les systèmes P2P. Nous présentons, dans le paragraphe suivant, la plateforme JXTA qu'on utilisera pour réaliser l'architecture hybride mettant en oeuvre le modèle AGRS.

6.3 Présentation de JXTA

6.3.1 Introduction

JXTA [JXTA, n.d.] est un middelware pour des applications en peer-to-peer développé au sein du modèle Apache par Sun Microsystems. L'origine du nom JXTA provient du verbe « juxtaposer », c'est à dire le fait de mettre deux entités côte à côte. L'idée de juxtaposition vient du fait que l'équipe de développement croit que les applications peer-to-peer vont coexister avec leurs ancêtres : les applications client-serveur. Les systèmes P2P ne vont pas remplacer les systèmes client-serveur, selon le type d'application et le problème posé, les développeurs choisiront l'une ou l'autre des architectures.

L'objectif de JXA est de créer une plateforme offrant des fonctionnalités assez standard pour permettre aux développeurs, d'horizons distincts, de pouvoir créer des applications peer-to-peer inter opérables. Afin d'atteindre cet objectif général, des sous buts spécifiques doivent être atteints. En effet, la plateforme doit en premier lieu offrir une interopérabilité entre les différents systèmes P2P d'une part, et entre ces systèmes et toute autre application (en particulier celles liés à Internet WebService, Grille...) d'autre part. En second lieu, la plateforme JXTA se doit d'être indépendant vis à vis des systèmes, des applications, des protocoles et des langages de programmations qui l'utilisent. Sans une telle indépendance, la plateforme ne pourra pas être massivement utilisée et finira par disparaître.

Le dernier point qui est sans doute le plus important dans ce genre d'application, est la sécurité. JXTA doit offrir un niveau de sécurité très élevé pour ses usagers. En réalité, pour les applications du type P2P tous les noeuds sur le réseau sont égaux, et on ne peut différencier les « gentils » des « méchants ». Il est donc primordial de prendre en compte l'aspect sécurité et de le traiter au plus bas niveau (noyau JXTA).

Ces objectifs ont été décortiqués en six protocoles, qui forment les services de base de la plateforme JXTA. L'implémentation de ses protocoles se fait dans un niveau plus bas et peut être réalisé en n'importe quel langage, respectant ainsi l'indépendance de la plateforme vis à vis des systèmes et des langages de programmation. Dans la suite des paragraphes, nous définissons les concepts clés de JXTA. Nous présentons ensuite, le modèle conceptuel et les principaux protocoles qui le forment, ainsi que l'architecture générale de la plateforme. L'objectif de ce paragraphe est de présenter le sens de quelques termes partagés par la plupart des applications et architectures P2P. Nous définissons ces termes du point de vue de l'application JXTA. Le premier concept à définir est sans aucun doute celui du « peer » ou pair. Nous

expliquerons qu'est ce qu'un pair au sens JXTA. Nous définirons ensuite ce qu'est selon JXTA un « peerGroup », « Pipe », « Message » et « Advertisment ».

6.3.2 Peer

Définition

Un pair JXTA est toute entité matérielle ou logicielle implémentant les protocoles nécessaires de JXTA permettant son adressage et identification sur le réseau. Tout pair est considéré comme étant un noeud d'un réseau formé d'autres pairs. Il faut cependant distinguer entre noeud et ordinateur, car un pair peut aussi être un simple téléphone portable ou un PDA contenant un programme qui implémente les protocoles de bases de JXTA. D'autre part, une application répartie sur plusieurs machines et qui implémente les protocoles basiques de JXTA est un pair. Enfin un système hôte peut accueillir plusieurs pairs.

Un pair peut partager, mais ce n'est pas une obligation, des ressources avec d'autres pairs au sein d'un même groupe. Un pair qui implémente la spécification d'un service, pourra l'offrir aux autres. Des pairs, qui implémentent la même spécification d'un service, sont interchangeables pour un pair qui utilise le service, abstraction faite sur le langage utilisé pour sa réalisation.

C'est de là que vient la notion d'utilisation des services dans le modèle AGRS. Rappelons qu'un agent utilisateur d'un service, décrit dans un rôle, ne distingue pas quel agent joueur du rôle va lui fournir le service en question. Les agents joueurs d'un rôle et fournisseurs d'un service donné sont interchangeables pour l'agent utilisateur du rôle et demandant le service. On peut dire que les agents du modèle AGRS sont en quelque sorte des Peer-Agent : ils sont à la fois fournisseurs de services qu'ils décrivent au travers des rôles et en même temps utilisateurs des services des autres agents.

Nous distinguons trois types de pairs dans la plateforme JXTA : les pairs simples, les pairs rendez-vous et les pairs routeurs.

Le pair simple

C'est un pair conçu pour servir un seul utilisateur, lui permettant d'utiliser des services offerts par d'autres pairs ou de proposer et publier ses propres services. Dans les environnements actuels et par soucis de sécurité, ces pairs sont confrontés à plusieurs types de protections qui constituent des contraintes pour le pair de deux points de vues.

Si le pair simple se trouve dans réseau isolé de l'extérieur par (sous la protection d'un FireWall, derrière un Proxy ou un NAT, dans un réseau privé...), il ne sera pas visible de l'extérieur, ce qui pause évidemment un gros problème pour la diffusion et l'utilisation massive des services qu'il fournit au monde extérieur. Même si le pair n'est pas protégé et a un accès libre au monde extérieur cela peut ne pas être le cas des autres pairs et il se trouve donc privé des services qu'ils peuvent offrir. Le problème de visibilité et accessibilité des pairs, que ce soit de ou vers l'extérieur, est un défi majeur pour les applications ouvertes et déployés à grande échelle, comme celles basés sur des architectures P2P.

Afin de résoudre ces problèmes majeurs pour les applications P2P déployés dans des environnements ouverts tel que Internet ou même sur des grands réseaux privés, JXTA propose deux autres types de pairs : le pair routeur et le pair rendez vous.

Le pair rendez-vous

Un pair rendez-vous se distingue des autres pairs simples par le fait d'offrir un service particulier. En réalité, ce pair permet la découverte des pairs se trouvant sous sa responsabilité et les ressources qu'ils offrent. Il reçoit des requêtes envoyées par certains pairs, et se charge de renvoyer des réponses en fonction des informations qu'il possède sur les pairs qu'il connaît. Les différents pairs rendez-vous d'un même réseau peuvent collaborer en s'échangeant des informations, et en diffusant les requêtes auxquels ils ne peuvent pas répondre. Deux cas de figure sont envisageables pour la localisation du pair rendez-vous par rapport à un réseau local qu'il représente. Le premier cas est le plus classique, c'est qu'il est à l'extérieur du réseau et donc il sera visible pour les autres pairs. La deuxième possibilité, c'est que le pair rendez-vous se trouve à l'intérieur du réseau et dans ce cas il doit, soit avoir les privilèges nécessaires pour qu'il puisse outrepasser les protections du réseau, soit utiliser un pair spécial : le pair routeur que nous présentons dans le paragraphe suivant.

Le pair routeur (relaie)

Un pair de type routeur permet à des pairs, séparés du réseau externe par mécanismes de protection (FireWall, NAT...), de communiquer avec les autres pairs. Un pair qui veut communiquer avec un autre pair isolé doit, en premier lieu, déterminer quel pair routeur il doit utiliser. Ensuite, c'est au routeur de trouver l'adresse IP dynamique relative au pair destinataire. Le mécanisme ressemble par ces principes au DNS utilisé traditionnellement. Les détails de mise en oeuvre des mécanismes du pair-routeur dépassent les objectifs de ce travail.

Notons enfin qu'un simple pair peut jouer le rôle de rendez-vous ou de routeur ou les deux simultanément. Il suffit d'implémenter les services en question pour devenir un pair rendez-vous ou un routeur.

Nous expliquons dans le paragraphe suivant la notion de Peer-Group pour JXTA.

6.3.3 Peer Group (groupe de pairs)

Définition

Un Peer Group ou (groupe de pairs) est une collection de pairs qui se sont mis d'accord pour le partage d'un certain nombre de services (échange de fichiers, résolution de problèmes...). A chaque groupe est attribué un identifiant unique. Deux étapes, au minimum, sont nécessaires pour la formation d'un groupe. La première est la création d'un processus d'inscription pour les pairs qui seront membres du groupe. Un mécanisme de protection de type (nom, mot de passe) peut être utilisé pour réaliser cette étape, sinon le groupe est ouvert à tout membre désirant le rejoindre. Si un pair crée un groupe et lui rattache des mécanisme de sécurité restreignant son accès, le groupe est alors considéré comme privé. La deuxième condition nécessaire est que tous les pairs puissent communiquer selon le même type de messages et se basent sur les mêmes protocoles de communication.

Un pair peut appartenir à plusieurs groupes en même temps, et tous les pairs dès leurs créations appartiennent automatiquement à un groupe par défaut dit « World Peer Group ». Ce groupe est l'ancêtre de tous les autres groupes qui peuvent être créés. Un deuxième groupe par défaut existe en JXTA il s'agit du « Net Peer Group ». Ces deux groupes de base doivent offrir des services essentiels pour le fonctionnement des autres pairs sur le réseau.

Les services d'un groupe de pairs

Les services de base pour les groupes de pairs JXTA sont définis comme suit :

- Service de découverte : permet de découvrir les ressources publiés par les pairs d'un autre groupe de pairs.
- Service d'appartenance : Ce service permet aux pairs du groupe d'accepter ou de refuser l'entrée d'un nouveau pair candidat au groupe. La stratégie du groupe peut imposer un système de vote pour valider ou rejeter une candidature.
- Service d'accès : Ce service permet à un pair du groupe de vérifier si une demande d'accès à un service donné, offert par le groupe, lui est autorisé ou non. Cette vérification ne concerne pas tous les services mais seulement ceux qui l'exigent.
- Service Pipe : Ce service est utilisé pour la création et la gestion des Pipes (canaux de communication) entre deux pairs d'un même groupe. Nous reviendrons, avec plus de détails, sur ce point au moment de présenter le Pipes.
- Service de Résolution : Ce service est utilisé par les pairs du groupe pour envoyer des requêtes génériques entre eux, et surtout d'identifier ensuite les réponses correspondantes.
- Service de Monitoring : Ce service permet à un pair donné de contrôler le comportement et l'activité des autres membres de son groupe.

Notons que si les groupes de pairs peuvent offrir d'autres nouveaux services, ils peuvent aussi choisir de ne pas offrir la totalité des services essentiels. C'est aux pairs après de choisir de rejoindre le groupe ou non.

Conclusion

En conclusion, le groupe de pairs est un moyen de rassembler des pairs partageant un intérêt commun. Cette association permet d'une part, une meilleure organisation des pairs, et d'autre part, elle facilite la communication et la découverte. Les politiques d'accès réduisent les risques, en matière de sécurité, pour les pairs d'un même groupe.

6.3.4 Pipe

Définition

Le concept de « Pipe » n'est pas une innovation propre à JXTA. Le principe est utilisé dans les systèmes d'exploitation Unix et le fonctionnement est sensiblement le même. Un émetteur met l'information à émettre d'un coté du « Pipe » (tunnel), et le destinataire la récupère de l'autre côté. Ni l'émetteur ni le destinataire ne se préoccupent des mécanismes de transmission du message au niveau des couches basses du niveau réseau. Notons cependant qu'en réalité, les communications sont souvent basées sur les protocoles TCP/IP pour la simple raison que ces deux protocoles permettent l'existence des pairs sur Internet environnement basé sur TCP/IP. Néanmoins, la couche communication peut se baser sur la technique XML/RPC ou SOAP, c'est le cas s'il s'agit d'un échange entre services.

Un « Pipe » offre un mode de transfert de messages asynchrone et non discriminant par rapport au contenu puisqu'un message peut être une chaîne de caractères, un objet, une séquence binaire etc. Deux pairs correspondants, qui n'ont pas un lien physique direct au

niveau réseau, peuvent être liés par un « Pipe ». En fait, le « Pipe » n'est pas un lien physique direct mais plutôt un canal de communication virtuel qui lie l'émetteur au destinataire. Pour lier deux pairs, qui ne sont pas connectés physiquement l'un à l'autre, des pairs intermédiaires sont alors utilisés.

La plateforme JXTA propose deux types de « Pipes » : « Unicast Pipe» et « Broadcast Pipe» que nous présentons dans le paragraphe suivant.

Les types de « Pipe »

Les deux types de « Pipe » correspondent en réalité à deux modes de communications distincts. Le premier mode est la communication « unicast » ou (point à point), le second correspond au mode « broadcast » ou (diffusion).

- Unicast Pipe : C'est la connexion d'un pair à un autre. JXTA propose deux types de connexion point à point, les connexions sécurisées et les non sécurisées.
- Propagation Pipe : Ce « Pipe » connecte un émetteur (l'entrée de « Pipe ») à plusieurs récepteurs (les sorites de « Pipe ») appartenants à un même groupe. L'émetteur dépose son message sur l'entrée du « Pipe » et les récepteurs le reçoivent à leurs sorties respectives.

Nous avons expliqué au cours de ce paragraphe qu'un « Pipe » sert à transporter des messages entre « Peers ». Au cours du paragraphe suivant nous présentons ce qu'est un message pour JXTA, ainsi que les formes possibles pour la représentation d'un message.

6.3.5 Les messages

Définition

Un message est un objet échangé entre deux pairs. Il est formé d'une séquence ordonnée d'éléments. Un élément est la paire couple (nom, type) plus un contenu. Le contenu peut être de n'importe quel type. L'émission et la réception d'un message se fait grâce aux « Pipe ».

Dans JXTA Deux types de représentation de messages sont possibles : XML et Binaire. L'utilisation de la représentation XML offre plus de flexibilité. Par exemple si un pair, pour des raisons diverses, ne veut interpréter que des parties particulières d'un message il peut le faire grâce à la forme décomposée des messages XML.

Format des messages

Les messages échangés entre « Pairs » doivent être sous un format bien précis, pour que les destinataires arrivent à trouver facilement l'information qu'ils cherchent, et qui est contenue dans le message. Tous les systèmes P2P utilisent leurs propres formats de messages (Napster, Gnutella, AIM Messanger...), et c'est pour cette raison que des « Peers » de ces applications ne peuvent pas communiquer ensemble.

L'idée de JXTA est de proposer un type de format de message précis que toutes les applications basées sur la plateforme doivent utiliser. Des pairs de deux applications différentes peuvent facilement communiquer sans un protocole et des mécanismes définis par les développeurs.

L'un des principaux services de JXTA est celui de la découverte des ressources. Tout pair peut avoir des informations concernant les autres pairs, les PeerGroups, les services qu'ils proposent... Ceci est assuré par le mécanisme de la publicité, traduction à la lettre du concept Advertisement, sujet du prochain paragraphe.

6.3.6 Les advertisements

Définition

Un « advertisement » est un document XML qui décrit, sous la forme de meta-données, les ressources JXTA. Tous les protocoles de la plaetforme JXTA utilisent les « advertisement » pour s'échanger des informations. Les « advertisements » sont de nature hiérarchique et spécifient les types de ressources qu'ils décrivent. L'identifiant de la ressource est un des éléments les plus importants que doit contenir un « advertisement ». Selon les types de ressources, il y aura plus ou moins d'éléments dans un « advertisement ».

Exemple d'« advertisement »

Afin de mieux comprendre le principe d'un « advertisement » et vu l'importance de ce concept dans le reste de notre travail nous avons décidé de présenter un exemple concret. La figure ci-dessous représente un « advertisement » d'un PeerGroup nommé « P2PagentsGroup3 ». Une description, en langage naturel, du groupe, est fournie. Les deux premiers éléments GID et MSID sont les identifiants garantissant l'unicité de la ressource JXTA (PeerGroup) pour ce cas de figure.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE jxta:PGA>
<jxta:PGA xmlns:jxta="http://jxta.org">
<GID>
urn:jxta:uuid-44D4B7ABE0FF4461B19FE2EA8FB08E3302
</GID>
<MSID>
urn:jxta:uuid-EADBEEFDEAFBABAFEEDBABE000000010306
</MSID>
<Name>
    MadkitJXTACommunicatorGroup
</Name>
<Desc>
    The JXTA Madkit Communicator Group
</Desc>
</jxta:PGA>
```

Fig. 6.1 – Un exemple d'un « Advertisement » d'un PeerGroup.

6.4 Modèle conceptuel : Les protocoles JXTA

6.4.1 Introduction

Le modèle conceptuel de JXTA se base sur un certain nombre de protocoles spécifiés et identifiés en tant que concepts de bases pour les applications P2P. Les protocoles JXTA sont conçus sans imposer des contraintes sur l'environnement d'exécution des pairs, et font peu ou pas de suppositions concernant les couches de transport. Les pairs utilisent les protocoles pour publier et rechercher leurs ressources à travers le réseau. Les protocoles sont aussi utilisés pour la communication entre les pairs, sans que ces derniers se préoccupent des problèmes liés aux différentes topologies réseaux sur lesquelles se basent leurs connexions. Bien que les protocoles soient indépendants, ils travaillent ensemble pour faciliter la découverte, l'organisation, le monitoring et la communication entre pairs. Un pair n'est pas obligé d'implémenter tous les protocoles de JXTA.

6.4.2 Les protocoles JXTA

Nous présentons brièvement dans la suite ces différents protocoles, sans entrer trop dans les détails car cela dépasse le cadre de notre travail.

Le protocole : Peer Discovery (PDP)

C'est le protocole de découverte pour les pairs. Ce protocole permet aux pairs de découvrir les services offerts par les autres membres en mode P2P sur le réseau. Ce protocole permet aussi à un pair de se faire découvrir. En effet, chaque pair l'utilise pour se présenter et ainsi faciliter aux autres de connaître sa nature et les services qu'il propose.

Protocole : Peer Information Protocol (PIP)

Grâce à ce protocole un pair peut avoir des informations sur l'état d'un autre pair sur le réseau. Un pair peut savoir si un autre pair est toujours vivant ou non et recevoir plus d'information sur lui tel que son ID.

Le protocole : Peer Resolver (PRP)

C'est le protocole qui permet d'envoyer des requêtes à n'importe quel nombre d'autres pairs et d'en recevoir autant de réponses que de requêtes émises. Les requêtes peuvent être envoyer à un groupe de pairs précis ou à un ensemble précis de pairs. Les protocoles PDP et PIP utilisent le PRP grace à sa générécité. Le premier (le PIP) l'utilise pour formuler des requêtes questionnant sur l'état d'un pair distant, alors que le second le (PDP) est utilisé pour découvrir les ressources offertes par d'autres pairs.

Le protocole : Pipe Binding (PBP)

Ce protocole permet à un pair d'établir une communication, via un canal de communication virtuel (le Pipe), avec un ou plusieurs autres pairs du réseau. Le protocole propose de types de message un de type requête et permet de trouver le pair de l'autre bout d'un canal défini

et établir la communication entre les deux pairs. Le message de type réponse contient les informations concernant le pair recherché et qui se trouve de l'autre bout du canal.

End Point Routing Protocol

C'est le protocole qui propose un ensemble de fonctionnalités pour permettre le routage des messages d'un pair source vers le pair destination. Les messages de type requête permettent de trouver un chemin vers un pair destination. Les messages de type réponse contiennent les informations envoyées par un pair (qui sert là de relaie) et contient le ou les chemins permettant d'atteindre le pair destinataire. Chaque pair peut sauvegarder une liste de chemins vers d'autres pairs avec qui il était en communication. Ces informations peuvent servir pour renseigner d'autres pairs.

Protocole : Rendez-vous Protocol (RVP)

Ce protocole permet la propagation des messages au sein d'un groupe de pairs. Un pair peut jouer le rôle de Rendez-vous et offrir ainsi le mécanisme de propagation à un groupe de pairs qui doivent se connecter à lui pour pouvoir propager et recevoir des messages.

6.4.3 Remarques

La plateforme JXTA implémente les protocoles spécifiés par le modèle conceptuel dans une couche spécialement dédiée à cette tâche, et permettant de traduire ces protocoles en différentes architectures de transport. Nous reviendrons dans la suite avec plus de détails sur les différentes couches de la plateformes qui ne sont pas à confondre aux protocoles. En effet, les couches de l'architecture sont au niveau implémentation tandis que les protocoles sont des spécifications qui se situent au niveau conceptuel, et peuvent être implémentés de différentes façons.

Même si théoriquement un pair n'est pas obligé d'implémenter tous ces protocoles, il est néanmoins obligé, s'il veut être adressé dans le réseau et donc reconnu en tant que pair, d'implémenter au moins les protocoles Peer Resolver Protocol et le Endpoint Routing Protocol. Selon leurs besoins, les pairs implémentent un ou plusieurs des autres protocoles. Par exemple, pour pouvoir découvrir les ressources disponibles et faire publier les siennes un pair a besoin du protocole PDP.

Comme nous l'avons mentionné précédemment, les protocoles de JXTA sont indépendants des langages de programmation qui les implémentent, ils constituent la partie « Quoi faire ? » plutôt que le «Comment faire ?». « Le comment faire » fait plutôt partie du niveau le moins élevé en terme d'abstraction : l'implémentation. Cependant au niveau des protocoles, un certain nombre de messages XML sont définis, afin de permettre la coordination entre les différents composants du système P2P. L'utilisation de XML comme langage de représentation de ces messages est sans doute dû à son expansion et sa simplicité d'utilisation, et surtout à l'abondance des outils de traitement des messages XML existants.

La figure1 présente l'organisation des différents protocoles de JXTA par ordre d'interdépendance. Nous remarquons que les protocoles PeerDiscovery, PipeBinding et Information-Protocol sont au même niveau et se situent au dessus des autres. Nous notons auusi que le

protocole PeerResolver est à un niveau intermédiaire entre les protocoles de haut niveau, et les deux de bas niveau qui sont : PeerEndPoint et RendezVous.

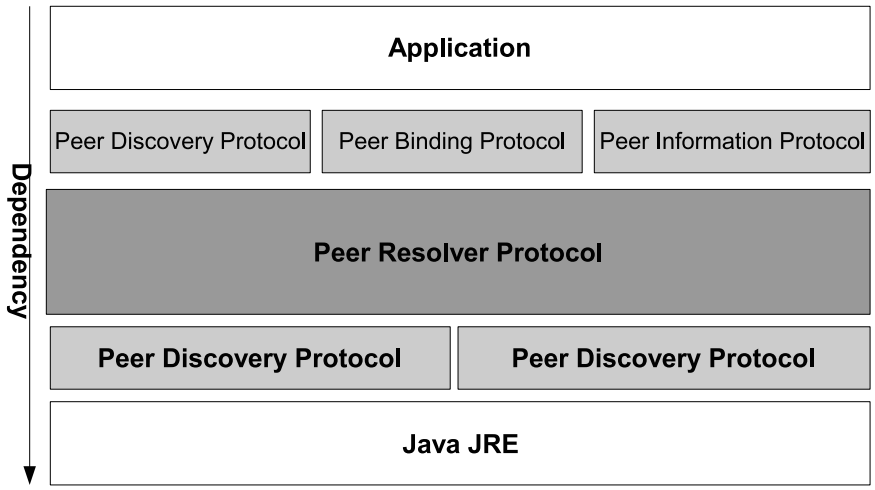

FIG. 6.2 – L'architecture en couches de la plateforme JXTA.

Enfin, on distingue le niveau implémentation des protocoles qui se trouve en plus bas niveau et les applications P2P qui elles se baseront au dessus des protocoles et leurs implémentations.

Avec les protocoles ainsi présentés, les développeurs peuvent créer leurs applications P2P qui vont pouvoir tourner de façon décentralisée, et ce même en présence de FireWall ou au sein de réseaux privés, c'est là une des clés du succès de JXTA. Afin de permettre à ces différents protocoles de bien fonctionner il est nécessaire d'avoir une architecture adéquate. Nous exposerons dans le paragraphe suivant l'architecture de la plateforme JXTA qui implémente les protocoles énoncés par le modèle conceptuel.

6.5 L'architecture JXTA

L'architecture de JXTA que nous présentons dans cette section est mise en oeuvre dans le but d'implémenter les différents protocoles qui ont été précédemment définis, et qui constituent le modèle conceptuel de haut niveau de JXTA. Elle permettra l'utilisation, de façon harmonieuse, des différents protocoles pour bâtir un système complet, facilitant la réalisation d'application P2P. Cette architecture est formée de trois couches. La première couche est le "Core Layer" : la couche noyau. La deuxième couche est le "Service Layer" : la couche des services et enfin le "Application Layer" : la couche application. La figure ci-dessous représente l'architecture de la plateforme décomposée en ses différentes couches.

Nous présenterons plus en détails ces différentes couches de l'architecture de JXTA tout au long des paragraphes suivants.

6.5.1 La couche Noyau

C'est au niveau de cette couche que sont réellement implémentés les six protocoles clés du modèle conceptuel de JXTA, précédemment présentés. Nous remarquons sur la figure7.3, que cette couche contient les mécanisme permettant la gestion des groupes de pairs, des "pipes"

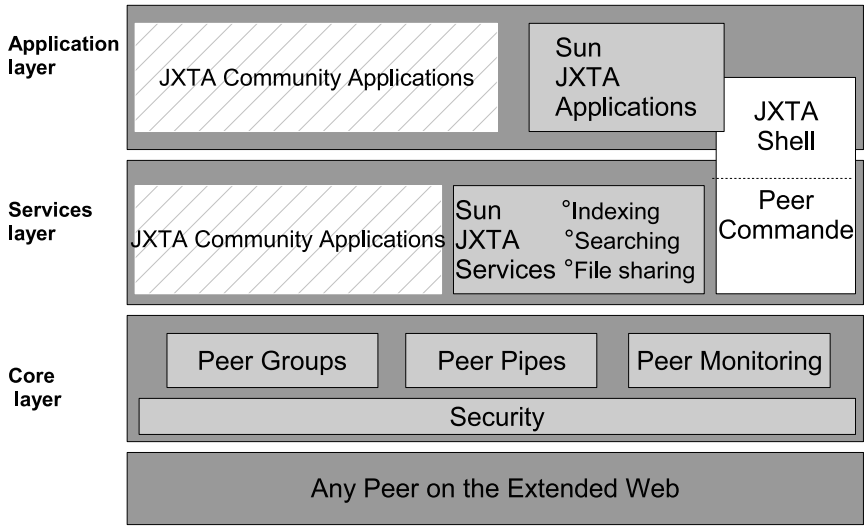

FIG. 6.3 – L'architecture en couches de la plateforme JXTA.

qui sont les canaux permettant la communication entre pairs. En plus, la couche noyau traite l'aspect de tutorat que peut avoir des pairs sur d'autres. La sécurité est aussi traitée à ce niveau si bas de l'architecture afin de maximiser les chances de succès des politiques de sécurité : moins d'abstraction signifie plus de fiabilités. Notons enfin, que c'est là qu'il existe, au dessus de ces protocoles, le groupe universel nommé WorldPeerGroup. Dès qu'un pair est lancé, il est rattaché automatiquement à ce groupe. Cette technique offre deux atouts. Premièrement, le pair aura ainsi accès à tous les services implémentés pour ce groupe, et deuxièmement tous les pairs existants étant rattaché à ce groupe, l'exploration des autres pairs peut ainsi débuter dès la création d'un pair.

6.5.2 La couche Service

Cette couche est un ensemble de services offerts aux utilisateurs afin de leur faciliter la tâche de la création d'applications en P2P. Cette couche est composée de deux types de services : les services de base et les services complémentaires. Avant d'expliquer la différence entre ces deux types de services, il faut spécifier ce qu'est concrètement un service. Un service est une fonctionnalité située au dessus de la couche Noyau et utilise les protocoles qu'elle implémente dans le but de réaliser une tâche précise. C'est la qu'intervient la notion de services basiques et services complémentaires. En effet, il y a des services incontournables et nécessaires pour la réalisation des applications P2P. Ces services sont souvent utilisés par n'importe quel pair, nous pouvons citer à titre d'exemple la création d'un "pipe", qui rappelons est le canal permettant d'établir une communication en P2P entre deux pairs est un service essentiel. Par contre, un service permettant de traduire des messages en différents types de langages, n'est pas un service essentiel et ne sera pas requis par la majorité des applications. Plusieurs services des deux types sont présents dans la plateforme.

6.5.3 La couche Application

Cette couche est réservée aux développeurs d'application P2P, c'est leur champ d'action et de création. C'est à ce niveau que les programmeurs mettront en oeuvre leurs application

P2P en se basant sur les deux couches de bas niveau : le Noyau et les Services. Cette couche permettra à des pairs de travailler conjointement au sein d'une même application ayant un objectif global commun, comme le partage de fichier ou la téléphonie en P2P... Notons cependant, que si l'application est plutôt orientée service, dans la mesure où elle offre des fonctionnalités aux pairs plutôt qu'utiliser les pairs pour faire des applications, elle se trouve dans la couche Service même si cela n'est pas l'objectif initial de ses concepteurs. Cela est sans doute dû à la fragilité de la barrière existante entre les deux concepts : Service et Application. Mais de toutes les façons cela n'a aucune incidence sur le fonctionnement de l'application.

6.6 Conclusion

Dans ce chapitre, nous avons étudié JXTA, un modèle conceptuel basé sur un ensemble de protocoles, et implémenté par une architecture de trois couches. Cette étude nous a permis, d'une part, de présenter les différents principes et concepts de la plateforme JXTA qui sont dans la plupart utilisés dans la majorité des applications P2P, et d'autre part d'expliquer les sources d'inspiration qui nous ont permis de concevoir les concepts du modèle AGRS que nous proposons. Le choix de JXTA se justifie par la multitude de fonctionnalités offertes par cette plateforme. Hormis les mécanismes de recherches et de publication efficaces, JXTA offre une solution transparente de communication indispensable dans un environnement comme Internet. Cette transparence constitue sans doute un avantage qui nous a aidé à trancher dans notre choix de cette plateforme. En effet, cela facilite la combinaison avec la plateforme Multi-Agents que nous allons utiliser pour implémenter le modèle AGRS. Cette combinaison fournira l'architecture hybride Agents-P2P sujet du prochain chapitre.

Chapitre 7

Une architecture hybride Agents-P2P pour la mise en oeuvre du modèle AGRS

7.1 Introduction

Dans ce chapitre nous allons présenter l'architecture que nous avons développé pour mettre en oeuvre le modèle organisationnel AGRS. Pour vérifier la réalisabilité de notre proposition organisationnelle, il nous fallait la mettre en oeuvre sur une plateforme multi-agents et vérifier son intérêt pratique sur un ensemble d'applications. Nous soulèverons des questions qui s'inscrivent dans le cadre d'une problématique qui se rapporte au déploiement à grande échelle des systèmes multi-agents. Ici, nous nous intéressons aux problèmes réseaux qui constituent un handicap au déploiement des plateformes agents dans des environnements comme Internet. Mais nous nous focalisons plus particulièrement sur les problèmes imposés par le modèle AGRS et les concepts qu'il introduit (publication et recherche de services, utilisation des rôles, gestions décentralisée...).

Nous implémentons notre modèle AGRS sur la platforme multi-agents Madkit. La première étape consiste donc à examiner la plateforme Madkit. Notre intérêt porte plus particulièrement sur les aspects de communication et de distribution qui constituent le talon d'Achille de toute plateforme destinée au déploiement à grande échelle.

Nous essaierons ensuite de déterminer la problématique qui se pose au déploiement à grande échelle de Madkit et les problèmes qu'imposent les nouveaux concept du modèle AGRS.

Nous examinerons ensuite en détails l'architecture proposée : Madkit-JXTA, tant au niveau de sa construction interne, que sur son déploiement sur Internet. Nous finirons par présenter un exemple d'utilisation concret de la plateforme, qui permettra de mieux cerner les modifications apportés à la plateforme tant au niveau interne qu'au niveau de la programmation des agents.

7.2 La plateforme multi-agents Madkit

Madkit (Multi-Agent Developpement Kit) est une plateforme multi-agents modulaire écrite en java et qui se base sur le modèle organisationnel AGR. Ce modèle sert à la réalisation des applications implémentées sur Madkit, mais aussi pour le fonctionnement interne de la

plateforme. Tout dans l'architecture interne de la plateforme Madkit est exprimé en fonction des trois concepts du modèle organisationnel AGR (Agent, Groupe et Rôle). Madkit se base sur trois principes architecturaux : un micro noyau, une agentification des services et une interface componentielle.

Nous exposerons dans le paragraphe suivant le micro noyau, la structure et le fonctionnement des agents sur la plateforme.

7.2.1 L'architecture du micro-noyau agent

Le terme "micro-noyau" est utilisé ici de façon intentionnelle. En effet, le système est inspiré du modèle des systèmes d'exploitation à micro-noyau. L'environnement est de petite taille (moins de 40K), il contient un nombre réduit d'outils qui facilitent le déploiement des services du système [25]. Ce micro noyau a pour mission d'assurer un certain nombre de fonctionnalités qui sont les suivantes :

Gestion des groupes et des rôles

Etant donné que la plateforme Madkit se base sur le modèle AGR, il est évident qu'il faut traiter les mécanismes permettant sa gestion au niveau le plus bas, afin de faire abstraction sur le modèle et l'état interne des agents. C'est donc le noyau qui se charge de la gestion des mécanismes permettant l'utilisation des notions de groupes et de rôles [26]. Afin de réussir sa mission, le noyau doit maintenir les informations concernant les rôles joués par les agents, au sein de quels groupes ? Quels sont les rôles associés à chaque groupe de la plateforme ?... Le noyau doit aussi être capable de répondre aux requêtes concernant l'organisation des agents, des groupes et des rôles.

Gestion du cycle de vie des agents

Tout agent, quel que soit son architecture interne, a un cycle de vie classique qui est constitué par : la création, l'activation, l'exécution et la mort. Le noyau Madkit comme tout autre plateforme multi-agents gère ce cycle de vie. Il associe à chaque agent un identifiant unique afin de le distinguer des autres agents. Cette adresse peut être modifiée pour faciliter l'intégration avec d'autres plateforme hétérogènes [26].

L'architecture de communication sur Madkit

Différents types de messages sont disponibles et faciles à réaliser et à transmettre. Mais nous distinguons deux cas d'envoie de messages. Les messages locaux qui sont échangés entre agents sur la même plateforme, et les messages distants où le destinataire se situe dans une plateforme externe sur le réseau.

Un agent qui désire envoyer un message n'a pas à préciser si c'est un message local ou distribué. Le noyau intercepte l'envoi de message et vérifie s'il est local ou distribué. Une des particularités du noyau Madkit est que l'échange de messages entre agents locaux se fait par un simple échange de références. S'il s'agit d'un message distant un service particulier, assuré par un agent responsable de la communication distante, est invoqué et se chargera de l'envoi du message.

7.2.2 L'agentification des services

Le principe de micro noyau signifie que la plateforme assure un ensemble minimal de fonctionnalités permettant le déploiement d'autres systèmes. Dans ce sens, le micro noyau Madkit maintient, comme on l'a vu, l'organisation (table de groupes et rôles), gère les cycles de vies des agents et assure le transfert des messages en local. Tous les autres services ne font pas partie du micro noyau et en particulier la communication en distribué. Ces services sont assurés par des agents encapsulés par un seul agent initial : c'est l'agentification des services. En particulier, la communication inter noyaux est un service assuré par un agent spécial (le SiteAgent) qui utilise d'autres agents offrants différents modes de communications. Nous examinons dans le paragraphe suivant le mécanisme d'envoi de communication entre noyaux utilisé par Madkit.

7.2.3 La communication entre plateformes Madkit distantes

Quand le noyau intercepte un message à envoyer et identifie qu'il s'agit d'un message distribué, le SiteAgent (l'agent offrant le service de communication distante) est sollicité et se chargera de son envoi. Le SiteAgent, contactera son Communicator (un agent responsable de la communication au bas niveau) en lui confiant la responsabilité de transférer le message. Le SiteAgent comme l'agent émetteur, ne se préoccupent pas du moyen de communication utilisé par les « Communicator » (Corba, Socket, RMI...). A la réception d'un message, le Communicator le transfert à son SiteAgent qui saura s'il s'agit d'un message de synchronisation ou à destination d'un autre agent de la plateforme. Au premier cas, il l'interprète et met à jour l'organisation distribuée. Si par contre le message reçu est destiné à un autre agent, il sera remis au noyau qui le délivrera au destinataire. La figure ci-dessous illustre le mécanisme mis en place sur Madkit pour assurer la communication distribuée.

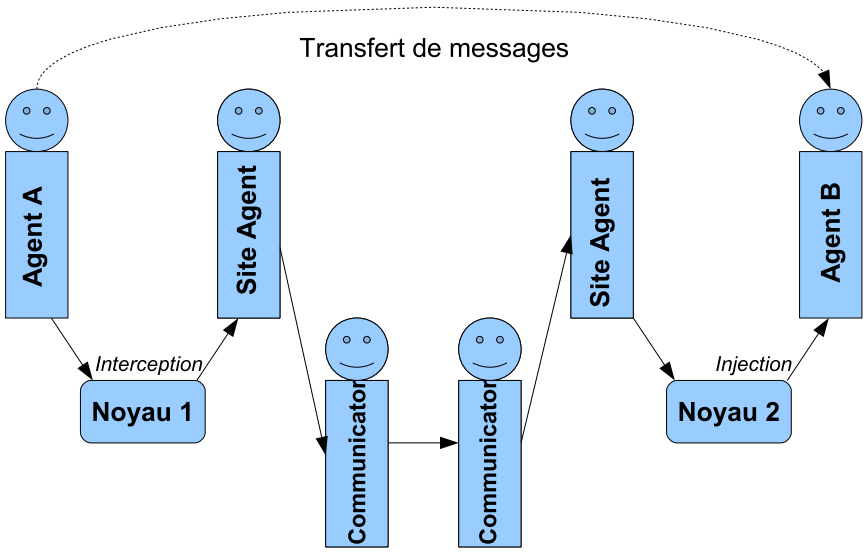

FIG. 7.1 – La communication distribuée entre plateformes Madkit distantes.

Le mécanisme de communication distribuée présenté s'appuie sur une contrainte forte d'intégrité de l'environnement multi-agents au niveau des communications. Cette contrainte implique qu'il ne doit pas exister, en outrepassant leurs environnements respectifs, de relation directe entre deux agents distants. Ce concept d'intégrité s'étend sur d'autres services et

opérations sur l'environnement, tel que la création ou suppression de groupes, la migration pour les agents mobiles ou même encore la gestion du cycle de vie de l'agent. C'est à la plateforme d'effectuer toutes ses opérations pour le compte de l'agent.

L'avantage de cette séparation est l'utilisation unique du SiteAgent indépendamment du mécanisme de transfert de messages utilisé. La liaison entre le SiteAgent et les agents responsables de la couche physique de transport des messages se fait par envoi de messages, ce qui assure une plus grande flexibilité et modularité. En effet, le SiteAgent ne connaît rien sur son agent de communication à part qu'il doit jouer le rôle de « Communicator ». Il est ainsi envisageable, que plusieurs types d'outils de communication soient utilisés au cours du cycle de vie d'un noyau. Il est même possible que l'utilisateur mette en place son propre « Communicator ». Derrière ce rôle peuvent se cacher plus d'un agent et différents types et moyens de communication.

7.2.4 Gestion de la distribution

Madkit utilise la diffusion totale pour gérer la distribution de l'organisation sur les différents noyaux connectés sur le réseau. Chaque nouveau noyau reçoit le schéma organisationnel complet existant sur le réseau. Les groupes et communautés publics sont alors instanciés au niveau de chaque noyau. Concrètement, c'est le SiteAgent qui reçoit de la part des autres SiteAgent distants les copies des tables, des groupes, des rôles et de communautés publics. Toute modification sur un des noyaux, qui touche à l'organisation (par exemple : un agent qui joue ou quitte un rôle par exemple) ou à sa structure (par exemple : création d'un nouveau groupe), est propagée à tous les noyaux connectés. Ce mécanisme est loin d'être satisfaisant, surtout dans des environnements dynamiques et ouverts ; contexte de notre travail.

7.2.5 Structure et fonctionnement d'un agent

Tout agent qui tourne sous la plateforme Madkit est issu d'une classe d'agent abstraite nommée « AbstractAgent ». Cette classe d'agent offre un certain nombre de fonctionnalités de base, nécessaires pour le bon fonctionnement d'un agent quel que soit son modèle conceptuel. Ces fonctionnalités sont : les outils de communication et la manipulation d'une interface graphique qui lui est associée. En plus tout agent peut invoquer des primitives lui permettant d'avoir une vue sur l'organisation locale.

Un agent créé, ne débutera réellement son cycle de vie qu'une fois il est activé. En fait, à l'activation de l'agent le noyau lui affecte une adresse unique, il devient ainsi identifiable par les autres membres de la société et pourra aussi faire partie de l'organisation et jouer des rôles dans différents groupes. Un autre aspect de liberté de Madkit est qu'il n'impose aux développeurs aucun concept d'exécution pour l'agent. En effet, une section est réservée pour démarrer le cycle de vie et ensuite c'est à l'utilisateur de programmer le comportement de l'agent.

La figure ci-dessous présente l'architecture générale de Madkit.

Madkit est donc une plateforme particulière par sa taille réduite et son abstraction quant aux types d'agents, leurs modèles conceptuels et états internes. Cependant, les mécanismes de communication distante et de la gestion de la distribution ne sont pas adaptés à un environnement ouvert et dynamique tel que Internet. Nous présenterons dans le paragraphe suivant tous les obstacles qui entravent le déploiement à grand échelle de Madkit.

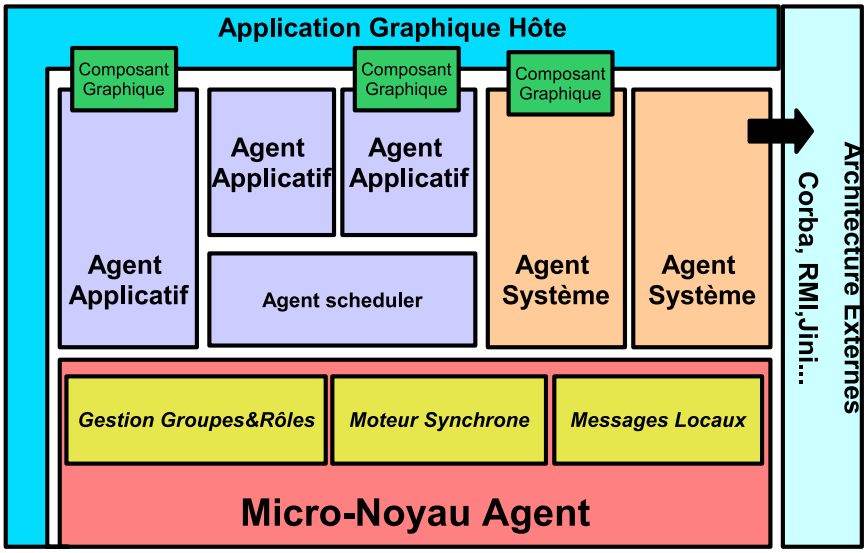

FIG. 7.2 – L'architecture générale de Madkit.

7.3 Problématique

Une architecture Multi-agents est par définition distribuée [Wooldridge 2002]. Dans une application distribuée différents composants sont situés de part et d'autre du réseau et arrivent à communiquer et à s'interconnecter. Pour une application Multi-agents distribuée, ces composants sont situés sur des plateformes agents (noyaux). Ces plateformes doivent offrir un certain nombre de fonctionnalités afin d'assurer la distribution des applications. Parmi ces fonctionnalités nous pouvons retenir :

- Localisation des agents de l'application d'une façon unique (nommage, adressage).
- Communication entre agents d'un même noyau, mais aussi entre les agents situés sur différents noyaux.
- Gestion de l'organisation partagée par les agents situés sur les différentes plateformes.
- Publication et recherche des services et compétences proposés par les différents agents indépendamment de leurs localisations.

Plus le nombre de noyaux est important, plus l'application distribuée est difficile à maintenir. Le problème devient plus complexe si en plus, l'application est ouverte et déployée sur un réseau public comme Internet. Dans un tel contexte, le nombre de noyaux interconnectés est imprévisible et varie continuellement, les noyaux (et donc leurs agents) se connectent et déconnectent de façon aléatoire.

Pour être viable, un système multi-agents, déployé sur un tel environnement, doit être flexible et capable de s'adapter dynamiquement, de manière transparente pour l'utilisateur. Une gestion centralisée n'est pas possible, vu le nombre important de noyaux entraînant les problèmes classiques de surcharge. En outre, chaque agent peut être à la fois client et serveur, il peut offrir et chercher en même temps des services et compétences situées sur le réseau. Les noyaux accueillants les agents sont donc à la fois clients et serveurs. Enfin, le déploiement dans un réseau ouvert pose le problème de l'hétérogénéité des noyaux tant au niveau logique (synchronisation des structures, compréhension des messages), que sur au niveau physique transfert de messages (protocoles et techniques de communications) [Ferber, 1999].

7.3.1 Gestion de la distribution sur Madkit

Nous avons expliqué dans un précédent paragraphe que Madkit utilise la diffusion totale pour gérer l'organisation distribuée sur les différents noyaux du réseau. Il est évident que cette solution n'est pas des plus optimales et n'a que l'unique avantage d'être simple à mettre en oeuvre. La contrainte d'ouverture et de dynamique des agents entraîne souvent une saturation du réseau, vu le très grand nombre de messages échangés. Cela pourrait être anodin, dans le cas d'un système fermé où le nombre de noyaux est connu à l'avance et limité, mais cela s'avère particulièrement gênant dans les applications ouvertes et déployées à grande échelle où le nombre d'agents, de groupes et rôles ou même le nombre de noyaux n'est pas connu et peut être très grand et varier en cours d'exécution.

Certaines modifications au niveau de l'organisation générale ou même de la structure de celle-ci ne sont pas nécessaires à la majorité des plateformes connectés. Ces modifications peuvent ne concerner que certains agents sur certaines plateformes spécifiques impliquées dans une application ou une partie d'une application. Il est dans ce cas inutile de propager les modifications et changement survenus sur l'organisation, à toutes les plateformes existantes sur le réseau. D'autre part, tous les évènements, qui modifient l'organisation générale ne nécessitent pas toujours une diffusion immédiate. Le fait qu'un agent entre dans un groupe, n'est pas toujours un évènement que tous les agents de toutes les plateformes sont sensés connaître à l'instant même de sa production. Nous proposons dans notre travail des mécanismes de gestion de la distribution adaptée au contexte de dynamique, d'ouverture et de large distribution dans lequel nous travaillons.

7.3.2 Problèmes réseau

Les plateformes agents existantes se basent, dans la plus part des cas, sur les adressages IP fixes, qui ne peuvent plus êtres considérés comme moyen de localisation des systèmes sur le réseau. En effet, faire associer une machine à une même adresse IP utilisée sur le réseau local comme sur Internet est de moins en moins possible pour diverses raisons. D'une part, cela peut poser des problèmes de sécurité et d'anonymat sur Internet. D'autre part, les plages d'adresse IP existantes ne peuvent satisfaire tous les utilisateurs sur le réseau Internet et les fournisseurs d'accès optent souvent pour des adresses IP dynamiques. Pour pallier à ces défaillances plusieurs techniques tels que les NAT, Proxy ou les Firewall sont souvent utilisées.

NAT : Network Address Translation

Le terme NAT représente les initiales de "Network Address Translation", ou "Traduction d'Adresse Réticulaire" en français. C'est une translation de l'adresse de réception d'un paquet d'une adresse publique connue du réseau externe en sa vraie valeur dans le réseau local, afin de délivrer le message à son destinataire réel. Plus concrètement, il s'agit d'une modification de l'adresse IP, effectuée par les routeurs, dans l'en-tete d'un datagrame IP. Nous distinguons deux types de NAT : statique et dynamique.

– La NAT statique fait correspondre à une adresse IP privée interne, une adresse IP externe publique. Il s'agit donc d'une association n-n entre adresses publiques et privées. Ce principe s'il permet à une machine, ayant une adresse privée, d'être connectée sur Internet, il ne permet pas de résoudre le problème de pénurie d'adresses IP sur Internet.

– La NAT dynamique ou mascarade IP, à l'opposé du NAT statique, associe une seule adresse publique à n adresses privées. Le routeur aura plus de travail à effectuer lors de la réception des paquets, mais cela permet de résoudre considérablement le problème de pénurie d'adresses IP. Cette technique est parfaite quand il s'agit d'application désirant être passives et ne faisant que de la connexion consultative sur Internet. Mais cela devient problématique quand il faut être joignable de l'extérieur. C'est le cas des applications désirant offrir des services ou se comporter en tant que serveurs. Les plateformes multi-agents désirant être interconnectées, se trouvent dans ce dernier cas de figure. Chaque plateforme doit pouvoir se connecter à l'autre et donc elles doivent toutes êtres joignables depuis l'extérieur, ce qui est impossible avec la mascarade IP. Afin de remédier à ces manques, des extensions du NAT dynamiques ont vu le jour.

Le port forwarding

Le port forwarding, ou redirection de port, consiste à associer un port à une adresse IP d'un service donné. La redirection d'un paquet vers une machine précise s'effectue en fonction du port de destination de ce paquet. Ainsi, lorsque on n'a qu'une seule adresse publique avec plusieurs machines derrière en adressage privé, il est possible d'initialiser une connexion de l'extérieur vers l'une de ces machines (une seule par port TCP/UDP). Bien qu'il résout le problème de visibilité de l'extérieur pour des machines à adresses privées, un autre problème reste néanmoins présent. Le cas de deux applications utilisant le même port sur une même machine est impossible. Cette technique requière l'intervention humaine au niveau du routeur afin de préciser les règles de routage : quelle port correspond à quelle adresse IP ? Cela n'est pas sans poser des problèmes pour le déploiement des plateformes mulit-agents.

Les Proxy

Un proxy est un intermédiaire entre le client et une application donnée. C'est à dire qu'il sert de mandataire pour l'application dans une connexion avec le client pour relayer la requête qui est initialisée. Le client ne voit pas l'application mais s'adresse toujours au proxy, et ce dernier s'adresse ensuite au serveur. Les proxy requièrent dans la plupart des cas les techniques NAT pour fonctionner. Par conséquent, les mêmes problèmes du NAT se retrouvent pour les Proxy. L'intervention de l'utilisateur est aussi requise pour configurer les proxy.

Les Firewall

En plus des problèmes d'adresses privés publiques et de visibilités posées par les NAT, s'ajoute les firewall. Le Firewall (ou Pare-feu) s'intercale entre la machine ou le réseau local le réseau externe ou Internet, et surveille les échanges d'information, dans les deux sens, entre ces deux réseaux. Il examine chaque paquet d'information qui le traverse et décide, en application des règles définies lors de sa configuration, de le laisser passer ou de le détruire. Les règles permettant à chaque application de recevoir et émettre de et vers l'extérieur, doivent être ajoutés par l'utilisateur au niveau de la configuration du Pare-feu.

Ces différentes techniques et mécanismes sont indispensables et très largement utilisés dans les applications de nos jours et même encore plus compliqués avec d'autres protocoles comme le DHCP (Dynamic Host Configuration Protocol). Il est toutefois devenu impossible d'utiliser des systèmes multi-agents à base d'adresses IP fixes de façon transparente et sans se soucier de

l'adressage publique ou privé, ou encore configurer une multitude d'outils de sécurité et d'accès réseau. Tout cela empêche le déploiement des plateformes agents sur les réseaux publics. JXTA résout proprement et de façon transparente tous ces problèmes et débarrasse les utilisateurs et programmeurs de tous ces soucis encombrants. Nous proposons de positionner la plateforme Madkit au dessus de la plateforme JXTA pour interconnecter sur des réseaux distants, via Internet ou d'autres larges réseaux, des plateformes Madkit.

7.3.3 Problèmes imposés par le modèle AGRS

Rappelons les principaux concepts apportés par le modèle AGRS au modèle AGR, base de la plateforme Madkit. L'activité sociale des agents dans AGRS s'exprime en terme de services. Il est essentiel de gérer le concept de service au niveau de la plateforme. Le noyau a la responsabilité d'offrir aux agents les fonctionnalités permettant de créer, de publier et de chercher des services existants sur le réseau. Cela nécessite, une structuration des services offerts par chaque groupe d'une façon optimale, permettant d'une part de trouver le plus grand nombre de services répondant à une requête donnée, et d'autre part le temps de réponse doit être optimal.

Le rôle dans Madkit comme pour AGR n'est qu'une simple étiquette à laquelle des agents sont rattachés et enregistrés sous forme d'un tableau d'adresses. Dans AGRS le rôle encapsule un ensemble de services offerts aux agents qui l'utilisent et qui seront fournis par ceux qui le jouent. Il faut que la réalisation du rôle passe d'une simple étiquette à une implémentation plus riche, permettant d'exprimer les services mais aussi les règles et attributs indiqués au niveau organisationnel.

Selon les spécifications du modèle AGRS, comme pour AGR, le groupe est le cadre de toute activité sociale. Cependant, les agents d'un groupe donné interagissent directement sans passer par aucun autre intervenant externe. Le concept de AgentInRôle est la solution que nous proposons dans AGRS pour structurer concrètement les interactions entre agents. L'execution d'un rôle par un agent jouant le rôle passe par l'entité AgentInRole. L'utilisateur du rôle interagit à travers la relation AgentInRole avec l'agent joueur du rôle pour obtenir un service. Le groupe ne contient plus uniquement l'ensemble des rôles et la liste de ses agents, mais aussi les relations de type AgentInRole et leurs instances. En plus, Madkit doit proposer les mécanismes nécessaires pour la conception des règles du groupe. Enfin, le groupe doit fournir une description de l'ensemble des services qu'il propose et des conditions permettant de le rejoindre.

Pour conclure cet examen des problèmes posés par l'implémentation du modèle AGRS à la plateforme Madkit, on pourrait dire que toutes les implémentations des concepts de base de AGR (Agent Groupe et Rôle) sont à modifier en intégrant les règles aux groupes et rôles. En plus, il faut mettre en oeuvre les fonctionnalités permettant de gérer les services au niveau de tous ces concepts.

7.4 L'architecture Proposée

Madkit n'a bien évidement pas les propriétés nécessaires pour un déploiement à grande échelle dans un environnement ouvert et dynamique. Nous avons identifié trois axes problématiques différents qu'on a étudié et qu'on essaie de résoudre dans le paragraphe suivant.

Notre réflexion a été la suivante :

- Les mécanismes de communication distribuée de Madkit ne sont pas adaptés aux nouvelles contraintes réseau. Les NAT, FireWall et autres techniques de protection constituent un réel handicap pour le déploiement de Madkit et les plateformes agents d'une façon générale sur des environnements ouverts comme Internet. A ces problèmes réseaux, nous proposons l'utilisation des fonctionnalités offertes par JXTA, outil désigné pour les architectures distribuées P2P déployées sur des environnements ouverts comme Internet.
- Madkit repose sur le modèle organisationnel AGR qui, comme on l'a montré, ne permet pas de répondre aux besoins de dynamique et n'est pas adapté à une distribution massive, deux propriétés essentielles dans notre contexte de travail. Ces lacunes sont palliées par les concepts du modèle AGRS que nous essayerons de mettre en oeuvre dans Madkit.
- La gestion des groupes d'agents et de l'organisation générale des applications déployées sur Madkit, est centralisée et se base sur la diffusion massive. Nous avons proposé de considérer le groupe comme une mémoire partagée entre les différentes plateformes. Chaque plateforme possède sa propre copie du groupe. La gestion se base sur les principes du « knowledge pump », nous proposons d'utiliser là aussi les fonctionnalités de JXTA pour assurer la communication entre plateformes et maintenir la cohérence entre les copies.

7.4.1 JXTA comme outil de communication et distribution de Madkit

C'est en fait là que réside le choix radical de notre plate-forme : déléguer la communication distante et les fonctionnalités de recherche et publication des services sur le réseau. Au lieu d'utiliser l'architecture de communication native de Madkit, nous proposons de composer les deux architectures afin de proposer un ensemble regroupant d'une part, les forces du paradigme multi-agents qu'offre la plateforme Madkit et d'autre part, d'exploiter les services proposés par JXTA pour le large déploiement et la communication sur le réseau, abstraction faite des mécanismes de protection. Nous avons dégagé deux approches différentes permettant de mettre en oeuvre une telle technique.

Approche individualiste

Une approche possible simple et intuitive est d'offrir à chaque agent toutes les fonctionnalités de la plateforme JXTA. Tout agent pourra, d'une façon autonome, utiliser sa propre plateforme JXTA et exploiter les fonctionnalités qu'elle offre (communication, recherche et publication...). Cependant, permettre aux agents de créer et d'effectuer des communications directes entre eux, sans aucun contrôle par le noyau, est une entrave à l'intégrité de l'environnement énoncée précédemment. En effet, deux agents ne doivent pas communiquer directement sans passer par le noyau.

Hormis ce problème conceptuel, cette solution a une autre limite qui se situe cette fois au niveau de la complexité. Charger tout agent avec tous ces services produira d'une part, des agents très lourds en taille et gourmands en ressources et d'autre part, un système impossible à contrôler dès que le nombre d'agents devient important. Il est fortement souhaitable que tous les agents aient recours, via la plateforme, à un seul et unique agent de la plateforme qui fournira l'ensemble de services leur permettant un comportement de PeerAgent. De fait, il est très intéressant de remarquer qu'on se retrouve alors dans la logique d'agentification de

services, principe architectural de base de Madkit. C'est l'approche que nous adoptons et que nous étudions dans le prochain paragraphe.

Approche globalisante

Cette approche, que nous adoptons pour notre architecture, consiste à englober les fonctionnalités de communication, de recherche et publication par la plateforme Madkit ce qui évite le déploiement d'un noyau JXTA par Agent. Il s'agit d'un effort d'intégration et d'interopérabilité entre la plateforme mulit-agents Mdakit et la plateforme P2P JXTA. Par soucis de généricité, nous optons pour une encapsulation des services offerts par JXTA dans un agent hybride qui sera l'interface entre les deux plateformes, il sera agent part naissance et endossera la tenue de Pair en implémentant la plateforme JXTA.

La figure ci-dessous illustre l'architecture résultante de cette intégration entre les deux plateformes. En comparaison avec les deux architectures présentées séparément, Madkit est situé au niveau « Application » de la pile de protocoles composant la plateforme JXTA présentée dans la figure 7.3. JXTA fait partie des architectures externes dans la structure de Madkit présentée dans la figure 8.2. Remarquons que le lien entre les deux couches (couche Madkit et couche JXTA) doit être établi, selon l'architecture de la figure 8.2, par l'intermédiaire d'un agent Système. C'est l'agent hybride qui incarnera cet agent système et fera le lien entre les deux plateformes.

Nous allons explorer plus en détails cette proposition en décrivant le rôle de cet agent et sa position dans le système Madkit, ses liens et ses fonctionnalités.

7.4.2 Le JXTACommunicator agent hybride interface entre JXTA et Madkit

Le JXTACommunicator est un agent système qui joue le rôle « Communicator ». Il est en contact direct avec le « SiteAgent » et a pour responsabilité de recevoir et transférer des messages de, et vers les plateformes distantes. A chaque plateforme Madkit est associé un seul et unique agent « JXTACommunicator ». Le « JXTACommunicator » reçoit le message à envoyer et le transfère à son homologue dans le noyau distant. Réciproquement, à la réception d'un message reçu de l'extérieur, le « JXTACommunicator » distant délivrera le message à son « SiteAgent » qui l'envoie au noyau, ce dernier le délivre à l'agent destinataire.

La communication des JXTACommunicator

La communication entre le « SiteAgent » et son « JXTACommunicator » se fait en local par le mécanisme de communication local de Madkit. Par contre, les communications entre les « JXTACommunicator » se fait par le biais des mécanismes de communication de JXTA : les « pipes ».

Une plateforme Madkit peut accueillir différentes applications développées par différents utilisateurs et qui peuvent appartenir à plusieurs communautés. La plateforme représentée par son « JXTACommunicator » doit appartenir aux différentes communautés auxquels appartiennent ses applications. Deux plateformes ne peuvent communiquer que si leurs « JXTACommunicator » sont interconnectés. La communication entre pairs dans JXTA se fait dans le cadre de groupes de pairs. Les agents « JXTACommunicators » ne peuvent donc communiquer

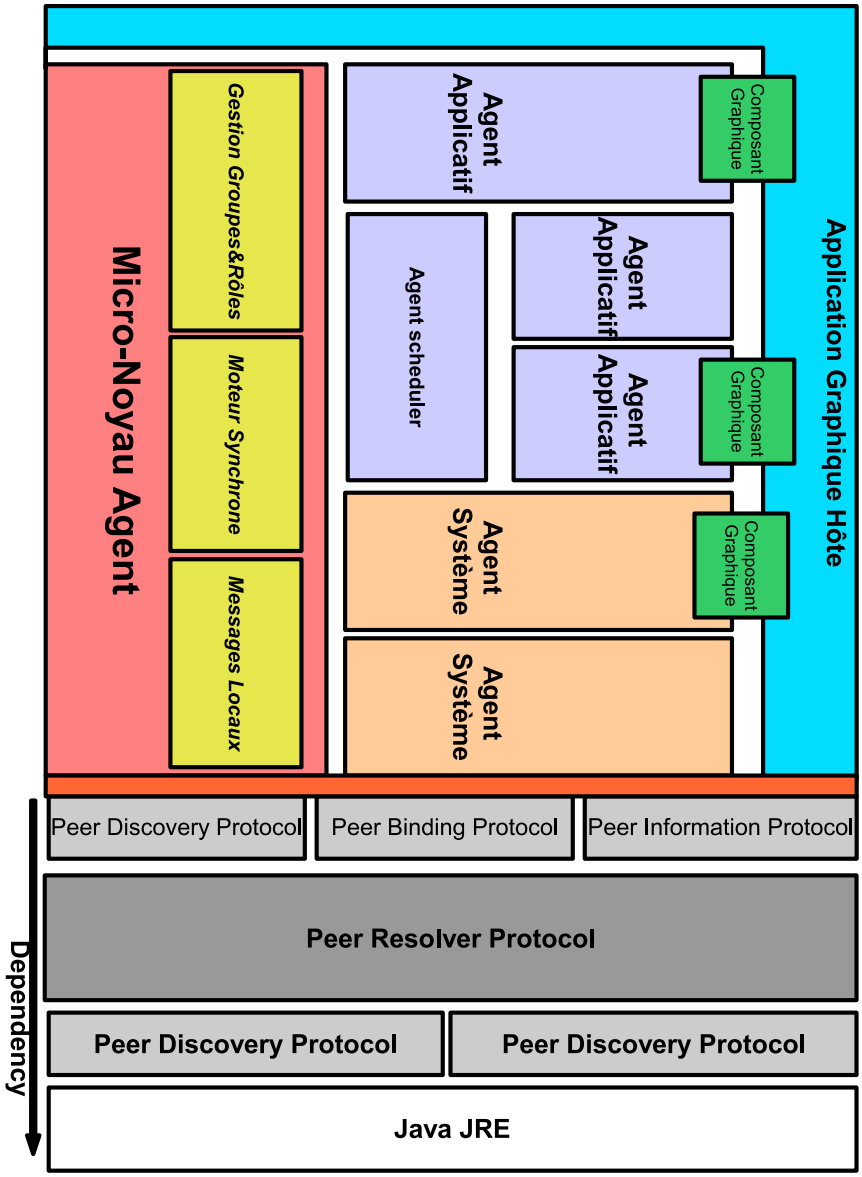

Fig. 7.3 – Architecture hybride proposée.

que s'ils appartiennent à un même groupe de pairs, donc à une même communauté Madkit. La figure ci-dessous, illustre la communication entre différentes plateformes appartenants à des groupes JXTA différents. Il est important de remarque que même si au niveau de la topologie réseau les plateformes sont situés sur le même réseau, la communication entre eux ne tient compte que de la position dans les groupes JXTA. Par exemple les noyaux « Madkit1 » et « Madkit 5» peuvent communiquer bien qu'ils soient sur des réseaux différents, alors que « Madkit1 » et « Madkit 2» ne sont pas interconnectés, bien que situés sur un même réseau (le réseau 1). Evidement cette configuration n'est pas figée, et la dynamique du système peut amener les deux plateformes (la 1 et la 5) à se retrouver dans un groupe JXTA commun, pour des besoins des applications qu'elles accueillent, et seront donc interconnectés. De même, deux noyaux comme « Madkit4 » et « Madkit 2» peuvent ne plus être en relation si l'un d'eux est amené à quitter le groupe « Groupe JXTA B ».

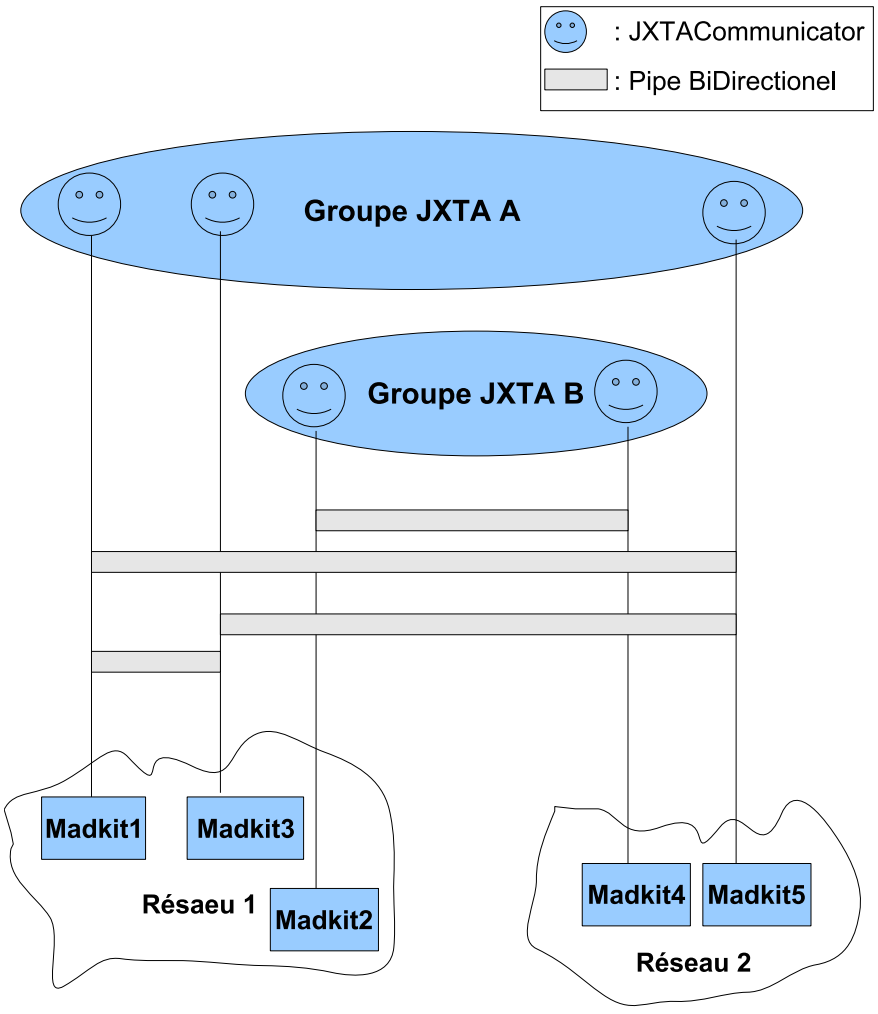

Fig. 7.4 – Organisation de la communication entre JXTACommunicator.

Il est important de noter qu'un seul lien (Pipe) lie deux JXTACommunicator même s'ils sont membres communs de plusieurs groupes JXTA.

Reste à savoir comment les JXTACommunicator s'organisent au niveau du réseau JXTA, et comment ils établissent et maintiennent leurs connexions pour se conformer à l'organisation distribuée des noyaux Madkit qu'ils représentent. Nous expliquons dans le paragraphe suivant

la relation entre l'organisation JXTA que nous implémentant et l'implémentation des concepts du modèle AGRS au niveau de Madkit.

Organisation des JXTACommunicator sur le réseau

Les plateformes sont groupées dans des groupes de pairs au sens JXTA (PeerGroup), ces groupes peuvent être invisibles du reste du réseau. Une plateforme est représentée par son « JXTACommunicator » dans les différents groupes auxquels elle appartient. Nous confondons les concepts de « PeerGroup » (ou réseau JXTA privé) avec le concept de communauté de Madkit. Au niveau de l'organisation de Madkit, une communauté est un ensemble de groupes souvent dédiés à une même application partagée sur le réseau. C'est là le premier point de rattachement entre l'implémentation des deux modèles organisationnels (celui de JXTA et de AGRS). Le développeur d'une application sous Madkit créé son organisation en termes de Communautés, Groupes, Rôles et Services. La création de la communauté sera traduite par la création d'un PeerGroup au niveau de la couche JXTA d'une façon transparente au développeur. En revanche, les concepts de Groupes, Rôles et Services seront encapsulés par la communauté, et ne seront présents sur le réseau JXTA que via la description des groupes. Cette description est le deuxième point d'ancrage entre l'implémentation des concepts de JXTA et ceux de Madkit. Nous y reviendrons plus tard.

Dans des environnements ouverts, contexte dans lequel nous situons notre travail, les communautés se doivent d'être protégées et l'accès peut nécessiter des mécanismes d'authentification. D'autre part, certaines applications ne concernent pas le grand public et ne doivent même pas être visible de l'extérieur, elles doivent appartenir à des communautés invisibles aux autres plateformes du réseau. Nous proposons de répartir les groupes JXTA selon deux types, les groupes privés et les groupes publics. Ces deux types se décomposent, selon les modes d'accès qu'ils adoptent, en deux catégories : libres ou protégés.

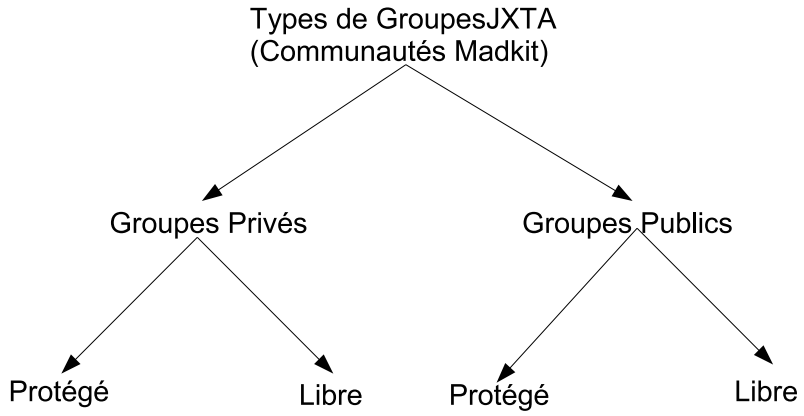

FIG. 7.5 – Classification des différents types de Groupes JXTA proposés.

– Le MadkitJXTACommunicatorGroup : Ce groupe, qu'on notera par soucis de simplicité MadJXTComGrp, dans la suite du travail, est un cas particulier de groupes publics. En réalité, c'est la racine de tous les groupes publics existants. Ce groupe est accessible par tous les groupes connectés sur internet. Toute plateforme Madkit est capable de se connecter directement et à tout moment à ce groupe. Si on veut faire une comparaison simpliste, il serait pour les autres groupes JXTA publics qu'on propose, ce qu'est Internet pour les sites Web. Tout groupe JXTA conçu pour être découvert par les

utilisateurs du réseau Madkit public, et déployé sur Internet devra être représenté par son JXTACommunicator dans le groupe MadJXTComGrp. Cela sera primordial pour l'ensemble des activités de découverte et de publication qu'effectueront les différentes applications déployées sur Internet. Nous pouvons déjà remarquer que tous les JXTA-Communicators, de toutes les plateformes déployées sur le réseau, peuvent communiquer et s'interconnecter. Il est évident que ce groupe est à accès libre.
Il est à noter que nous avons mis en place concrètement ce groupe et qu'il est mis à la disposition de toute la communauté Madkit. Nous expliquerons, lors de l'examen des moyens de gestion des groupes JXTA, comment ce groupe particulier est maintenu et est accessible en continu.

– Les groupes publics : Ces groupes JXTA sont visibles sur Internet. Tout agent peut découvrir un de ces groupes, y trouver un intérêt (un service recherché découvert dans un des groupes Madkit de ce groupe JXTA) et peut tenter d'y accéder. Ces groupes peuvent être à accès libre ou protégé. Si un groupe public est protégé, les plateformes qui désirent le rejoindre doivent connaître ses codes d'accès. Toutes les descriptions des groupes Madkit (au sens AGRS) existants, dans les différents Groupes JXTA appartenants au réseau public sont publiées dans le groupe MadJXTComGrp. Cela permet les recherches des services existants sur le réseau. Ici encore, on peut reprendre la comparaison avec Internet. Les opérations de recherche par service ne sont pas sans rappeler les recherches par mots clés, utilisé dans les moteurs de recherche sur le Web.

– Les groupes privés : Une application qui n'est pas dédiée à une utilisation publique, ne doit pas être exposée à la découverte par les agents appartenants à des plateformes ne faisant pas partie de l'application. Il faut que ces applications soient dans des groupes JXTA privés et sans aucune connexion avec le groupe public MadJXTComGrp. Ces groupes privés sont créés par les utilisateurs qui ont la responsabilité d'indiquer, à toutes les plateformes susceptibles d'y appartenir, le moyen de les localiser. Ces groupes peuvent être à accès protégé ou libre. Ils peuvent aussi être déployés sur le réseau Internet, on sera alors dans un réseau public parallèle à celui que propose Madkit (le MadJXTComGrp) à sa communauté d'utilisateurs, ou dans un réseau interne (dans le réseau interne société par exemple).
Notons enfin qu'un noyau peut bien appartenir, en même temps, à un ou plusieurs groupes publics (le MadJXTComGrp par exemple) et à un ou plusieurs groupes privés. La question qu'on peut légitimement se poser est comment les groupes sont gérés ? Comment les JXTACommunicator peuvent découvrir, localiser et entrer dans les groupes ?

Gestion des groupes JXTA (exemple du MadkitJXTACommunicatorGroup)

Afin de mieux comprendre comment mettre en place un groupe JXTA (une communauté public au sens Madkit), nous allons prendre le groupe public MadJXTComGrp comme exemple. Tout le fonctionnement du système, au niveau de la couche JXTA, se base sur les « Advertisement ». La publication et la recherche de groupes JXTA (communautés), de pairs de des services (encapsulés dans des descriptions de groupes au sens Madkit) ... se fait en terme d'Advertisement. Comme on l'a présenté dans la partie état de l'art, un JXTA Advertisement est une suite de caractères en format XML structurés selon un modèle précisé par JXTA ou par l'utilisateur (si c'est un CustomAdvetisement), et décrivant une ressource existante sur le réseau. La création d'un groupe JXTA passe par un ensemble d'étapes.

A la création d'un groupe JXTA, la première étape consiste à vérifier qu'il n'est pas déjà créé et qu'il n'existe pas déjà sur le réseau. Ce problème ne se pose pas pour le groupe MadJXTComGrp puisque nous garantissons son existence en continu. Mais comment chercher qu'un groupe existe ? Où chercher l'existence ou non d'un groupe ?

Si on compare un groupe JXTA à une maison rassemblant des JXTACommunicator (donc des plateformes Madkit) différentes, les RendezVous en seront sans doute les fondations. Tout se base sur les pairs RendezVous. Un pair Rendez-vous facilite l'échange d'informations concernant les ressources existantes sous sa responsabilité, ces informations sont sous la forme d'Advertisement JXTA. A chaque groupe JXTA doit correspondre au moins un RendezVous. C'est ces pairs qui vont permettre la recherche des groupes.

Pour le groupe MadJXTComGrp, nous mettons en place un ensemble de RendezVous distribués sur différentes machines, afin de garantir l'accessibilité au groupe de façon continue. Le noyau, créant le groupe MadJXTComGrp, est localisé concrètement dans les locaux du Lirmm [Lirmm], et son JXTACommunicator est à la fois agent, simple pair et RendezVous il est un des RendezVous que nous mettons à la disposition du public.

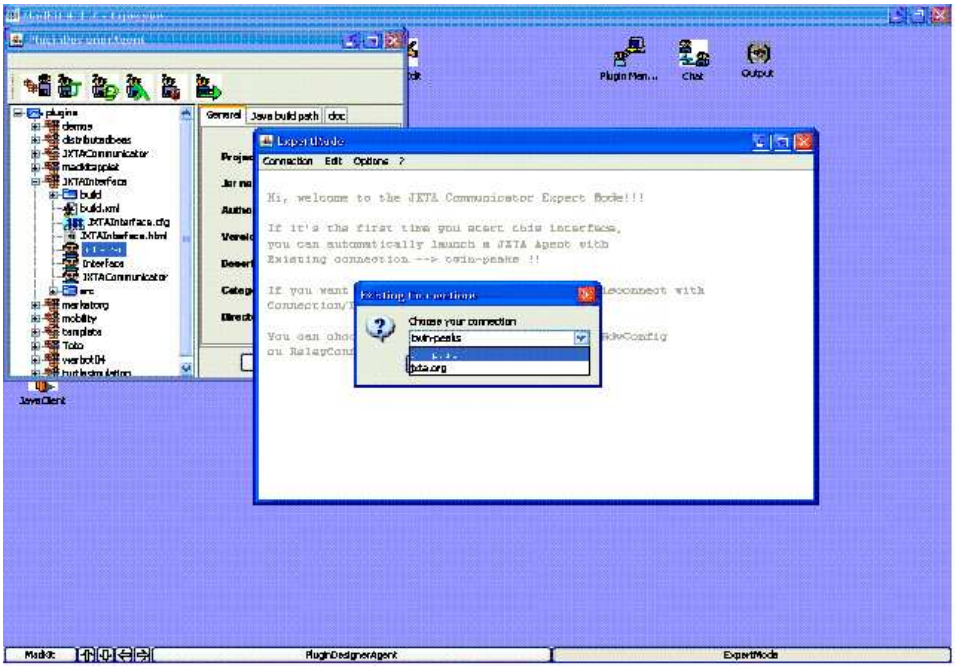

FIG. 7.6 – Connexion au RendezVous public de Madkit et entrée au réseau public.

A la création du groupe MadJXTComGrp, l'Advertisement le décrivant est publié dans les différents RendezVous connectés. Le texte ci-dessous illustre l'Advertisement du groupe MadJXTComGrp.

C'est cet Advertisement qui permettra de localiser le groupe, et de pouvoir le rejoindre dans un second temps. Pour entrer dans le groupe, le JXTACommunicator d'une plateforme doit suivre le processus d'entrée suivant :

1) Demander au « SiteAgent » l'autorisation de jouer le rôle « Communicator » du noyau. 2) Se connecter à un des RendezVous du groupe MadJXTComGrp. 3) Récupérer l'Advertisement du groupe JXTA (le groupe est créé au préalable). 4) Demander à entrer dans le groupe MadJXTComGrp en utilisant l'Advertisement du groupe découvert. L'accès à ce groupe étant sans contrainte d'accès, toute plateforme peut entrer. 5) Créer un « BiDiPipeServer » puis

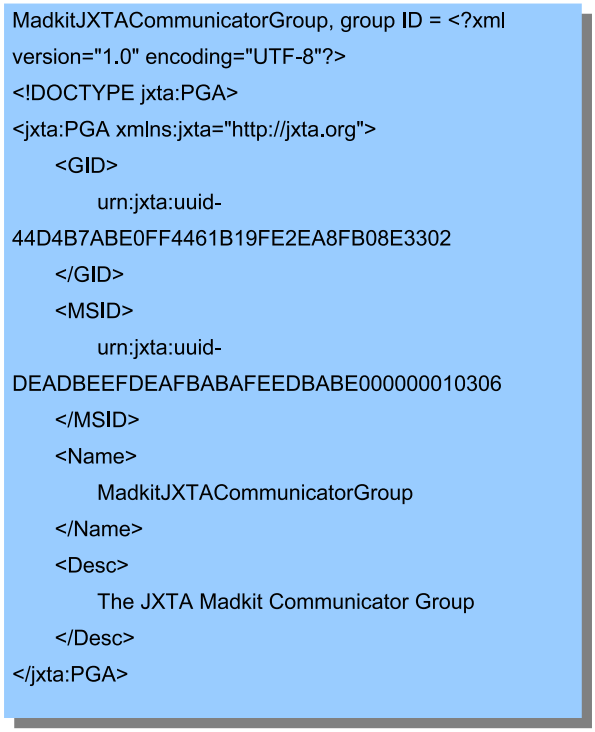

Fig. 7.7 – L'Advertisement du groupe MadkitJXTACommunicatorGroup.

le publier dans le RendezVous auquel il est connecté et rester à l'écoute des demandes de connexion des anciens JXTACommunicator existants dans le groupe MadJXTComGrp. 6) Le RendezVous signale l'arrivée d'un nouveau membre à tous les « JXTACommunicator » connectés. 7) Les JXTACommunicator existants récupèrent l'Advertisement du « BiDiPipeServer » du nouveau JXTACommunicator à qui ils envoient des demandes de connexions. 8) Accepter les demandes de connexions reçues des JXTACommunicator déjà existants et établir une connexion avec chaque demande de connexion en utilisant un « JXTABiDiPipe ».

La figure ci-dessous illustre la mise en place du réseau de connexion entre les différents membres JXTACommunicator du groupe MadJXTComGrp. Dans ce cas, il y a deux plateformes connecté au réseau en plus de la plateforme native qui crée le groupe et lance le premier agent RendezVous.

Si on analyse la structure ainsi établie on peut dire que l'entrée de toute nouvelle plateforme déclenche un processus de connexions en cascade. Pour relier un nouveau noyau au réseau existant, deux possibilités sont possibles. La première solution consiste à ce que ça soit le nouveau JXTACommunicator qui récupère la liste des Advertisement des « BiDiPipeServer » existants sur le réseau et établi la connexion avec chacun des JXTACommunicator connectés. Cette solution surchargerai énormément le nouveau JXTACommunicator et risque de le saturer. En effet, si on considère qu'il existe n plateformes dans le groupe, le nombre de messages reçus et envoyés par le nouveau JXTACommunicator sera comme suit :

n réponses d'Advertisement des « BiDiPipeServer » reçus suite à la demande de recherche. n demandes de connexions envoyés à tous les JXTACommunicator existants.

L'agent JXTACommunicator doit donc recevoir quasiment en même temps n messages et établir n connexions ce qui est trop pour un seul Agent en un temps réduit.

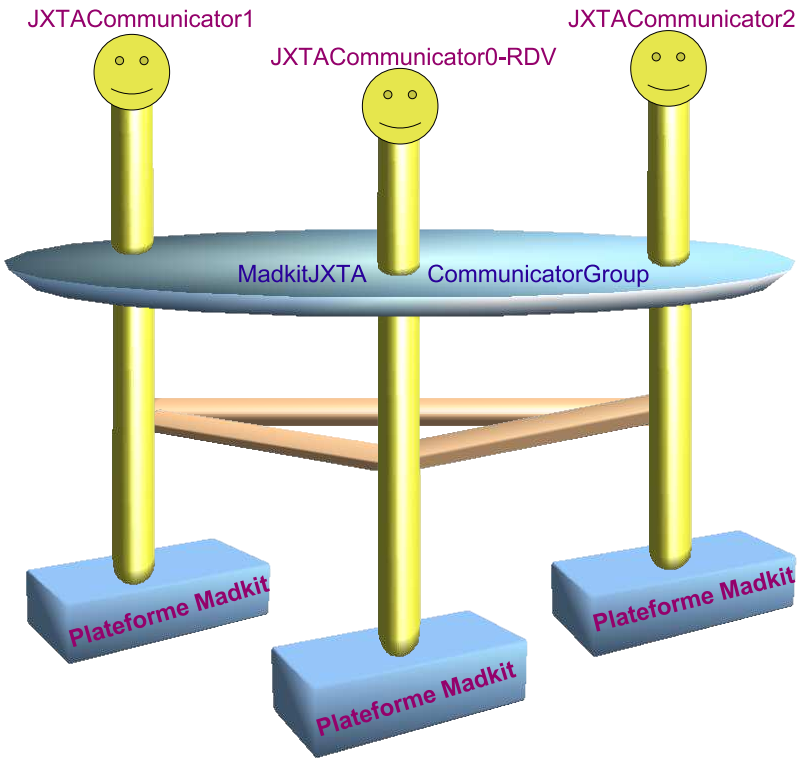

FIG. 7.8 – L'interconnexion entre les JXTACommunicator du groupe MadJXTComGrp.

La deuxième solution, celle que nous avec adopté, consiste à inverser le processus. C'est les JXTACommunicator déjà sur place qui découvrent l'arrivée d'un nouveau membre, et lui envoient des demandes de connexions. Pour le nouveau JXTACommunicator on divise par deux le nombre de messages à traiter, il ne fait que recevoir les demandes de connexions et établi les connexions. Pour les JXTACommunicator existants cela se traduit par une recherche de l'Advertisement du « BiDiPipeServer » et établir la connexion avec le JXTACommunicator. L'opération de propagation qu'effectue le RendezVous, pour informer les membres connectés de l'arrivée d'un nouveau membre, n'est pas coûteuse (en terme d'envoi de messages), elle s'effectue par un mécanisme d'événements proposé par JXTA (RendezVousEvent).

On pourrait rétorquer, à juste titre, que l'interconnexion de tous les noyaux du groupe va conduire à un alourdissement du système. C'est en effet le cas, mais il faut rappeler que ce groupe est un cas particulier et que l'interconnexion entre les différents noyaux existants est un choix d'implémentation. D'autre part, les utilisateurs qui désirent entrer dans le Mad-JXTComGrp, peuvent modifier leurs stratégies et refuser les demandes de connexion aux JXTACommunicator connectés. Pour les groupes publics, selon les besoins du concepteur, il pourra imposer ou non une interconnexion totale entre les différents JXTACommunicator.

Mais ce qui fait essentiellement la force de ce mécanisme d'interconnexion générale, c'est le choix de voir dans cette approche, non pas un élément négatif car ça alourdit le système, mais bien le biais d'une gestion solide de l'organisation d'agents globale. Ceci conduit alors à l'étude des mécanismes de coordination et de synchronisation entre les différents noyaux, pour la gestion des groupes d'agents auxquels appartiennent les agents qu'ils accueillent.

7.4.3 Gestion du niveau organisationnel

Jusqu'ici, nous avons expliqué comment sont formées et gérées les organisations de pairs (les PeerGroup), mais nous n'avons pas expliqué comment se reflète l'organisation agents au niveau de la couche P2P. On peut dire que ce qu'on a fait c'est de bâtir les liens physiques entre les noyaux, il nous faut donc des mécanismes pour construire, au dessus de ces liens, l'activité sociale des différents agents de ces plateformes. Cette activité sociale se base sur le modèle AGRS. Pour mettre en oeuvre ces mécanismes, nous basons notre travail sur la définition d'un Advertisement qu'on a crée (un « Custom Advertisement ») qui décrit les groupes et qu'on appelle « GroupAdv ».

Description d'un groupe d'agents et recherche de services

Le concept de service dans le modèle AGRS est la colle qui relie les agents entre eux. Toute l'activité des agents est exprimée en termes de services. Nous adoptons cette philosophie, autant pour fournir un modèle organisationnel, que pour le fonctionnement interne du système. Les agents du système interagissent selon le principe d'offre et demande de services dans le cadre des rôles. Cependant, nous exportons l'organisation des agents du niveau local (la plateforme) vers l'extérieur (sur le réseau) et ce à traavers la description du groupe. Ce qu'on publie au niveau du réseau ce ne sont pas les services offerts par les agents et structurés par les rôles, mais plutôt le contenu de tout un groupe.

Pourquoi ce choix, par rapport à un modèle où tout est exprimé en terme de services ? Principalement par soucis de conformité à la vision globale que nous avons présenté dans la figure (figure 4.11). Dans cette vision, notre intérêt ne porte pas uniquement sur le fonctionnement interne des agents, mais plus sur ce que perçoivent les agents qui sont à l'extérieur du groupe. Ce qui intéresse les agents externes c'est de savoir quels sont les groupes qui contiennent les rôles offrant des services qui les intéressent. On pouvait imaginer de faire l'inverse et commencer par le bas : publier les services en indiquant pour chaque service, à quel rôle est rattaché, et dans quel groupe ce rôle appartient. Mais cela n'aurait pas été faisable, il y aurait un grave problème d'efficacité. En effet, on aura en circulation sur le réseau autant de d'Advertisement que de services, ce qui est énorme pour des systèmes ouverts où le nombre de groupes peut être compté par dizaines, le nombre d'agents par centaines et donc le nombre de services serait par milliers.

Il nous reste à voir comment exprimer, au sein d'une description de type Advertisement, la composition d'un groupe. Nous allons donc reprendre les propriétés nécessaires pour un agent externe et comment les exprimer dans le cadre d'un Advertisement de type JXTA. Un groupe appartient à une communauté. Donc en plus du nom du groupe la description doit contenir en premier lieu le nom de la communauté dans laquelle le groupe est défini. L'ensemble nom du groupe + nom de la Communauté, constitue un identifiant unique. En effet, le nom de la communauté se traduit par le nom d'un PeerGroup au niveau du réseau JXTA, or la première vérification qu'effectue tout JXTACommunicator lors de la création d'un PeerGroup est de vérifier qu'il n'existe pas un groupe au même nom sur le réseau. De même au niveau de Madkit, on vérifie avant la création d'un groupe d'agents qu'il n'y a pas risque de duplication. Le troisième élément à décrire est les rôles du groupe. Il faut énumérer les rôles que propose le groupe et qui seront l'interface régulant l'activité des agents. Chaque rôle est décrit par les services qu'il offre aux agents utilisateurs, mais aussi par les services requis pour le jouer et ceux qu'il fourni à ces agents joueurs. Conformément aux spécifications du modèle AGRS, certains des services fournis sont transférables aux agents joueurs, il est

aussi nécessaire de spécifier ces services dans la description des groupes. La figure ci-dessous, illustre un exemple simple d'un groupe nommé « Calc » appartenant à une communauté qui s'appelle « MathComm ». Le groupe contient un seul rôle « Calculus » qui offre les services « Add, Minus, Mul Div » et ne requière aucun service.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE GroupAdv>
<GroupAdv xmlns:jxta="http://jxta.org">
    <CommunityName>
        MathComm
    </CommunityName>
    <GroupName>
        Calc
    </GroupName>
    <GroupFullName>
        MathComm.Calc
    </GroupFullName>
    <ExistingRoles>
        |Calculus|
    </ExistingRoles>
    <OfferedServices>
        |Add,Minus,Mul,Div,|
    </OfferedServices>
    <RequiredServices>
        ||
    </RequiredServices>
    <ProvidedServices>
        ||
    </ProvidedServices>
    <TransferableServices>
        ||
    </TransferableServices>
    <SiteAgent>
        Mansour:K1184795990406
    </SiteAgent>
</GroupAdv>
```

Fig. 7.9 – Exemple d'une description d'un groupe présenté sous la forme d'un Advertisement.

La description des rôles et des services, avec l'utilisation des caractères spéciaux "|" et "," n'est pas le fruit du hasard, elle puise ses raisons d'un problème technique lié aux mécanisme de recherche proposé par JXTA. Le mécanisme de découverte d'Advertisement de JXTA se base sur le concept de « Tags ». Chaque type d'Advertisement a un ensemble de champs qui permettent son indexation et facilitent la recherche. Toute requête de découverte doit porter sur un de ces champs et le résultat dépendra de la valeur du champ cherché. Si aucun des Advertisement publiés sur le réseau ne contient la valeur recherchée pour le champ ciblé, le résultat sera vide. Ce mécanisme rend impossible une représentation simple, où chaque service occupe un champ à part. Cela produira une confusion sur les champs à indexer dans la mesure où tous les services auront le même nom de champ ("Service" par exemple). Afin de surmonter cette difficulté, nous avons adopté la structuration par type de service (offert, fournis, requis ou transférable). La position des rôles dans la liste des rôles existants est importante. Pour découvrir les services offerts du premier rôle, il faut extraire les premiers services du champ (offered Services) entre les deux séparateurs "|", pour les services du deuxième rôle, il faut extraire la deuxième série de services entre les séparateurs etc. On fait de même pour les autres

types de services (fournis, requis ou transférable). Sachant que JXTA permet l'utilisation des caractères Jokers "*", pour remplacer des caractères dans la valeur du champ recherché, nous réussissons à construire des requêtes capables de trouver la liste des groupes offrant un service dont on connaît le nom. Mieux encore, on peut lancer des recherches portant sur plusieurs services en utilisant le Joker "*", cela permet de raffiner la recherche aux groupes proposant plusieurs services qui intéressent l'agent. Afin de mieux comprendre ce mécanisme, essayons d'analyser un exemple réel d'une requête portant sur le groupe présenté dans la figure 8.8 :

grp.getDiscoveryService().getRemoteAdvertisements(null, DiscoveryService.ADV, "Offered-Services", ""+"Mul"+"*",10, this) ;*

La requête est lancée pour trouver les dix premiers Advertisement, contenant dans leurs champs de type « OfferedServices » une suite de caractère contenant « Mul ». Lors de la réception des résultats les JXTACommunicator ne retiennent que les Advertisement contenant exactement le service recherché (Mul ici). Il va sans dire que ces détails techniques ne sont pas visibles pour l'utilisateur final ni pour les agents d'ailleurs. Ces derniers expriment leurs besoins en terme de services, par le biais de méthodes qu'on fourni à tous les agents de la plateforme. Pour clarifier le mécanisme de recherche et les interactions entre agent demandeur, noyau, SiteAgent, JXTACommunicator et RendezVous, nous avons schématisé ce processus présenté dans la figure suivante :

L'identificateur du service indique l'ontologie dont il fait partie et le chemin complet menant au service : (Nom du rôle, Nom du groupe, Nom de la communauté). A la réception de la liste des identificateurs de services correspondants à sa recherche, l'agent demandeur sélectionne le service qui correspond le mieux à ses besoins et demande d'entrer dans le groupe, contenant le rôle offrant le service recherché. Cette demande ne pose aucun problème si le noyau appartient déjà à la communauté contenant le service et que contient une copie du groupe. Par contre, si le service est offert par un rôle se trouvant dans un groupe qui ne fait pas partie des groupes connus par le noyau, il faudra entamer une procédure pour importer le groupe conformément au modèle AGRS.

Pour ce faire, il faudra se connecter au noyau distant offrant le service et contenant le groupe. C'est ce qui explique la signification du dernier champ de l'exemple de la figure 8.8, le SiteAgent. Ce champ indique quel est le noyau qui a publié l'Advertisement. Cela nous conduit à examiner plus en détails ce qui se passe à la réception des réponses d'une recherche par l'agent demandeur et comment exploiter l'identificateur du service pour importer le groupe.

Importer un groupe

Pour jouer ou utiliser un rôle, l'agent doit appartenir au groupe contenant ce rôle. Conformément au modèle de gestion proposé, chaque noyau doit maintenir une copie de tout groupe auquel appartient un des agents qu'il accueille. C'est les SiteAgent qui se chargent de maintenir et synchroniser les organisations. Si un agent désire entrer dans un groupe que le noyau ne contient pas, ce dernier doit l'importer. Pour importer un groupe, le noyau doit le demander à son SiteAgent en lui indiquant l'identifiant du groupe (nom du groupe + nom de la communauté) et l'adresse du SiteAgent qui en est responsable (l'adresse est dans l'identifiant du service sélectionné par l'agent). Deux cas de figures se présentent :

Le premier cas : Si le noyau appartient déjà à la communauté contenant le groupe. Si la communauté se base sur la connexion totale entre les différents JXTACommunicator, comme c'est le cas de la communauté MadJXTComGrp, il ne reste au SiteAgent qu'à demander au

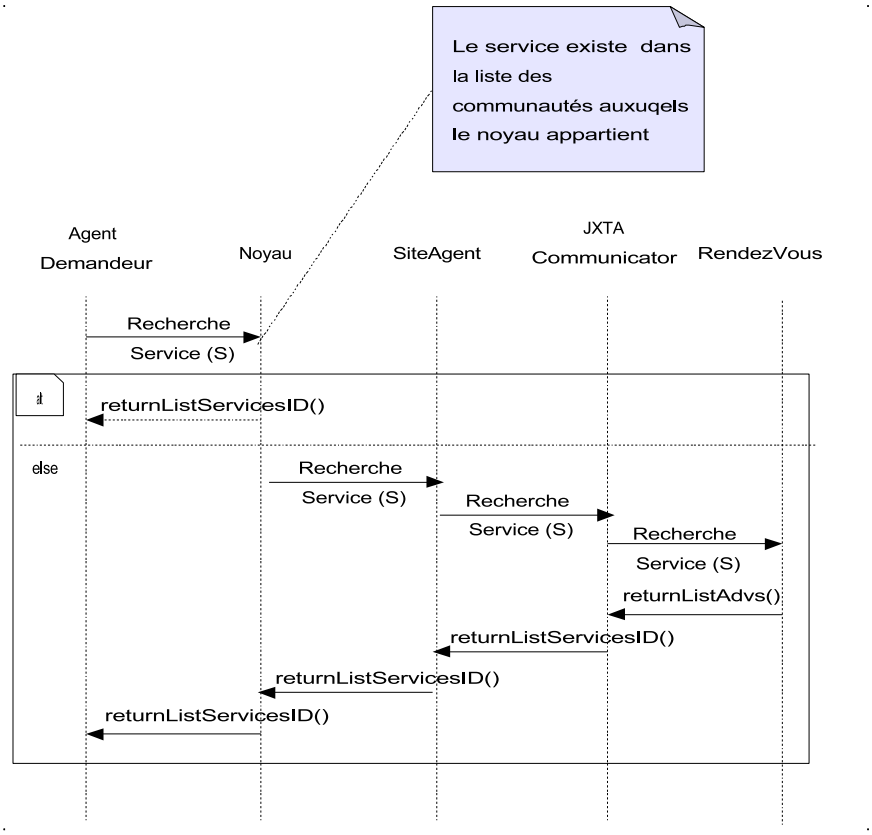

FIG. 7.10 – Processus de recherche d'un service.

SiteAgent distant d'importer le groupe. Ce sont les JXTACommunicator qui assurent la communication entre les SiteAgent. Le groupe étant un objet, nous nous basons sur le mécanisme de la sérialisation [] pour l'envoi et la réception des copies de groupes. Si la connexion n'est pas totale entre les différents JXTACommunicator des noyaux d'une communauté, des mécanismes de découvertes, des JXTACommunicator, sont utilisés pour trouver le Pair représentant le noyau ayant l'adresse du SiteAgent passée en paramètre.

Par contre, si le noyau n'appartient pas à la communauté qui contient le groupe recherché, il faudra la rechercher et ensuite y appartenir. Cette phase se fait comme décrit précédemment dans l'exemple du groupe public MadJXTComGrp. Une fois le noyau entre dans la communauté, le processus suit le premier cas.

Un point délicat demeure malgré tout dans le processus d'importation : Que faire si le noyau qui a publié la description du groupe, qui a été découverte, n'est plus connecté ? Il serait bien évidemment impossible d'effectuer l'opération d'importation du groupe à partir de ce noyau. Il nous faut donc un mécanisme pour traiter ce problème, sans pour autant relancer dès le début la recherche au niveau de l'agent demandeur du service qui a amené au déclanchement de ce processus d'importation. La solution consiste à ce que l'agent demandeur sauvegarde la liste des identifiants de services qu'il a obtenu comme réponse à sa requête, jusqu'à son entrée dans le groupe qui contient le rôle qu'il désire jouer (ou utiliser). Si l'opération d'importation échoue, l'agent demandeur sélectionne un autre identifiant de service et demande au noyau d'entrer dans le groupe qui le propose. L'opération se répète jusqu'à la réussite d'une opération d'importation, ou que la liste contenant les réponses soit épuisée.

Assez logiquement, après avoir vu comment rechercher un service puis entrer dans le groupe qui le contient, il nous faut maintenant se pencher sur l'utilisation concrète d'un service dans la plateforme, selon l'architecture implémentée et conformément au modèle AGRS proposé.

Utilisation d'un rôle

Une fois l'agent qui cherche un service entre dans un des groupes le proposant, il demande à utiliser le rôle proposant le service. La gestion de cette demande se fait localement, au niveau du noyau accueillant l'agent. C'est le point fort du mécanisme de gestion, par copies de groupe, proposé. La gestion du groupe se fait en local et il n'y a pas besoin d'établir une connexion avec un noyau distant pour la faire. Si l'agent demandeur répond aux critères d'entrée au groupe et d'utilisation du rôle, sa requête est acceptée et devient utilisateur du rôle. Ce n'est qu'à cet instant qu'il pourra demander à obtenir un service offert par le rôle. L'instance du rôle sélectionne, selon la stratégie conçue par le concepteur, un agent joueur qui aura la responsabilité d'exécuter le service demandé par l'agent utilisateur. Si aucun agent joueur n'existe au moment de la demande, l'agent utilisateur devra soit attendre, soit quitter le rôle pour en trouver un autre capable de fournir le service recherché. L'agent utilisateur du rôle obtiendra un identifiant de l'instance « AgentInRole » correspondant à ce rôle et qui rattache l'agent joueur au rôle, comme le spécifie le modèle AGRS. D'une façon générale, deux cas de figures se présentent : l'agent joueur se trouve sur le même noyau que l'agent utilisateur ou chacun d'eux se situe sur un noyau différent.

Dans le premier cas, les choses sont plus simples. L'agent utilisateur du service utilise l'identifiant de l'instance « AgentInRole », qu'il a reçu, pour envoyer sa requête à l'agent joueur. Nous mettons en oeuvre dans la classe de base des agents de la plateformes les primitives nécessaires pour demander un service. Comme toutes les opérations de la plateforme Madkit, les demandes de services passent par un envoi de message à l'agent joueur.

```
grp.getDiscoveryService().getRemoteAdvertisements
(null,DiscoveryService.ADV,"OfferedServices","*"+
"Mul"+"*",10, this);
```

Fig. 7.11 – Un exemple réel d'une requête de recherche de services.

C'est ce mécanisme d'envoi de messages qui permet une abstraction et une transparence complète par rapport au lieu d'exécution de l'agent joueur. L'instance « AgentInRole » se situe dans la plateforme où se trouve l'agent à qui elle est rattachée. L'agent utilisateur du rôle n'a pas à savoir où se trouve cette instance ni l'agent qui lui fournira le service. Le noyau délivrera le message à son SiteAgent dès qu'il découvre que l'agent joueur n'est pas sur sa plateforme. Le SiteAgent transfère le message au JXTACommunicator qui se chargera de le délivrer à son homologue représentant la plateforme accueillant l'agent joueur. L'agent joueur reçoit la demande d'exécution, exécute le service et renvoie le message à l'agent utilisateur.

Selon les choix du concepteur, la demande et l'exécution d'un service, peut nécessiter des opérations de sauvegarde, récompense ou pénalités pour l'agent joueur ou un coût pour l'agent utilisateur...

Jouer un rôle

Nous avons jusqu'ici un point de vue utilisation du rôle, en étudiant comment l'agent cherche un service, entre dans le groupe contenant le rôle l'offrant, demande l'utilisation du rôle et requière puis reçoit la réponse à sa demande d'exécution du service. Néanmoins, la partie agents joueurs du rôle et qui fournissent les services demandés par les agents utilisateurs n'est pas encore traitée. Cela implique donc d'analyser et de voir comment, au niveau de la plateforme, les agents jouent les rôles. Cette analyse peut être décomposée en trois phases. En premier lieu, l'agent doit choisir les rôles à jouer. Comme pour l'utilisation, et à la différence d'autres plateformes, les agents peuvent chercher les rôles à jouer en plus de ceux qu'ils jouent à leurs créations, imposés par le concepteur. L'agent lance une recherche de services et de découverte de rôles qui requièrent ou transfèrent des services. Le mécanisme de recherche est sensiblement similaire à celui de l'utilisation des rôles. La requête passe de l'agent au noyau qui vérifie s'il ne peut pas répondre à la demande de l'agent. En cas d'échec, le noyau transfère la requête à son SiteAgent qui l'envoi à son tour au JXTACommunicator. Ce dernier, formule la requête et l'adapte au système de recherche de JXTA, puis la diffuse sur le réseau par l'intermédiaire des RendezVous auxquels il est connectés.

A la réception d'une réponse positive concluante à sa recherche, l'agent demande à son noyau d'entrer dans le groupe accueillant le rôle qui propose le service recherché. Comme pour l'utilisation d'un rôle, le noyau entame le processus d'importation d'une copie du groupe. Si tout se passe correctement, l'agent entre dans le groupe et demande de jouer le rôle. Chaque copie de groupe décide selon ses règles et l'état général du groupe d'accepter ou non la demande de l'agent de jouer le rôle. A l'acceptation de cette demande, un objet de type AgentInRole est créé et retourné à l'agent qui l'ajoute à la liste des agents qu'il joue.

La troisième phase correspond au traitement des demandes d'exécution de services liés au rôle. Bien que nous essayions d'être toujours de fidèles au modèle AGRS proposé, le scénario présenté à la figure 4.8 ne se déroule pas exactement de la même façon au niveau de la plateforme. En effet, lors de la phase de conception, dans la figure 4.8 c'est l'entité AgentInRole qui est responsable de la réception de demande de services et du renvoi des réponses à ces demandes. Cela n'est pas ce qui se passe réellement, les entités AgentInRole ne sont pas des agents et ne peuvent donc pas recevoir ni émettre des messages. C'est le noyau qui intercepte les messages de demande de service et de réponse et en informe l'entité AgentInRole impliqué, avant de les transférer à l'agent joueur ou utilisateur du rôle. L'entité AgentInRole conformément aux spécifications modèle AGRS, peut sauvegarder ces échanges et infliger des sanctions aux agents qui ne respectent pas les règles du rôle, récompenser l'agent joueur pour le service rendu ... C'est au concepteur de définir les actions à entreprendre par l'entité AgentInRole en conséquence à la suite d'une activité dans le cadre du rôle. Pour ce faire le concepteur doit surcharger la méthode que nous proposons :

```
protected void saveOperation(int type, String
servName, AgentAddress user){
}
```

Fig. 7.12 – La méthode permettant de sauvegarde les écahnges entre agents joueur et utilisateur.

L'entité AgentInRole est le lien entre le rôle et l'agent joueur. En plus de sauvegarder et de réagir aux activités que cadre le rôle, elle remplace les interactions entre le rôle et l'agent joueur.

L'entité AgentInRole

Selon le modèle AGRS, trois types de services composent le rôle. Les services offerts Un agent joueur peut utiliser les services fournis et transférables. Un agent joueur, peut utiliser les services fournis par le rôle pour répondre à une demande d'un agent utilisateur, ou même pour ses propres besoins. Comme nous l'avons précédemment mentionné, à partir du moment que l'agent joueur se trouve rattaché à son entité AgentInRole il n'a plus de contact direct avec le rôle qu'il joue. Afin de faire usage d'un service fourni par le rôle, il doit donc passer par l'entité AgentInRole. L'agent spécifie le service fourni qu'il désire invoquer et l'entité AgentInRole vérifie que le demandeur est bien l'agent auquel elle est rattachée, vérifie que le service requis est un service fourni, exécute le service et retourne le résultat à l'agent joueur.

De même, l'agent joueur peut demander le transfert d'un service transférable. Il s'agit d'un transfert d'objet. Les services au niveau de l'entité AgentInRole sont des objets. A la réception d'une telle demande et après vérification de l'identité de l'agent demandeur, l'entité AgentInRole fait une copie du service demandé et la retourne à l'agent joueur. Ce dernier pourra l'ajouter à la liste des services qu'il possède et l'utiliser ultérieurement indépendamment du fait qu'il joue ou non le rôle.

Lorsqu'une modification a lieu au niveau du rôle, comme par exemple l'ajout ou la suppression d'un service, il faut que tous les agents joueurs soient informés en temps réel de ces modifications. Le rôle met à jour ses entités AgentInRole. Chacune de ces entités informe son agent joueur de ces modifications par l'intermédiaire du noyau.

L'activité des agents dans le groupe peut modifier sa structure, les règles qui le régissent peuvent exiger une mise à jour. Le problème qui se pose est comment se fait cette mise à jour alors que le groupe est distribué et partagé en différentes copies. C'est le sujet du prochain paragraphe.

Gestion d'un groupe

Chaque groupe se trouve implanté sur différents noyaux du réseau. Chaque modification nécessitant une mise à jour, doit se répercuter en même temps sur les différentes copies du groupe afin de garantir sa cohérence. Le mécanisme de mise à jour est le suivant : Dès qu'un événement au niveau d'une copie du groupe exige une mise à jour, le noyau reçoit un message de mise à jour avec la liste des noyaux implémentant ce groupe. En effet, chaque copie du groupe contient la liste des noyaux sur lesquels le groupe est distribué. Le noyau transfère le message au SiteAgent qui formule une requête de mise à jour à ses homologues dans les différents noyaux implémentant le groupe. Mais reste le problème de la mise à jour de l'Advertissement sur le réseau. Chaque groupe diffuse son propre Advertissement (pour garantir une égalité de traitement des demandes d'entrée au groupe). A la réception d'une mise à jour, selon la politique de chaque groupe, deux solutions sont possibles. Soit le groupe exige la diffusion immédiate de toute modification au niveau de la structure du groupe, auquel cas la copie du groupe demande au noyau la mise à jour de son Advertissement. La deuxième solution se base sur les temps de vie des Adverttisement et privilégie une mise à jour cyclique. Dans cette solution seule la copie qui a été modifiée effectue une mise à jour de son Advertisement, les

autres le feront automatiquement à la fin du temps de vie de leurs Advertisement. C'est un compromis à faire entre exactitude des informations sur les copies de groupe existantes sur le réseau et la surcharge des noyaux (l'envoi et mise à jour consomme en temps et ressources) et du réseau.

Le système de gestion des groupes ainsi fait est totalement distribué, aucune copie du groupe n'est indispensable à la vie du groupe. Néanmoins, certains problèmes de cohérence peuvent survenir. La qualité des connexions n'étant pas garanti, des messages de mise à jour peuvent arriver en retard ou ne jamais arriver. Cela risque de donner des copies du groupe incohérentes et donc un fonctionnement des agents incorrect. Ce problème est un problème classique dans les systèmes distribués [] et plusieurs solutions existent et réussissent à le résoudre en partie[]. Nous laissons la liberté aux concepteurs de groupes d'instaurer les procédures qu'ils jugent conformes au contexte dans lequel leurs applications sont déployés (qualité des connexions, temps des mise à jour, fréquence des mise à jour). Cependant, nous effectuons au niveau du noyau des mises à jours automatiques, basées sur un temps de mise à jour que le concepteur peut spécifier au lancement de son application.

7.5 Exemple d'application

Afin de mieux cerner comment travailler avec l'approche AGRS implémenté par la plateforme Madkit, nous avons choisi de présenter un exemple simple mais qui permet de faire le tour sur les différents concepts de notre modèle.

Prenons l'exemple d'un rôle de calcul (CalculusRole) joué par des agents spécialistes en mathématique et d'autres qui cherchent des services de calcul. La première chose à faire est de définir les services qui composent le rôle et les répartir selon les trois catégories : Services offerts, fournis et transférables. Pour l'exemple, nous supposons que le rôle CaluclusRole est formé de quatre services : (Add, Minus, Mul, Div). Les trois premiers sont des services offerts, le service Div est à la fois un service offert, fourni et en même temps transférable. Une fois les services du rôle définis, le concepteur peut alors commencer la première phase d'implémentation. La première étape n'est pas, comme ce qu'on peut le croire, l'implémentation de la classe du Rôle, mais plutôt la classe RoleExecution. En effet, chaque rôle doit avoir une classe RoleExecution qui permet de le rattacher à ses agents joueurs. Si aucune classe de type RoleExecution n'est créée par le concepteur le système utilise une instance de la classe RoleExecution par défaut. Cette classe n'a rien d'autre que l'implémentation de la classe AgentInRole du modèle AGRS. Dans le cas de cet exemple, le CalculusExecution surcharge la méthode « saveOperation ». Cette méthode est appelée automatiquement à chaque demande ou réponse de service. Aux développeurs de programmer l'activité à faire suite à ces opérations.

Dans la deuxième étape, il s'agit de coder la classe du rôle CalculusRole, seul le service Div est à implémenter par le concepteur, les autres services sont à la charge des agents joueurs. Le code est assez simple, dans la mesure où l'essentiel du travail est fait par la plateforme par la classe mère RoleExecution. Il faut ajouter les services en mentionnant à chaque fois le type du service. Par défaut, sans aucune mention particulière, le service est un service offert.

Maintenant, il ne reste plus au concepteur que d'ajouter ce rôle dans un groupe donné. Cela peut être fait par n'importe quel agent ayant les autorités d'ajouter et modifier la structure du groupe. Le code nécessaire pour ce faire peut être comme suit :

```
public class CalculusExecution extends
RoleExecution {
private Hashtable <Integer, Vector>
savedOperations;
int codeOperation=0;
public CalculusExecution(){
   super();
   savedOperations = new Hashtable<Integer,
Vector>(0,0.75f);
}
public CalculusExecution(AgentAddress
playerAgent, RoleExecutionID roleID,
Vector<String> offeredServices,
Hashtable<String, Service> providedServices)
{
   super(playerAgent, roleID, offeredServices,
providedServices);
}
protected void saveOperation(int type,String
serviceName, AgentAddress user){
   codeOperation++;
   Vector operation = new Vector(3);
   operation.add(type);
   operation.add(serviceName);
   operation.add(user);
   savedOperations.put(codeOperation,
operation);
System.out.println("Saved Operation
"+codeOperation);
   }
}
```

FIG. 7.13 – Première étape de la classe CalculusExecution.

```
public CalculusRole(String name, GroupID
groupID, String roleExecutionClassName) {
   super(name, groupID);
   Service ping = new Service();
   this.addService("madkit.games.Add");
   this.addService("madkit.games.Minus");
   this.addService("madkit.games.Mul");

this.addService(Offered_Transferable_Service,"m
adkit.games.Div");
}
public int Div(int a, int b){
      return (a/b);
   }
}
```

FIG. 7.14 – Deuxième étape de la classe CalculusExecution.

```
CommunityID communityID =
this.createServiceableCommunity(communityName);
ServiceableGroup grp = new
ServiceableGroup(communityID,"Calc",null);
CalculusRole calculusRole = new
CalculusRole("Calculus",
grp.getID(),"madkit.games.RoleExecution");
grp.addServiceableRole(calculusRole);
this.createServiceableGroup(grp, null);
```

FIG. 7.15 – Troisième étape de la classe CalculusExecution.

Supposons maintenant que le rôle existe, est ajouté au groupe qui est publié sur le réseau. Tout agent peut alors le découvrir, et essayer de le jouer. Nous proposons un agent nommé Calculus, cet agent implémente les services offerts par le rôle (Add, Min, Mul), cherche

7.6 Conclusion

Au cours de ce chapitre nous avons présenté l'implémentation du modèle AGRS. Notre solution repose sur l'utilisation de la plateforme JXTA pour permettre le déploiement de la plateforme Madkit dans le cadre des réseaux ouverts et plus particulièrement sur Internet. D'une part, nous nous appuyons sur le concept d'agentification de services utilisé dans Madkit pour mettre en place l'outil de communication distribuée le mieux adaptée à l'environnement dans lequel est plongée la plateforme. Ainsi, on peut utiliser un agent offrant la communication à bases de Sockets dans un réseau fermé, et changer dynamiquement pour utiliser notre solution JXTA quand l'application communique sur Internet. D'autre part, l'encapsulation des fonctionnalités JXTA (communication, publication, et recherche) dans un agent simplifie le déploiement sur Internet de la plateforme Madkit, par délégation de la gestion de la distribution à la couche JXTA. Enfin, l'introduction du concept de service au modèle AGRS et son implémentation sur Madkit, s'adapte bien avec la vision P2P (publication et recherche de service).

Nous avons expérimenté l'architecture hybride que nous proposons sur les différentes applications distribuées de Madkit. Différents noyaux s'interconnectent à travers Internet et les groupes de chaque noyau publient leurs services. Ces services sont découverts par les agents des différentes plateformes et utilisés selon leurs besoins.

L'architecture hybride proposée (intégration des techniques P2P pour la répartition des applications multi-agents) semble intéressante dans les domaines comme la grille de calcul ou les composants. La possibilité de faire des recherches d'information distribuée dans des réseaux ouverts et à grande échelle associée aux caractéristiques des systèmes multi-agents peut résoudre plusieurs des problèmes de ces domaines. Au cours du prochain chapitre nous présentons une utilisation de notre modèle et de notre architecture pour résoudre un des problèmes du processus de la e-Qualification.

```
public class CalculsAgent extends
ServiceableAgent {
String communityName = "MathComm";
public CalculsAgent(){
   addService("Add");
   addService("Mul");
   addService("Minus");
}
public void activate(){
this.requestUseRole(communityName,"Calc","Calculu
s", null);
RoleExecution rl =
this.requestPlayRole(calculusRole.getID(), null);
}
void handleMessage(Message m){
if (m instanceof ServiceableMessage){
println("message:
"+((ServiceableMessage)m).getServiceFullName());
this.handleServiceableMessage((ServiceableMessage
)m);
}
else
println("Non ServiceableMessage receieved"+m);
}
public void live()
{
   this.pause(2000);
  while(true){
   Message msg = this.waitNextMessage();
   handleMessage(msg);}
}
public int Add(Integer a, Integer b){
return (a+b);}
public int Mul(Integer a, Integer b){
   return (a*b);}
public int Min(Integer a, Integer b){
   return (a-b);}
```

Fig. 7.16 – Code de l'agent CaculusAgent.

Chapitre 8

Utilisations du modèle AGRS dans le processus de la e-Qualification

DANS ce chapitre nous allons présenter les possibilités d'utilisation du modèle AGRS et de son implémentation dans le processus de e-Qualification. Comme nous l'avons introduit au chapitre 2, la problématique initiale de ce travail de thèse était la résolution de certains problèmes rencontrés lors du traitement du problème de qualification du e-Learning. Notre intérêt s'est ensuite porté sur le problème général des groupes d'agents ouverts et dynamiques. Nous avons tout de même participé activement dans les travaux autour du problème de la e-Qualifcation en apportant notre pierre à l'édifice. Nous ne prétendons pas avoir résolu tous les problèmes de la e-Qualification, notre objectif est de participer sur les axes qui concernent la gestion des groupes ouverts et dynamiques d'agents, la recherche de services et interactions entre agents. Nous exposons au cours de ce chapitre notre participation à travers certains travaux [Gouardères *et al.* , 2005] et [Lacouture & Mansour GESM07].

L'objectif principal du processus de e-Qualification est de proposer une réponse à deux questions de recherche : « Comment concevoir un processus de e-Qualification operationnel dans un environnement d'apprentissage basé sur les compétences ? » [Kreijns & Kirschner, 2001] et « Comment l'apprentissage est réellement validé durant la session d'apprentissage ? » [BRSJ03]. Comme nous l'avons mentionné au cours du 2, cela nous a poussé à cadrer le processus de e-Qualification dans l'apprentissage collaboratif et utiliser la stratégie de réciprocité. Cela nous a conduit à introduire un nouvel outil permettant de guider et suivre l'activité des apprenants. Il s'agit du e-Portfolio [Barret, 2001] [Razmerita *et al.* , 2005a] un vrai espace partagé sur lequel se base en grande partie le processus d'e-Qualification...

8.1 Le e-Portfolio comme un service de grille adaptable

8.1.1 Introduction

Le terme Portfolio fut longtemps utilisé dans le domaine de l'apprentissage pour signifier l'emplacement où l'étudiant sauvegarde les meilleurs résultats et les documents les plus importants [Barret, 2004]. Le concept évolua avec l'émergence des nouvelles technologies et fut adopté par le domaine du e-Learning pour devenir le e-Portfolio (pour Electronic-Portfolio) [Vanides & Morgret, n.d.]. Malgré d'énormes hésitations concernant l'apport que peut apporter le Portfolio à l'apprentissage, le concept a suscité par la suite beaucoup plus d'intérêt et

plusieurs recherches ont confirmé son apport tant aux appreneurs qu'aux apprenants. Dans le domaine du e-Learning le e-Portfolio peut être utilisé pour : 1) Organiser la présentation du travail fait, 2) offrir un contexte de discussion, 3) présenter des révisions et réactions des enseignants, 4) confirmer le progrès et les réalisations de l'apprenant, 5) reconnaître les capacités de l'apprenant et prendre les décisions.

Ainsi, l'analyse et le partage de l'e-Portfolio peuvent aider à améliorer l'intelligence collective des étudiants et à faciliter leur acquisition de nouvelles connaissances [Razmerita *et al.* , 2005b].

Alors que dans la version ancestrale du e-Portfolio (le protfolio) c'était à l'apprenant de faire tout le travail d'enrichissement et de mise à jour, cela a évolué avec l'apport d'entités externes à savoir les agents intelligents. En effet, la complexité des applications du e-Learning qui ne cesse de s'accroître a accéléré l'utilisation des agents intelligents. Ces agents ont pris en charge les tâches les plus délicates et pouvant être automatisés concernant la gestion des e-Portfolio : Extraction des données, mise à jour... Notre idée à long terme est d'étendre l'utilisation de ce concept pour dépasser le seul domaine du e-Learning et contenir toute application qui met en relation directe des agents intelligents et humains. Cela consiste donc à considérer le e-Portfolio non seulement comme un entrepôt statique de données mais plutôt comme un processus dynamique, ouvert et partagé [Mansour *et al.* , 2005].

Pour la e-Qualification et durant la simulation l'e-Portfolio est continuellement mis à jour avec les action et résultats des apprenants. Les connaissances acquises de façon informelle sont ajoutées durant les échanges collaboratifs. Cette évolution perpétuelle des compétences objectifs et besoins de chaque apprenant ainsi que ses relations avec les autres qui apparaissent et disparaissent justifient le besoin de composer de façon dynamique les e-Portoflio.

Dans le cadre des recherches autour des systèmes distribués et plus particulièrement celui de la grille sémantique [De Roure *et al.* , 2004], l'enjeu majeur reste la capacité des applications à offrir, intégrer et coordonner à la demande les services disponibles sur le réseau [Goble & De Roure, 2001]. De ce fait la composition des services devient la piste la plus prometteuse et qui suscite le plus d'intérêts [LP06]. Le travail de [LP06] propose une solution à base de composants adaptables permettant de composer de façon dynamique des composants de différentes applications. Dans [LP06] un composant est défini comme une abstraction d'une entité logicielle existante (application, service, programme...). Le e-Portoflio comme service de grille est dans notre vision un cas concret d'un composant. Le modèle de composants adaptables proposé par [LP06] étend le modèle de composants nommé « Ugatze » proposé par [Sey04] et repose sur le principe des points d'adaptation qui sont associés à chaque sous-service du composant permettant son adaptation.

A chaque composant il faut identifier les points d'adaptation, cette étape revient à distinguer quels sous-services sont offerts par le composant. A chaque point d'adaptation est associée une méta description. Cette description est composée de deux parties. La description fonctionnelle indique la fonction du sous-service. Elle décrit le service proprement dit. La deuxième partie non fonctionnelle permet de spécifier certaines propriétés (qualité de service, sécurité, performance...).

Durant la phase d'échanges informels entre apprenants, certains tentent d'enrichir leurs connaissances quand leurs objectifs ne sont pas atteints en interagissant avec d'autres qui ont les compétences requises et qu'ils partagent au sein des communautés de compétences (CoC). Ce scénario est un cas typique d'adaptation de composants. Nous concevons le e-Portfolio comme un composant adaptable réagissant aux besoins de son propriétaire mais aussi aux besoins des autres apprenants particpant ainsi à l'activité collaborative de la communauté

d'apprenants. Nous concevons le e-Portfolio comme un composant adaptable selon le modèle proposé dans [LP06]. Notre intérêt porte plus la phase de spécification plus particulièrement sur les points d'adaptation qui serviront pour l'utilisation du modèle AGRS dans la composition des e-Portfolio.

8.1.2 Adaptable e-Portfolio

Pour la e-Qualification et durant la simulation l'e-Portfolio est continuellement mis à jour avec les action et résultats des apprenants. Les connaissances acquises de façon informelle sont ajoutées durant les échanges collaboratifs. Cette évolution perpétuelle des compétences objectifs et besoins de chaque apprenant ainsi que ses relations avec les autres qui apparaissent et disparaissent justifient le besoin de composer de façon dynamique les e-Portoflio.

Dans le cadre des recherches autour des systèmes distribués et plus particulièrement celui de la grille sémantique, l'enjeu majeur reste la capacité des applications à offrir, intégrer et coordonner à la demande les services disponibles sur le réseau. De ce fait la composition des services devient la piste la plus prometteuse et qui suscite le plus d'intérêts [LP06]. Le travail de [LP06] propose une solution à base de composants adaptables permettant de composer de façon dynamique des composants de différentes applications. Dans [LP06] un composant est défini comme une abstraction d'une entité logicielle existante (application, service, programme...). Le e-Portoflio comme service de grille est dans notre vision un cas concret d'un composant. Le modèle de composants adaptables proposé par [LP06] étend le modèle de composants nommé « Ugatze » proposé par [Sey04] et repose sur le principe des points d'adaptation qui sont associés à chaque sous-service du composant permettant son adaptation.

A chaque composant il faut identifier les points d'adaptation, cette étape revient à distinguer quels sous-services sont offerts par le composant. A chaque point d'adaptation est associée une méta description. Cette description est composée de deux parties. La description fonctionnelle indique la fonction du sous-service. Elle décrit le service proprement dit. La deuxième partie non fonctionnelle permet de spécifier certaines propriétés (qualité de service, sécurité, performance...).

Durant la phase d'échanges informels entre apprenants, certains tentent d'enrichir leurs connaissances quand leurs objectifs ne sont pas atteints en interagissant avec d'autres qui ont les compétences requises et qu'ils partagent au sein des communautés de compétences (CoC). Ce scénario est un cas typique d'adaptation de composants. Nous concevons le e-Portfolio comme un composant adaptable réagissant aux besoins de son propriétaire mais aussi aux besoins des autres apprenants particpant ainsi à l'activité collaborative de la communauté d'apprenants. Nous concevons le e-Portfolio comme un composant adaptable selon le modèle proposé dans [LP06]. Notre intérêt porte plus la phase de spécification plus particulièrement sur les points d'adaptation qui serviront pour l'utilisation du modèle AGRS dans la composition des e-Portfolio.

Conformément au modèle à base de composants chaque e-Portfolio a un corps et des sous-services. La figure ci-dessous donne un aperçu d'un exemple simplifié du point d'adaptation crée pour aider l'apprenant la procédure d'atterrissage.

Nous reposons notre solution d'adaptation dynamique d'e-Portfolio et de composants en général sur l'association entre le concept de point d'adaptation exprimé par le modèle de composants proposé par Lacouture dans [LP06] et le concept de rôle comme spécifié dans notre modèle AGRS.

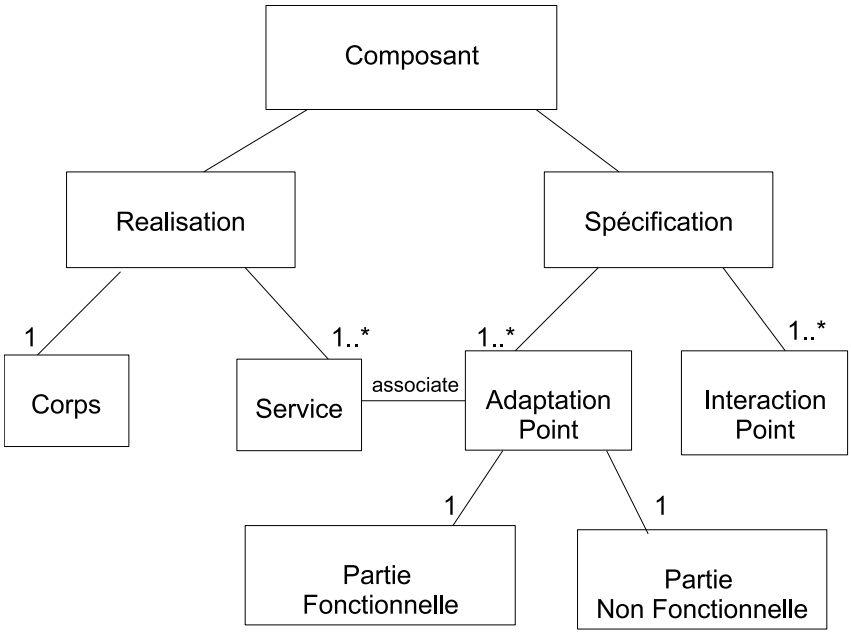

Fig. 8.1 – Modèle d'adaptation de composants basé sur les points d'adaptation.

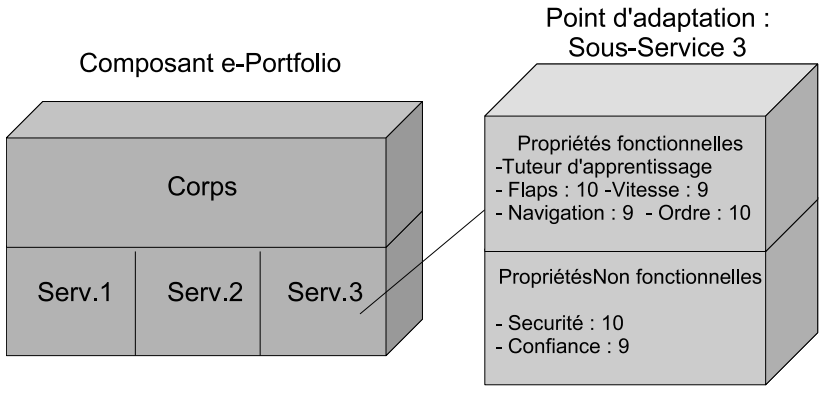

Fig. 8.2 – Exemple d'e-Portfolio d'un apprenant avec Zoom sur un point d'adaptation proposant une aide sur la procédure d'atterrissage.

8.2 Les agents pour des composants de grille auto-organisés

8.2.1 Introduction

L'intérêt de l'utilisation des agents avec les composants est de plus en plus reconnu par les chercheurs des deux communautés [Bri05]. D'une part, les concepts des composants sont utiles pour la réalisation et le déploiement des systèmes multi-agents modulaires et réutilisables. D'autre part, l'autonomie, l'adaptation et la coordination sont des qualités très enrichissantes pour les composants et leurs compositions.

Généralement les composants sont indexés dans des catalogues ce qui facilite leur réutilisation. En contre partie, cette technique nécessite un effort considérable pour le déploiement des composants. La solution que nous proposons dans [Lacouture & Mansour GESM07] consiste à rendre les composants plus adaptables en se basant sur l'approche agent afin de rendre le processus de recherche, d'adaptation et de composition le plus transparent possible. Nous proposons donc d'associer à chaque composant un agent capable de réagir à deux types d'évènements majeurs : les évènements internes qui concernent le composant qu'il gère et les évènement externes reçus des composants externes.

8.2.2 Les agents du modèle AGRS pour supporter les composants de grille

Dans notre solution nous proposons d'associer un agent à chaque composant qui aura pour responsabilité de superviser continuellement l'évolution de son composant. Comme on l'a vu chaque composant représente un e-Portfolio et peut en même temps soit offrir ou avoir besoin de sous services. Les besoins des composants sont spécifiés en terme de composition. Chaque nouveau sous service a besoin d'être associé à un sous service similaire d'un autre composant. La responsabilité des agents est de trouver les agents existants et qui possèdent des descriptions qui ressemblent le mieux au nouveau sous service recherché par son e-Portfolio. En effet, comme on l'a déjà mentionné, les agents sont en écoute continue de leurs composants (e-Portfolio dans notre cas). Dès que l'agent détecte un nouveau événement il le traite. S'il s'agit d'un nouveau sous service créé par son e-Portfolio, l'agent met à jour, au niveau du réseau à travers un Advertisement (au sens JXTA), la description de l'e-Portfolio qu'il représente et lance une recherche d'un rôle qui correspond le mieux aux propriétés fonctionnelles du sous-service créé. Les propriétés fonctionnelles correspondent aux services au sens AGRS. Afin de raffiner sa recherche et joindre le plus rapidement possible le meilleur groupe correspondant à ses besoins, en plus de la description des services recherchés (les propriétés fonctionnelle au niveau composant), l'agent enrichi sa requête avec d'autres informations décrivant par exemple le type de rôle recherché, l'ensemble des ontologies connus, le propriétaire de l'e-Portoflio...La gestion des groupes d'agents représentants les e-Portfolio se fait selon les spécification et la mise en oeuvre du modèle AGRS comme présenté dans ce travail.

Grouper les agents en groupes et les catégorisant en rôles revient à faire des blocs au niveau composants. L'utilisation du modèle AGRS et des mécanismes que nous avons mis en place pour faciliter la recherche et la publication de services au niveau des agents, s'utilise d'une façon quasi transparente pour mettre en relation des composants (des e-Portfolio) différents et ce d'une façon dynamique et évolutive. L'évolution des connaissances d'un apprenant passe d'un simple enregistrement au niveau de l'e-Portfolio à une classification dans des groupes et plus important en qualifiant selon les rôles chaque nouvelle évolution au niveau de l'e-Portfolio.

Cette étape n'est que la phase finale du processus de qualification. La validation des échanges et l'enrichissement de l'e-Portfolio avec les connaissances acquises de façon informelle passe par une processus antérieur décrit dans [Gouarderes ...] [Gouardères canada] et réalisé grâce au modèle ARS pour (Actors, Roles & Signature). Le modèle ARS constitue le coeur du processus de e-Qualification et notre travail sur le modèle AGRS a été précédé par une contribution dans la construction du modèle ARS ce qui nous a permis de bien saisir les problèmes au niveau de la gestion décentralisée de groupes ouverts et dynamiques.

Chapitre 9

Conclusion

9.1 Contributions

L'origine de cette thèse est un intérêt pour les systèmes ouverts, dynamiques et largement distribués. Notre problématique s'est présentée lors de l'étude d'un cas particulier du domaine du e-Learning dans le cadre du projet ELeGI [Dimitrakos & Ritrovato, 2004] . Cela nous a permis d'appréhender concrètement les différents niveaux de difficulté de notre problématique. Le point principal de notre contribution est bien sûr le modèle Agent-Groupe-Role-Service, mais d'une façon générale nous avons voulu défendre l'idée qu'un traitement au niveau social des problèmes d'ouverture, de dynamique et de gestion distribuée des systèmes multi-agents, est possible et même indispensable pour aboutir à des systèmes opérationnels et exploitables dans des environnements ouverts et très dynamique comme Internet ou la grille.

Après avoir posé la problématique par rapport au problème de e-Qualification, point de départ de notre thèse, nous avons fait l'état de l'art des différents modèles agents existants. Nous avons expliqué pourquoi les modèles centrés agents ne pouvaient pas répondre aux besoins ds systèmes ouverts et ne peuvent donc pas être un cadre de notre travail. Notre intérêt s'est donc porté sur les modèles organisationnels. Afin de mieux cadrer notre étude, nous avons étudié le point de vue des sociologues vis-à-vis des organisations ouvertes et dynamiques. Cette étude nous a permis de dégager un point essentiel : les membres d'une organisation ouverte, étant dans l'impossibilité d'avoir une vue globale du système, doivent avoir les mécanismes facilitant la découverte de ce qu'englobe chaque organisation comme compétences sociales. Cela se traduit au niveau des modèles agents par une structure organisationnelle qui permet de refléter le contenu de l'organisation qu'elle structure. Il était donc nécessaire d'étudier les contributions de la communauté SMA pour la résolution de ces problèmes.

Nous avons vu au chapitre 3, qu'aucun des modèles existants ne pouvait réellement servir de cadre à notre question. Il manque à AGR [5] [10] l'expression explicite des compétences et services offerts par les agents. Les rôles ne sont que des étiquettes ne laissons qu'un vague sous entendu sur les services que doit offrir chaque agent joueur du rôle. GAIA [Wooldridge&al 2000] propose une méthodologie complète pour la conception des SMA, mais n'est pas destinée à un déploiement dans des systèmes ouverts. Les types d'agents sont fixés à l'avance et ne peuvent plus changer, les relations entre les agents sont statiques, enfin dans GAIA les droits et permissions définis pour les ressources sont supposés être connus à l'avance chose irréalisable dans un environnement ouvert. Colman&Han [2] expriment l'organisation en terme de relations entre les rôles qui la composent. L'introduction du concept de capacité comme point

de rattache entre le rôle et les agents qui le jouent est un concept très intéressant dans le cadre des systèmes ouverts. Cependant, les auteurs négligent l'aspect utilisation du rôle et ne cadrent pas les interactions entre agents au niveau organisationnel. Dans RoMas[9] le rôle est défini dès la phase d'analyse et les relations entre les rôles ne peuvent être modifiés en cours d'exécution, cela fige l'organisation et rend le modèle inadéquat pour des systèmes ouverts et dynamiques. Odell [4] introduit le concept de classificateurs en distinguant classificateur physique permettant de catégoriser les agents selon leurs types, capacités, attributs ... et classificateur de rôles permettant de spécifier pour un agent donné, les types de rôles qu'il peut jouer. Un autre aspect intéressant de l'approche proposée consiste en la séparation concrète de la relation Agent-Rôle en introduisant le concept de AgentRoleAssignement. Cette entité n'est par contre qu'une simple entité vide. D'autre part, les rôles sont fixes et ne peuvent pas changer, ce qui prive le système d'un degré de flexibilité nécessaire pour le déploiement dans un cadre comme celui de notre travail. Yamam [] propose l'utilisation de la notion de compétences rattachés au rôles mais délègue aux agent la tâche de construire leurs propres réseaux d'accointance. De même dans RIO [] c'est aux agent de créer et gérer leurs propres réseaux d'accointances. En plus RIO ne considère les agents que comme une encapsulation de compétences, une contrainte forte qui restreint considérablement le déploiement du modèle dans un environnement ouvert où on ne peut jamais contrôler l'état interne des agents. MOCA [] tout comme RIO, considère les agents comme des coquilles vides dans lesquels des compétences sont ajoutés. Cependant dans MOCA, l'autonomie des agents est réduite et déplacée au niveau des rôles. Les rôles sont insérés dans les agents ce qui conduit à un système où architecture interne des agents et structures organisationnelles sont mélangés, ce qui constitue un handicap majeur pour l'ouverture du système. Enfin, Cabri&Al proposent, via les modèles RoleSystem et RoleX, un modèle assez complet et adapté à la problématique des systèmes ouverts et dynamiques, mais nous avons vu que le modèle présente plusieurs faiblesses, notamment l'absence de séparation (fournisseur/demandeur) qui nécessite une décomposition d'un rôle en deux. D'autre part, la solution proposée dans RoleX, pour combler le défaut de centralisation de RoleSystem, consistant à fusionner le concept de rôle physiquement avec celui d'agent dans RoleX, n'est pas réalisable dans la pratique.

Nous avons présenté dans le chapitre **??** notre modèle nommé AGRS [Mansour & Ferber, 2007b], qui hérite du modèle AGR et l'étend avec des nouveaux concepts comme celui de services, inspiré des modèles existants tel que YAMAM, RIO ou MOCA, l'utilisation des rôles (Utilsiateur/Joueur) concept de base des systèmes P2P, ou encore la notion AgentInRole qui ressemble au concept RoleAssignement de Odell [] que nous enrichissons pour mieux exprimer l'interaction entre les agents d'une part et entre les agents et le rôle d'autre part. De plus, des raisons d'opérationnalité nous ont poussé à adopter une architecture hybride (agent/P2P) pour l'implémentation de notre modèle.

Dans Moca [Thèse], une des perspectives de la thèse défendue, consiste à proposer une gestion décentralisée des groupes du modèle AGR. Nous avons proposé dans le Chapitre 5 un modèle alternatif de gestion décentralisée des groupes assurant à la fois la fiabilité et la flexibilité du système. Le modèle proposé repose sur les principes du « knowledge pump », une technique de gestion des mémoires partagées. Les principes de la solution sont simples, chaque groupe est vu comme une mémoire partagée dupliquée sur différentes plateformes, les évènements jugés importants sont propagés sur les différentes copies pour mettre à jour le groupe.

Le modèle AGRS a été opérationnalisé par la plateforme Madkit que nous présentons dans le chapitre 7. L'implémentation du modèle ainsi que la mise en oeuvre du mécanisme de gestion décentralisée sont construits au dessus d'une architecture hybride Madkit/JXTA.

De l'implémentation de notre approche nous retirons deux enseignements. D'une part, nous avons pu vérifier, par la mise en ouvre du modèle AGRS et son implémentation ainsi qu'à travers les divers exemples applicatifs que nous avons construits, que l'utilisation des concepts de services, des rôles et l'adoption des mécanismes P2P de découverte et publication de services, combinés à une gestion décentralisée et distribuée des groupes permettent de résoudre les problèmes d'ouverture et de dynamique que nous avons identifiés. Mais d'autre part, l'expérience acquise ici nous a permis d'aborder la problématique de la composition de plusieurs technologies. Si l'utilisation des systèmes P2P avec les SMA fait partie de l'implémentation du modèle proposé, nous avons montré que d'autres technologies ; tels que les composants ou les services de grille, peuvent tirer profil des souplesses de la solution proposée [Lacouture & Mansour 2007].

9.2 Perspectives

Différents prolongements peuvent être envisagés à ce travail. La première piste serai de consolider les principes du modèle Agent-Groupe-Rôle-Service. Il serait intéressant d'étudier comment utiliser la solution proposée dans différents contextes où les concepts d'ouverture et de dynamique sont des problèmes épineux. Il s'agit, d'une part de confronter le modèle AGRS à des problèmes variés et de voir le comportement de l'implémentation proposée face à des contraintes distinctes. D'autre part, cela permettra sans doute de raffiner le modèle et d'enrichir son implémentation avec des nouveaux concepts. Il nous semble donc intéressant de pousser beaucoup plus loin l'utilisation de notre travail dans un cadre plus large d'analyse et de résolution de problèmes.

Il est clair cependant que le présent travail n'est pas complet et n'est en rien définitif et plusieurs modifications et extensions restent à faire. En particulier, la synchronisation entre différentes copies d'un même groupe peut poser des problèmes de conflits et d'erreurs qui sont loin d'être résolus dans l'implémentation que nous proposons. Il faudrait aussi perfectionner le travail sur l'ontologie des services. En effet, dans l'état actuel de son implémentation cette partie ne permet pas une réutilisation accrue des services relatifs à une application donnée. Il serai intéressant de voir comment concevoir et formaliser les ontologies de services pour permettre d'une part leur extension facile et rapide et d'autre part leur publication sur le réseau. De même, l'étude des aspects de sécurité et sûreté au niveau de la gestion des groupes, des interactions entre agents et échanges entre plateformes est une perspective intéressante, qu'il faudrai explorer et mieux étudier. Cela est d'autant plus important, que le travail que nous proposons est destiné à être utilisé dans des environnements tel que Internet où la réussite de toute application passe indéniablement par sa sécurité. Les accès restreints au niveau de l'entrée aux groupes que nous avons mis en place ne sont qu'une simple barrière loin d'être la meilleure parade contre les tentatives frauduleuses qui peuvent tenter d'attaquer ou d'infiltrer les applications privés déployés au dessus de la nouvelle version de Madkit.

Loin de l'aspect implémentatoire, la poursuite du travail sur le modèle AGRS nous paraît indispensable : il s'agit d'essayer de proposer toute une méthodologie qui se base sur les concepts du modèle AGRS. A l'instar de GAIA, il serai intéressant de proposer une méthodologie basée sur le paradigme rôle et enrichie du concept de service et celui d'AgentInRole. Les services doivent se rattacher ici aux rôles plus qu'aux agents. Il est aussi important de maintenir la distinction entre la phase d'analyse et celle de conception qui existe dans GAIA. Toutefois, il est évident que toute tentative de figer n'importe quel partie du système au niveau de l'analyse ou de la conception serai en contradiction avec le modèle AGRS. Toute la

difficulté consiste à proposer une méthodologie qui permet de guider l'utilisateur dans la mise en place d'une application basée sur AGRS, tout en garantissant l'ouverture et la dynamique les deux axes de bases de notre modèle.

Toujours dans l'aspect conceptuel, la prise en compte de la mobilité des agents par le modèle AGRS doit être traitée et étudiée en profondeur. Les agents mobiles constituent un axe de travail prometteur qui tente de faciliter le déploiement d'applications ouvertes et qui offre une flexibilité nécessaire dans certains types d'applications dynamiques. Les travaux dans ce domaine peuvent prendre comme point de départ nos travaux autour du modèle organisationnel MAGR [Mansour & Ferber, 2007a] qui se base déjà sur le concept de service. En effet, les agents mobiles migrent dans MAGR d'un noeud à l'autre à la recherche de services non disponibles localement. L'idée de rechercher des services et de se déplacer pour les obtenir puis revenir peut être une solution qui répond à certaines contraintes sécuritaires. En effet, certains groupes, pour des raisons de sécurité, peuvent ne pas être distribuables et donc les concepts de AGRS qui consistent à importer un groupe d'une plateforme distante pour le dupliquer localement ne peuvent plus être réalisables. La prise en compte des concepts du modèle MAGR par une extension de AGRS peut constituer une alternative prometteuse dans le traitement de la mobilité des agents dans des environnements ouverts et dynamiques.

Annexe A

Annexe B

Références bibliographiques

[Allison *et al.* , 2003]Allison, C., Cerri, S, A., Gaeta, M, Ritrovato, P., & Salerno, S. 2003. Human Learning as a Global Challenge : European Learning Grid Infrastructure. *Global Peace Through the Global University System.*, 465–488.

[Amiguet & Baez, 2004]Amiguet, M.and Nagy, A., & Baez, J. 2004. Towards an Aspect-Oriented Approach of Multi-Agent Programming MOCA'04. *Page 18 of : 3rd Workshop on Modelling of Objects, Components, and Agents* ,.

[Amiguet, 2003]Amiguet, M. 2003. *MOCA : Un modèle componentiel dynamique pour les systèmes multi-agents organisationnels.* Thèse de Doctorat, Faculté des sciences de Neuchatel.

[Barret, 2001]Barret, H, C. 2001. ICT Support for Electronic Portfolios and Alternative Assessment. *WCCE.*, 569–578.

[Barret, 2004]Barret, H, C. 2004. Differentiating Electronic Portfolios and Online Assessment Management Systems. *In : AERA and AAHE's Assessment Conference, SITE 2004.*

[Boella *et al.* , 2006]Boella, G., Damiano, R., Hulstijn, J., & Van Der Torre, L. 2006. The Roles of Roles in Agent Communication Languages. *Pages 381–384 of : International Conference on Intelligent Agent Technology.*

[Bowen & al, 2004]Bowen, F, & al. 2004. La médiation par les pairs à l'école primaire : conditions de réussite et perspectives de recherche. *The International Journal Of Victimology*, **1**(4), 77–96.

[Cabri *et al.* , 2003a]Cabri, G., Leonardi, L., & Zambonelli, F. 2003a. BRAIN : A Framework for Flexible Role-Based Interactions in Multiagent Systems Lecture notes in computer science. *Software Engineering for Large-Scale MultiAgent Systems*, 145–161.

[Cabri *et al.* , 2003b]Cabri, G., Leonardi, L., & Zambonelli, F. 2003b. Implementing Role-based Interactions for Internet Agents. *In : Proceedings of the 2003 Symposium on Applications and the Internet.*

[Cabri *et al.* , 2004a]Cabri, G., Ferrari, L., & Zambonelli, F. 2004a. Role-Based Approaches for Engineering Interactions in Large-Scale Multi-agent Systems. *Lecture Notes in Computer science*, 243–263.

[Cabri *et al.* , 2004b]Cabri, G., Ferrari, L., Leonardi, L., Klush, M., Ossowski, S., Kashyap, V., & Unland, R. 2004b. The RoleX environment for multi-agent cooperation. *In : International workshop on cooperative information agents No8.*

[Cabri *et al.* , 2005]Cabri, G., Ferrari, L., & Leonardi, L. 2005. The Role of Roles in Designing Effective Agent Organizations. *Science of Computer Programming*, **54**(1), 73–98.

[Chaib-draa & Levesque, 1996]Chaib-draa, B, & Levesque, P. 1996. Hierarchical Model and Communication by Signs, Signals and Symbols in Multiagent Environments. *Journal of Experimental & Theoretical Artificial Intelligence*, **8**(1), 7–20.

[Chen & Yeager, 2003]Chen, R, Y, & Yeager, B. 2003. Java Mobile Agents on Project JXTA Peer-to-Peer Platform. *In : The 36th Hawaii International Conference on System Sciences (HICSS'03).*

[Colman & Han, 2003]Colman, A., & Han, J. 2003. Organizational roles and players. *AAAI Fall Symposium, Roles, an Interdisciplinary Perspective*, 55–62.

[Dastani & al, 2004]Dastani, M., & al. 2004 (July). Enacting and Deacting Roles in Agent Programming. *In : 5th Internation Workshop, Agent-Oriented Software Engineering (AOSE).*

[De Roure *et al.* , 2004]De Roure, D, Jennings, N, R, & Shadbolt, N, R. 2004. The Semantic Grid : Past, Present, and Future. *Proceedings of the IEEE*, **93**(3), 669–681.

[Dillenbourg, 1999]Dillenbourg, P. 1999. *Collaborative-learning : Cognitive and Computational Approaches.* Pergamon.

[Dimitrakos & Ritrovato, 2004]Dimitrakos, T., & Ritrovato, P. 2004 (27-28 Avril). Towards a European Learning Grid Infrastructure : Progressing with a European Learning Grid. *In : 4th International LeGE-WG Workshop.*

[Ferber, 1999]Ferber, J. 1999. *Les systèmes multi-agents. Vers une intelligence collective.* Inter-Editions.

[Ferber & Gutknecht, 1998]Ferber, J, & Gutknecht, O. 1998. A meta-model for the analysis and design of organization in multi-agent systems. *Pages 128–135 of : Proceeding of the 3rd international conference on multi-agent systems.*

[Ferber *et al.* , 2003]Ferber, J., Gutknecht, O., & Michel, F. 2003. From Agents to Organizations, an Organizational View of Multi-Agent Systems. *Pages 214–230 of : Agent-Oriented Software Engineering (AOSE) IV.* Agent-Oriented Software Engineering (AOSE) IV.

[Flatcher, 2002]Flatcher, M. A. 2002. *In : Reflections on reciprocity in professional development : Learning partners as professional learning teams.*

[Foisel, 1998]Foisel, R. 1998. *Modèle de réorganisation de systèmes multi-agents : une approche descriptive et opérationnelle.* Thèse de Doctorat, Université Henri Poincaré - Nancy 1.

[Foster & Kesselman, 1999]Foster, I, & Kesselman, C. 1999. *The Grid : Blueprint for a New Computing Infrastructure.* Morgan Kaufmann.

[Foster *et al.* , 2004]Foster, I., Jennings, N, R., & Kesselman, C. 2004. Brain meets brawn : Why Grid and agents need each other. *In : 3rd Int.Conf. on Autonomous Agents and Multi-Agent Systems.*

[Gaeta & Ritrovato, 2004]Gaeta, M., & Ritrovato, P. 2004. The European Learning Grid Infrastructure Integrated Project. *ERCIM News*, **59**.

[Ganutella, n.d.]Ganutella. *The Gnutella file-sharing system.*

[Goble & De Roure, 2001]Goble, C., & De Roure, D. 2001. The Semantic Web and Grid Computing. *Real World Semantic Web Applications*, **92**.

[Gokhale, 1995]Gokhale, Anuradha A. 1995. Collaborative Learning Enhances Critical Thinking. *Journal of Technology Education*, **7**(1).

[Gouardères *et al.* , 2007]Gouardères, E., Dourisboure, Y., Gouardères, G., & Mansour, S. 2007. The ARS Model : Virtual Learning Communities with P2PAgents on the Grid. *In : First IEEE International Workshop on Cooperative Distributed Systems (CDS), held in conjunction with the IEEE International Conference on Distributed Computing Systems (ICDCS 2007).* Toronto, Canada.

[Gouardères *et al.* , 2005]Gouardères, G., Mansour, S, Yatchou, R., & Nkambou, R. 2005. The Grid-e-Card : An Architecture for Collective Intelligence Sharing on the Grid. *Applied Artificial Intelligence Journal, Special Issue on Learning Grid Services*, **19**(9-10), 811–824.

[Gouardères *et al.* , 2006]Gouardères, G, Mansour, S, & Gouardères, S. 2006 (OCtobre). Qualification of individual & collective competence on the grid. *In : Technologies de l'Information et de la Communication dans l'Enseignement Supérieur et l'Entreprise, TICE 2006.*

[Gradecki, 2002]Gradecki, J, D. 2002. *Mastering JXTA : Building Java Peer-to-Peer Applications.* 528.

[Gutknecht, 2000]Gutknecht, O. 2000. Madkit : A generic multi-agent platform. *Pages 78–79 of : AGENTS'00 : 4th International Conference on Autonomous Agents.*

[Gutknecht, 2001]Gutknecht, O. 2001 (14 séptembre). *Proposition d'un modèle organisationnel générique de systèmes multi-agents Examen de ses conséquences formelles,implémentatoires et méthodologiques.* Thèse de Doctorat, Université Montpellier II.

[Hameurlain, 2004]Hameurlain, N. 2004. Un modèle de spécification et d'implémentation de composants-rôles pour les systèmes multi-agents. *In : JMAC'04.*

[JXTA, n.d.]JXTA. *Sun Microsystems : Project JXTA.*

[KECHIDI, 2005]KECHIDI, Med. 2005. La théorie de la structuration : une analyse des formes et des dynamiques organisationnelles. *Revue Relations Industrielles /Industrial Relations*, **60**(2).

[Kreijns & Kirschner, 2001]Kreijns, K, & Kirschner, K. 2001. The social accordances of computer-supported collaborative learning environments. *Pages 12–17 of : In 31th ASEE/IEEE Frontiers in Education Conference.*

[Lacouture & Mansour, 2007]Lacouture, J, & Mansour, S. 2007. e-Portfolio, an auto-adaptable grid service with components using Peer-to-Peer agents. *In : GSEM 07.*

[Lethbridge, 2004]Lethbridge, T. 2004. *Object-oriented Software Engineering : Practical Software Development Using Uml And Java.*

[Livingstone, 2006]Livingstone, D.W. 2006. Informal Learning : Conceptual Distinctions and Preliminary Findings. *Learning in Places the informal education reader.*

[MadKit, 1998]MadKit. 1998. *MadKit, a **M**ulti–**A**gent **D**evelopment **Kit**, site officiel : http ://www.madkit.org.*

[Mansour & Ferber, 2007a]Mansour, S., & Ferber, J. 2007a (July). MAGR : Integrating mobility of agents with organizations. *Page 8 of : International Conference Intelligent Systems and Agents IADIS(2007).*

[Mansour & Ferber, 2007b]Mansour, S., & Ferber, J. 2007b (Octobre). Un modèle organisationnel pour les systèmes multi-agents ouverts. *Page 10 of : Journées Francophones des Systèmes multi-agents JFSMA'07.*

[Mansour *et al.* , 2005]Mansour, S., Gouardères, G., & Conté, E. 2005 (May). User modeling for services personalization in Virtual Learning Communities. *Page 10 of : AI-CBT'05 : Workshop on Formal AI Techniques in Computer-Based Training In conjunction with the Eighteenth Canadian Conference on Artificial Intelligence (AI 2005).*

[Marcio & Yu, 2002]Marcio, M, & Yu, E. 2002. Requirements Engineering for Large-Scale Multi-Agent Systems. *Pages 39–56 of :* Springer (ed), *Software Engineering for Large-Scale Multi-Agent Systems, Research Issues and Practical Applications [the book is a result of SELMAS 2002].*

[Mehan, 1982]Mehan, H. 1982. Le constructivisme social en psychologie et en sociologie. *Sociologie et société*, **14**(2), 77–96.

[Minko & Gouardères, 2004]Minko, A, & Gouardères, G. 2004. Towards Qualitative Accreditation with Cognitive Agents. *Intelligent Tutoring Systems : 7th International Conference*, **3220**(August 30 - September 3), 118–127.

[Moore, 2006]Moore, M. 2006. Theory of transactional distance. *Theoretical principles of distance education*, 22–38.

[Napster, n.d.]Napster. *The Napster file-sharing system.*

[Nejdl *et al.* , 2002]Nejdl, W., Wolf, S., & Qu, C. 2002. EDUTELLA : A P2P Networking Infrastructure Based on RDF. *In : International WWW Conference (11).*

[Nejdl *et al.* , 2004]Nejdl, W., Wolpers, M., Wolf, B., Löser, A., & al. 2004 (Spetembre). Super-Peer-Based Routing Strategies for RDF-Based P2P Systems. In Proceedings of the 2nd International Workshop On Databases. *In : The 2nd International Workshop On Databases, Information Systems and Peer-to-Peer Computing.*

[Odell *et al.* , 2003]Odell, J., Parunak, V, D., & Fleischer, M. 2003. The Role of Roles in Designing Effective Agent Organizations. *Software Engineering for Large-Scale MultiAgent Systems*, 55–62.

[Odell *et al.* , 2004a]Odell, J., Marian, H, N., & Levy, R. 2004a (July). A Metamodel for Agents, Roles, and Groups. *Pages 78–92 of : Agent-Oriented Software Engineering (AOSE04).*

[Odell *et al.* , 2004b]Odell, J., Parunak, V, D., & Fleischer, M. 2004b. Temporal Aspects of Dynamic Role Assignment. *Agent-Oriented Software Engineering (AOSE) IV*, **2953**.

[Plane, 2003]Plane, JM. 2003. *Théorie des organisations.* 123.

[Pourebrahimi *et al.* , 2005]Pourebrahimi, B., Bertels, K, L M., & Vassiliadis, S. 2005 (Novembre). A Survey of Peer-to-Peer Networks. *In : The 16th Annual Workshop on Circuits, Systems and Signal Processing.*

[Rasmussen, 1986]Rasmussen, J. 1986. *Information processing and human-machine interaction.* Elsevier Science Ltd.

[Ravitz, 1997]Ravitz, J. 1997. An ISD Model for Building Online Communities : Furthering the Dialogue. *In : Annual Conference of the Association for Educational Communications and Technology.*

[Razmerita *et al.* , 2005a]Razmerita, L., Gouardères, G., & Conté, E. 2005a. Ontology-based user modeling & e-Portfolio grid learning services. *Special Issue On LEarning Grid Services of the Applied Artificial Intelligence Journal.*, **19**(9-10).

[Razmerita *et al.* , 2005b]Razmerita, L., Gouardères, G., Conté, E., & Mansour, S. 2005b. Ontology based user modeling for personalization of grid learning services. *First International Elegi Conference on Advanced Technology for enhanced learning.*

[Rheingold, 1993]Rheingold. 1993. *The Virtual Community : Homesteading at the Electronic Frontier.* Addison-Wesley, A William Patrick Book.

[Rogers & Freiberg, 1994]Rogers, C.R, & Freiberg, H.J. 1994. *Freedom to Learn.* Prentice Hall.

[Rojot, 2000]Rojot, J. 2000. *La théorie de la structuration chez Anthony Giddens.*

[Saval, 2003]Saval, M. 2003. *Une architecture d'agents pour la simulation : Le modèle YAMAM et sa plate-forme Phoenix.* Thèse de Doctorat, INSA Rouen.

[Schollmeier, 2002]Schollmeier, A. 2002. A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications. *In : Proceedings of the First International Conference on Peer-to-Peer Computing.*

[Schollmeier, 1998]Schollmeier, R. 1998 (August). A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. *In : International Conference on Peer-to-Peer Computing (P2P2001).*

[Soloway, 1996]Soloway, E. 1996. Learning Theory in Practice : case studies of learner centered design. *Pages 189 – 196 of : Proceedings of the SIGCHI conference on Human factors in computing systems : common ground*. Roger Crawford.

[Steels, 1993]Steels, L. 1993. *Corporate knowledge management*.

[Sycara, 1998]Sycara, K, P. 1998. Multi-agents Systems. *AI magazine Volume*, **19**(2).

[Vanides & Morgret, n.d.]Vanides, J., & Morgret, K. *STEP ePortfolio Workspace : Supporting Pre-Service Teachers with Electronic Portfolio Creation, Reflection and Online Collaboration .* Tech. rept.

[Wai-Sing Loo, 2007]Wai-Sing Loo, A. 2007. *Peer-to-Peer Computing (Building Supercomputers with Web Technologies)*.

[Watson, 1930]Watson, J.B. 1930. *Behaviorism*. University of Chicago Press.

[Wooldridge *et al.* , 2000]Wooldridge, M., Jennings, N, R., & Kinny, D. 2000. The Gaia Methodology for Agent-Oriented Analysis and Design. *Journal of Autonomous Agents and Multi-Agent Systems (jaamas)*, **3**, 285–312.

[Xinjun & Yu, 2004]Xinjun, M., & Yu, E. 2004 (July). Agent Oriented Software Engineering 5th Internation Workshop (AOSE04).

[Yan *et al.* , 2003]Yan, Q., Shan, L, J., & Mao, X, J. 2003. RoMAS : A Role-Based Modeling Method for Multi-Agent System. *In : International Conference on Active Media Technology*.

[Yoo, 1999]Yoo, MJ. 1999 (Octobre). *Une approche componentielle pour la modélisation d'agents coopératifs et leur validation*. Thèse de Doctorat, Laboratoire d'Informatique de Paris 6 (LIP6).

Liste des tables

Liste des figures

Index

Une architecture Multi-agent pour un apprentissage autonome guidé par les émotions

Résumé

Dans cette thèse nous présentons une architecture multi-agent permettant à un robot mobile autonome d'apprendre de manière non supervisée. L'apprentissage autonome est réalisé grâce à l'utilisation d'émotions qui représentent les besoins primaires de l'apprenant. Le processus d'apprentissage que nous proposons est inspiré de l'organisation en colonnes corticales du cerveau chez les espèces vivantes. L'architecture multi-agent de type organisationnelle est utilisée pour décrire les interactions entre les entités impliquées dans le processus d'apprentissage.

Mots-clés : Systèmes Multi-Agent, Apprentissage, Agent émotionnel, Robotique.

A Multi-agent architecture for an emotion driven learning

Abstract

In this thesis, we present a multi-agent architecture giving a robot the ability to learn in an unsupervised way. An autonomous learning is achieved by using emotions which represent basic needs for the learning entity. The learning process we propose is inspired by the organization in cortical columns and areas of a living being brain. The organizational multi-agent architecture is used to describe the interaction among entities involved in the learning process

Keywords : Multi-Agent Systems, Machine Learning, Emotional agent, Robotics.