Agent Groupe Rôle et Service : Un modèle organisationnel pour les systèmes multi-agents ouverts

S.Mansour[1] mansour@lirmm.fr
J. Ferber[1] ferber@lirmm.fr

[1]Laboratoire d'Informatique, de robotique
et de microéléctronique de Montpellier
161 rue Ada 34392 Montpellier Cedex 5– France

Résumé

Nous présentons un cadre conceptuel et architectural pour la gestion des systèmes multi-agents (SMA) ouverts et dynamiques. Notre approche est fondée sur la notion de service, qui permet aux agents d'exprimer leurs besoins et compétences au travers de descriptions de rôles facile à publier et rechercher. Nous nous focalisons sur le modèle AGR que nous étendons afin de prendre en compte le concept de Service. Dans notre modèle, le rôle décrit les services qu'il fournit. Ces services sont exécutés par les agents joueurs du rôle et utilisés par ses agents utilisateurs. Les services fournis par un rôle sont publiés dans le cadre de la description de son groupe. L'architecture proposée et les concepts introduits ont été testés et validés en utilisant Madkit et JXTA.

***Mots-clés** : AGR, Agent Groupe Role Service, JXTA P2P, Madkit, systèmes multi-agents ouverts.*

Abstract

In this paper we present a conceptual and architectural framework for open and dynamic multi-agent systems' management. Our approach uses the service that allows to agents to present their requirements and competencies using roles' description easy to publish and research. We focus on the AGR model which we propose to extend in order to treat the service concept. In our model, the role describes the services it offers. These services are executed by the agent players of the role and used by the users' agents of the role. The services offered by a role are published with its group's description. The proposed architecture and concepts have been tested and validated using the Madkit and JXTA platforms.

***Keywords**: AGR, Agent Group Role Service, JXTA, P2P, Madkit, Open multi-agent systems*

1 Introduction

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 [1]. 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) [2], tantôt du point de vue de l'agent considéré (dans ce cas) comme une entité auquel des rôles sont rattachés [3]. Certains travaux essaient de trouver une association entre ces différentes appréhensions du concept de rôle et de son utilisation dans les organisations SMA [4]. Le développement d'applications ouvertes, réparties et dynamiques mettant en oeuvre un grand nombre de machines interconnectées par des réseaux généraux, notamment l'Internet ou la grille, est un domaine de recherche en plein essor. Dans ce contexte, les systèmes multi-agents organisationnels peuvent offrir une solution intéressante répondant à plusieurs besoins. Cependant, la plupart des approches existantes sont fermées dans le sens où elles ne considèrent les rôles et groupes que dans le cadre d'une seule application conçue par un seul développeur et isolée du reste du réseau. Afin d'exploiter au mieux les organisations et les mettre au service de systèmes ouverts nous proposons d'exploiter différemment les concepts de rôles et groupes. Nous suggérons de réétudier les fonctions incombées aux rôles dans les modèles organisationnels agents et de mieux explorer la relation rôle-agent. Nous étendons le modèle AGR [5] pour proposer un modèle organisationnel répondant à des besoins de plus en plus ressentis dans les systèmes multi-agents déployés à grande échelle, ouverts et très dynamiques[6].

Afin de faciliter la connexion de différentes applications nous préconisons l'utilisation des services qu'exécuteront les agents joueurs d'un rôle. Ces services seront exploités par les utilisateurs du rôle qui est donc considéré comme un moyen d'interactions entre agents joueurs et utilisateurs. Le rôle est donc une interface entre agents capables de produire des services d'une part et d'agents à la recherche d'un ou plusieurs de ses services. Nous considérons qu'étudier la relation entre agents et rôles uniquement de la vision « Agent joueur du rôle » ne couvre pas l'ensemble des utilisations réelles du rôle dans les applications basées sur les approches agents. L'intérêt de la vision « Agent utilisateur du rôle » est de mettre en lumière et cadrer au niveau organisationnel des interactions (agent utilisateur du rôle et rôle) souvent négligées et laissées à la charge des développeurs. A l'inverse de l'utilisation d'un rôle, jouer un rôle est une action qui dure dans le temps et qui a un état qui doit être géré indépendamment de l'agent joueur et du rôle. Cet état est le résultat de l'association de l'agent et du rôle.

En revanche, cette vision ne peut être exploitable dans des systèmes ouverts et hétérogènes sans faciliter l'exploration et découverte des services offerts par les différents rôles du réseau. Pour ce faire, les rôles sont décrits à travers les services qu'ils offrent. Le groupe reste dans notre cas le cadre de toute activité sociale entre les agents qui interagissent directement ou à travers les rôles du groupe. Les agents entrent dans un groupe pour atteindre des objectifs et satisfaire des besoins et doivent, pour ce faire, jouer ou utiliser des rôles. La description d'un rôle ainsi que sa publication dans le réseau doit se faire dans le cadre du groupe auquel il est rattaché. Les agents découvrent les services d'un groupe et l'intègrent pour jouer ou utiliser le ou les rôles qui offrent ces services.

On a tendance à qualifier d'abstrait et de structures sociales stables, les rôles dans les systèmes multi-agents. Mais en réalité, dans leurs utilisations dans les SMA les rôles ne sont ni abstraits ni stables. Ils évoluent et changent selon l'évolution du contexte dans lequel ils sont déployés. Associer des règles aux rôles assurant la gestion de leurs évolutions est pour nous une solution qui assure une souplesse au niveau de la modélisation et résout des problèmes récurrents dans les systèmes dynamiques.

Notre proposition tente de réduire au maximum l'intervention du concepteur au niveau de la modélisation des agents et plus particulièrement ceux déployés pour les applications ouvertes et dynamiques. Pour cela nous proposons l'utilisation des outils de publication et de recherche permettant aux agents de retrouver les meilleurs groupes et jouer ou utiliser les meilleurs rôles de façons décentralisée et transparente. Nous utilisons le langage XML et l'outil P2P JXTA [7] sur la plateforme Madkit [8] pour implémenter notre modèle.

Ce document est structuré comme suit, nous ferons l'état de l'art des approches organisationnelles basés sur la notion de rôle en section deux. Dans la section trois, nous définissons la relation entre agents et rôles. Nous donnerons dans la quatrième et cinquième section notre proposition étendant respectivement les concepts de rôle et de groupe du modèle AGR. Dans la section six, nous expliquons les mécanismes utilisés pour mettre en œuvre notre modèle. Enfin, en septième section nous concluons ce document en donnant quelques perspectives.

2 Aperçu des approches existantes

Dans cette section nous proposons un aperçu des différentes approches organisationnelles des systèmes multi-agents utilisant le concept de rôle. Chacune de ces approches abordent 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 tel que la relation entre agents et rôles.

Dans RoMAS [9] 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'approche prétend que les rôles existent durant toutes les phases de l'analyse au développement et RoMAS génère les rôles dès la phase d'analyse et plus précisément des cas d'utilisations.

Le modèle AGR [5] [10] est basé sur l'association de trois concepts clés : l'agent, le groupe et le rôle. Dans AGR un rôle est une représentation abstraite d'une fonction, 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 à un niveau organisationnel servant à encadrer leurs interactions et facilitant leurs localisations. La pauvreté descriptive au niveau des rôles si elle a l'avantage d'être générique, oblige néanmoins d'avoir une dépendance assez forte entre le rôle et les agents qui le jouent. En effet, c'est aux agents que sera incombé la tâche d'implémenter les comportements, services droits et obligations attendus par le rôle. Le concept de groupe permet d'avoir une séparation claire des modules d'une application et offre une grande clarté organisationnelle. Nous partons de ces inconvénients pour bâtir notre modèle.

Colman [2] étudie le rôle du point de vue organisationnel où il est vu comme « exécutant une fonction dans l'organisation ». 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. Le travail définit quatre types de dépendance entre rôle et agent qui le joue 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. Cette approche a le mérite de se pencher sur le problème des limites qui existent entre le rôle et l'agent qui le joue.

Odell [4] propose l'utilisation de classificateur 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 définit pour un agent quels types de rôles il peut jouer à un instant donné. 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.

Cabri&al dans le cadre du framework BRAIN [11] 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. Deux approches d'infrastructures d'interactions : RoleSystem [12] et RoleX [13] sont implémentés pour mettre en œuvre la définition de rôle précédemment énoncée.

RoleSystem est une approche centralisée où la relation s'effectue entre un rôle et ses agents et 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.

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 à ses besoins.

Les différentes approches présentées affectent un agent à un rôle sans préciser les raisons de cette affectation. En affectant, dès le stade de la conception, des agents à certains rôles on fige le système. Ces approches n'expliquent pas comment ni pourquoi les agents interagissent dans le cadre d'un rôle. Des agents doivent interagir dans le cadre d'un rôle pour atteindre un objectif (négociation, délégation, envoi de message…). Nous basons notre travail à partir de ces constations :

- Un agent joue un rôle pour acquérir des compétences, avoir des droits et offrir des services à d'autres agents en contre partie.
- Les agents utilisent le rôle pour interagir avec un de ses joueurs et obtenir un service.
- Le rôle décrit ce qu'on attend des agents qui vont le jouer et ce qu'obtiendront ceux qui vont l'utiliser. Les interactions entre agents dans le cadre du rôle sont donc définies.

Les deux approches de BRAIN, RoleSystem et RoleX sont les plus proches de la vision qu'on se fait 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. Mais ces deux approches n'apportent pas les réponses à toutes les questions qu'on a déjà posées dans l'introduction. Notamment, elle ne traitent pas les agents utilisant un rôle, ne séparent pas rôle, agent et agent jouant le rôle en mettant tout dans (RoleSystem) ou tout dans l'agent dans (RoleX). D'autre part, fusionner agent et rôle [14] 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. Nous abordons le rôle des points de vue joueur et utilisateur dans la section suivante et tout au long de ce travail nous désignerons par :

- Joueur de rôle : Tout agent qui s'inscrit comme joueur du rôle fourni les services proposés par ce dernier.
- Utilisateur de rôle : Un agent utilise un rôle pour entrer en contact avec un agent joueur du rôle afin d'obtenir un ou plusieurs services.

3 Le rôle, frontière entre agents qui le jouent et ceux qui l'utilisent

Nous désirons dans le présent modèle enrichir le concept de rôle en décrivant dès la 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. Le premier saura quels services il devra offrir, le second quels services il pourra demander. 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 [2].
- Vue centrée Organisation : Dans la deuxième vision le rôle est l'entité stable et les agents lui sont rattachés [3] [5].

Même si on associe les deux visions [15] 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 demande « 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.

Principe1: *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.

Principe2: *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.*

4 Redéfinition du concept de rôle dans AGR

L'analyse précédente montre que a) il est important d'enrichir la définition du concept de rôle b) de bien préciser ce qu'on attend du rôle (coté utilisateur) et c) qu'est ce qu'on y gagne coté joueur. d) Il faut préciser qui peut jouer ou utiliser un rôle donné et sous quelles conditions. e) Il faudra aussi assurer la dynamique d'un rôle à un niveau assez haut d'abstraction et ce dès sa conception. Enfin f) 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.

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.

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

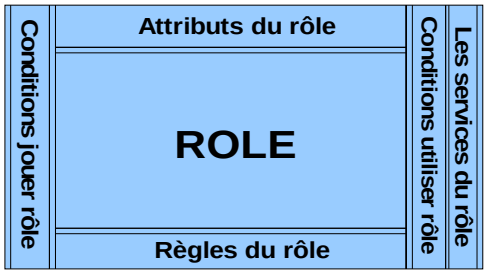

FIG.1 Architecture du rôle

4.1 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) [5] pour les agents en spécifiant pour chaque classe d'agent 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 des recherches, en continue, 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 dans 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.

4.2 Les services d'un rôle

Les services constituent dans notre vision la substantifique moelle des rôles. Un service est un ensemble de fonctionnalités que doit proposer ou demander un agent. Nous voulons

que tout soit exprimable en terme de service, même 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. D'un point de vue programmation le service est un objet (au sens java) 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.

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 auquel il appartient ou ailleurs. Nous expliquerons dans la suite comment les services sont recherchés puis utilisés.

Nous distinguons trois types de service : offerts, fournis et transférables. 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. Un agent client pour payer sa consommation 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.

Pour conclure, notons que ce ne sont pas les services eux-mêmes qui sont importants mais plutôt les fonctionnalités qu'ils représentent. Les services sont décrits (en xml) et implémentés sous forme d'objets (en java). Chaque agent ou rôle peut avoir une implémentation différente d'un service. Des mécanismes de certification du résultat de chaque service sont nécessaires.

4.3 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 attributs sont partagés par tous les rôles et font donc partie de la classe abstraite Rôle. 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. Les variables qui ont trait à l'évolution de l'agent joueur dans son rôle ne sont pas rattachés au rôle mais à ce qu'on appellera dans la suite « AgentInRole ».

4.4 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 [16], 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 propriétés 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 aussi peuvent 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 ses services et peuvent elles même être modifiées.

4.5 Jouer 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. 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. Par exemple un agent qui joue le rôle de serveur mémorise les commandes des clients. 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 qu'on appellera « AgentInRole ». L'agent est 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 mais l'exécute dans AgentInRôle.

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.

Ci-dessous le schéma UML modélisant la relation entre Rôle, Agent et AgentInRole au niveau classes du modèle et de l'exemple. Remarquons l'existence d'un patron de conception de type abstraction-occurrence [17] entre Rôle et AgentInRôle. Cette relation est justifiée par le fait qu'un rôle est à la fois une abstraction du concept AgentInRole, qui constitue son exécution réelles, et qu'à chaque rôle sont associés plusieurs occurrences AgentInRôles.

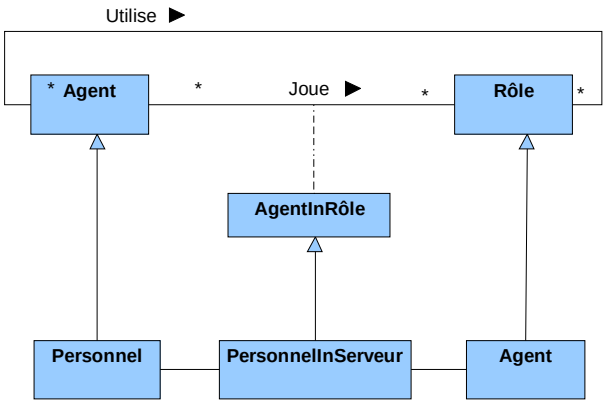

FIG. 2 Diagramme des classes : Agent Role AgentInRole

la classe Agent est liée avec la classe Rôle par deux relations: utilisation qui est directe et joueur qui passe par la classe AgentInRole. La classe Personnel hérite de la classe agent et la classe PersonnelInServeur est une extension de AgentInRole.

La figure 3 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 'ServeurResto'. Après vérification des conditions, l'agent 'Eric' devient joueur du rôle 'ServeurResto'. Le rôle serveur crée la relation 'EricServeur' instance de la classe 'PersonnelServeur' et qui sera le contexte d'exécution du rôle 'ServeurResto' par l'agent 'Eric'. Un agent client 'Jhon' désire utiliser le rôle Serveur pour avoir le service 'Menu'. Le rôle 'ServeurResto' vérifie qu'il peut utiliser le

rôle et lui indique la relation ‘EricServeur’ dans laquelle il pourra lui l’obtenir. L’agent ‘Jhon’ demande à traver ‘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: ‘EricServeur’. Notons que toute demande d’utilisation d’un service ainsi que la réponse à cette requête, peuvent être sauvegardées dans l’entité AgentInRôle.

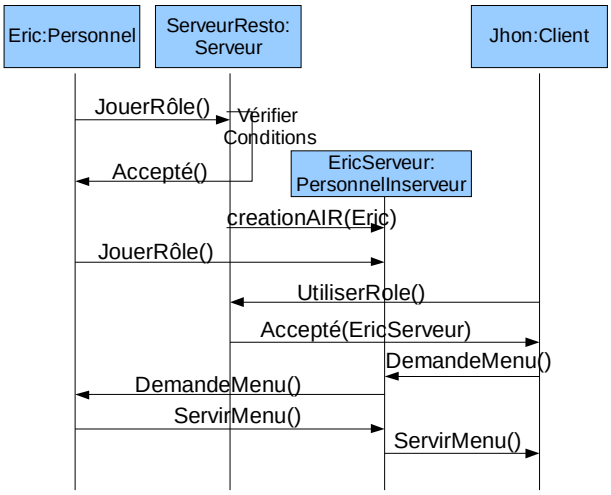

FIG. 3 Diagramme de séquences : Jouer et utiliser le rôle serveur

5 Redéfinition du concept de groupe dans AGR

Dans le modèle AGR [5] le groupe est un ensemble d’agents qui partagent certaines caractéristiques communes. Il est utilisé comme contexte pour l’activité des agents et moyen de partition des organisations. Malheureusement on ne trouve pas des indications sur comment les agents découvrent et joignent les groupes, où on va garder l’information décrivant le groupe et comment l’exploiter par les agents. Nous souhaitons avoir des groupes ouverts favorisant la dynamique mais rien ne met ça en œuvre 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.

5.1 Contraintes et contexte

Groupes ouverts : Les groupes peuvent accueillir de nouveaux membres. Ces membres peuvent provenir de différents environnements. Les groupes doivent être visibles afin d’être découverts et donc accessibles par les agents. La visibilité d’un groupe inclut, les moyens d’y accéder tout comme la description des services proposés par chacun de ses rôles.

Groupes dynamiques : La structure du groupe est modifiable à tout instant. Les conditions d’entrer au groupe, d’octroi des rôles ainsi que leurs utilisations doivent être facilement modifiables et sans intervention externe. Des rôles peuvent être supprimés, d’autres peuvent être ajoutés et créés à la demande des agents selon leurs besoins et objectifs.

5.2 Solutions apportées

La description du groupe s’effectue à partir de ses rôles. Les agents joignent le groupe pour jouer ou utiliser ses rôles. 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. La figure ci-dessous, illustre la relation entre agent, groupe, rôle et AgentInRôle.

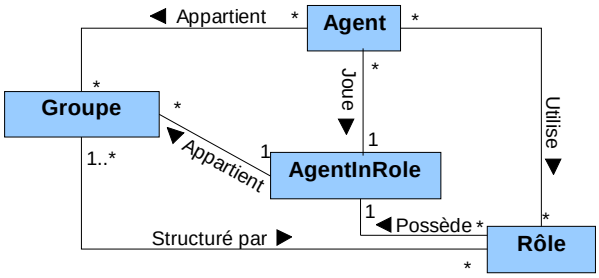

FIG. 4 Modélisation UML simplifié du modèle

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.

5.3 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 : 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: Les règles qui gèrent le groupe et sa dynamique elles utilisent les conditions et les attributs pour modifier l'état du groupe.

A chaque rôle correspond une relation « AgentInRole ». La structure du groupe contient les rôles, les relations « AgentInRôle » mais aussi les services des rôles. L'existence de la description des services du groupe se justifie par le fait que les agents effectuent les recherches par service. Une fois un service trouvé il leur faut rejoindre le groupe le contenant et jouer ou utiliser le rôle proposant le service. Il est donc plus pratique d'avoir le service, le rôle, et le groupe le contenant dans une même description.

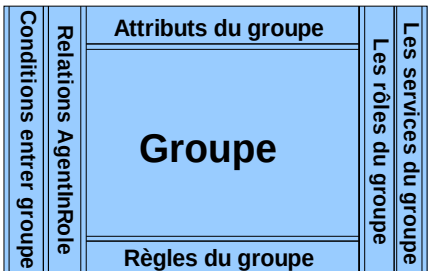

FIG. 5 Architecture du Groupe

6 Mise en oeuvre

Pour notre modèle, nous avons décidé d'implémenter notre architecture sur la plateforme multi-agents Madkit [8]. Nous avons développé un agent responsable de la publication et de la recherche de services. On appellera cet agent : « JXTACommunicator », il se base sur la technologie Peer-To-Peer (P2P) JXTA. Chaque groupe a une description qui est publié par l'agent « JXTACommunicator » de sa plateforme sous forme d'un advertissement (fichier XML structuré selon les spécifications JXTA). La description contient l'identifiant du groupe, ses services et ses conditions d'entrée. Les règles du groupe définissent les fréquences de mise à jour des descriptions sur le réseau. Sans entrer dans les détails de développement, notons que le « JXTACommunicator » de chaque plateforme est capable de rechercher un service et de renvoyer la liste des groupes contenant ce service à l'agent demandeur. Les plateformes sont groupées dans des réseaux privés. Un réseau privé est un groupe au sens JXTA invisible au reste du réseau et où ses membres peuvent communiquer. Une plateforme est représentée par son « JXTACommunicator » dans les différents réseaux auxquels elle appartient. Comme le montre la figure6 une plateforme n'est visible que des plateformes appartenant aux réseaux JXTA auxquels elle appartient.

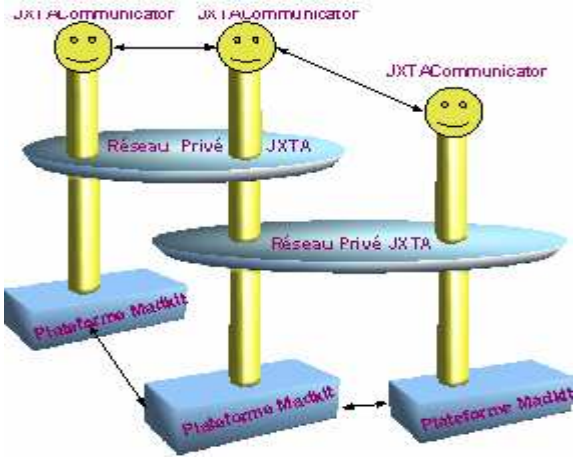

FIG.6 Plateformes Madkit sur un réseau JXTA

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.

Tous les agents systèmes de la plateforme utilisent la même logique de fourniture et utilisation de services. En particulier, l'agent « JXTACommunicator » joue le rôle de « communicator » et fournira les services « recherche » et « publication » et « communication ». Cette technique est assez flexible pour avoir différents types d'agents jouant le rôle de Communication.

7 Conclusion

La conception, développement et déploiement à grande échelle de systèmes multi-agents ouverts et dynamiques souffrent de problèmes occasionnés par la multiplicité de préoccupations à prendre en compte simultanément au niveau de la conception et

du développement. Dans de tels systèmes les agents ne se connaissent pas à l'avance, les services offerts par chaque agent ainsi que sa position sociale n'est pas figée. Toute solution doit prendre en compte ces paramètres et être capable d'organiser les agents, faciliter leurs communications et les aider à atteindre leurs objectifs. Nous avons cherché à importer les concepts du P2P dans le monde des SMA. Notre étude a porté sur le modèle AGR choisi pour sa généricité et simplicité.

Cette étude nous a permis de constater que le concept de rôle doit être mieux exploiter afin de répondre à nos besoins. Cela nous a amené à comparer différents modèles organisationnels à base de rôle et de proposer une extension du modèle AGR mieux adaptés aux systèmes ouverts dynamiques. L'extension que nous proposons repose sur de nouveaux concepts comme services, AgentInRole et utilisation de rôle. Cela permet de considérer les rôles autant du point de vue joueur qu'utilisateur. Le rôle est vu comme une frontière entre agents exécutant des services, et les agents utilisateurs des services. Le rôle décrit l'ensemble des services que peuvent utiliser les agents utilisateurs et que fourniront ses agents joueurs. Pour jouer ou utiliser un rôle l'agent doit entrer dans le groupe qui le contient. L'agent joue un rôle dans l'entité « AgentInRole », encapsulant les variables qui concernent le cycle de l'exécution du rôle. Afin de permettre la dynamique, les groupes, rôles, et « AgentInRôle » sont gérés par des règles.

Enfin nous adoptons une philosophie P2P dans la mesure où les services de chaque groupe sont publiés dans le réseau puis découverts par les agents. Les groupes sont décrits à partir de leurs services à travers des structures XML. Ces descriptions sont englobées dans des « Advertisements JXTA» pour être publiées.

Références

[1] J. Odell, H. Van Dyke Parunak, and M.Fleischer, The Role of Roles in Designing Effective Agent Organizations, *Software Engineering for Large-Scale MultiAgent Systems*, Alessandro Garcia et al, LNCS, Springer, 2003.

[2] G. Cabri, L. Ferrari and F. Zambonelli, Role-Based Approaches for Engineering Interactions in Large-Scale Multi-agent Systems, *Lecture Notes in Computer science*, pp 243-263, 2004.

[3] A. Colman, J. Han, Organizational roles and players, *AAAI Fall Symposium, Roles, an Interdisciplinary Perspective, Arlington*, pp. 55-62, 2005

[4] J. Odell, Marian H. Nodine, Renato Levy, A Metamodel for Agents, Roles, and Groups. AOSE 2004: 78-92

[5] J, Ferber, O. Gutknecht, and F. Michel, From Agents to Organizations, an Organizational View of Multi-Agent Systems Agent-Oriented Software Engineering (AOSE) IV, P. Giorgini, Jörg Müller, James Odell, eds, Melbourne, July 2003, LNCS 2935, pp. 214-230, 2004.

[6] G. Carvalho., Governance Frameworks for Open Multi-Agent Systems, *The 5th International Joint Conference on Autonomous Agents and MultiAgent Systems* (AAMAS'06)

[7] J. D. Gradecki, Mastering JXTA: Building Java Peer-to-Peer Applications, *John Wiley & Sons*, 528 pages, 2002.

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

[9] Q. Yan., L.J. Shan, X.J.Mao., RoMAS: A Role-Based Modeling Method for Multi-Agent System. Proceedings of International Conference on Active Media Technology 2003

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

[11] G. Cabri, L. Leonardi and F. Zambonelli BRAIN: A Framework for Flexible Role-Based Interactions in Multiagent Systems Lecture notes in computer science, pages 145-161, 2003

[12] G. Cabri, L. Leonardi and F. Zambonelli Implementing Role-based Interactions for Internet Agents, *Proceedings of the 2003 Symposium on Applications and the Internet*.

[13] G. Cabri, L. FERRARI L. Leonardi M. Klusch, S. Ossowski, V. Kashyap, R. Unland, The RoleX environment for multi-agent cooperation, *International workshop on cooperative information agents N°8*, Erfurt , Allemagne 2004

[14] G. Cabri , L. Ferrari , L. Leonardi, Exploiting runtime bytecode manipulation to add roles to Java agents, *Science of Computer Programming, v.54 n.1, pp. 73-98*, January 2005

[15] J. J. Odell, H. Van Dyke Parunak, and M.Fleischer. Temporal Aspects of Dynamic Role Assignment. *Agent-Oriented Software Engineering (AOSE) IV, Lecture Notes on Computer Science volume 2935, Springer, Berlin, 2004*

[16] G. Boella, R. Damiano, J. Hulstijn, L.Van Der Torre, The Roles of Roles in Agent *Communication Languages International Conference on Intelligent Agent Technology* pp. 381-384. 2006.

[17] T.Lethbridge., *Object-oriented Software Engineering: Practical Software Development Using Uml And Java*, Mcgraw-Hill College; Édition : 2Rev Ed Décembre 2004, pp528