RIO : Rôles, Interactions et Organisations

Philippe Mathieu Jean-Christophe Routier Yann Secq
mathieu@lifl.fr routier@lifl.fr secq@lifl.fr

Laboratoire d'Informatique Fondamentale de Lille, UMR 8022
Université des Sciences et Technologies de Lille
France

Résumé :
Les notions de rôle et d'organisation ont souvent été mises en avant dans les méthodologies orientées agents. Malheureusement, la notion d'interaction n'a que rarement été réifiée dans ces méthodologies. C'est pourquoi après avoir présenté succinctement deux méthodologies multi-agents et les avoir mises en relation avec les approches orientées interactions, nous proposons un formalisme pour la spécification de protocoles d'interaction entre micro-rôles. Ces derniers sont assemblés en rôles composites qui sont ensuite attribués aux agents du système. Cette spécification est rendue exécutable par la génération de Réseaux de Pétri Colorés pour chacun des micro-rôles participants. Cette spécification exécutable et l'utilisation d'un modèle générique d'agent nous permettent de proposer la méthodologie RIO facilitant la conception, la réalisation et le déploiement effectif de systèmes multi-agents.

Mots-clés : Conception orientée interactions, rôles et organisations, coordination d'agents, intergiciel sémantique.

Abstract:
Notions of role and organisation have often been emphasized in several agent oriented methodologies. Sadly, the notion of interaction has seldom been reified in these methodologies. We have defined the notion of runnable specification of interaction protocols. Then, we propose a methodology for the design of open multi-agent systems by defining an engineering of interaction protocols. In the first part of this paper, we briefly present two methodologies and we relate them to interaction-based works. In the second part, we introduce our runnable specification of interaction protocols. This specification is runnable thanks to a projection mecanism that translates it into Colored Petri Nets objects. Then, in the last section, we present RIO, our methodology to ease the design and implementation of open multi-agent systems..

Keywords: Interaction-based design, roles and organizations, agent coordination, semantic middleware.

1 Introduction

L'idée d'un génie logiciel orienté agents est apparue il y a maintenant une dizaine d'année avec l'article de Shoham, *Agent Oriented Programming* [18]. Depuis, plusieurs méthodologies ont été proposées pour guider les concepteurs dans leur tâche d'analyse et de conception [9, 2]. Pour cela, les notions de rôle, d'interaction et d'organisation sont souvent mises en avant, pour faciliter la décomposition et la description de systèmes complexes distribués ouverts.

Cependant, nous pensons que les méthodologies proposées n'identifient pas clairement les différents niveaux d'abstraction permettant de décomposer un système et surtout ne proposent généralement pas de concept ou de principe pragmatique facilitant la réalisation de tels systèmes.

Particulièrement, la notion d'interaction qui intervient dans la plupart des méthodologies proposées n'est généralement pas réifiée[1], et n'intervient que lors de la phase d'analyse. Ainsi, notre proposition repose sur un modèle générique d'agent minimal et un modèle de spécification exécutable d'interactions. Le modèle d'agent est l'infrastructure permettant le déploiement et la gestion des d'interactions, tandis que la spécification des interactions décrit une vue globale des conversations intervenants entre les agents du système.

Dans la première partie de cet article, nous présentons succinctement deux méthodologies qui ont été proposées pour l'utilisation de systèmes multi-agents lors de la conception de systèmes complexes distribués, puis nous les mettons en relation avec les approches orientées interactions.

Dans la seconde partie, nous proposons un modèle générique d'agent minimal et un formalisme pour la spécification de protocoles d'interaction entre micro-rôles. Ces derniers sont assemblés en rôles composites qui sont ensuite attribués aux agents du système. Cette spécification est rendue exécutable par la génération de Réseaux de Pétri Colorés pour chacun des micro-rôles participants. Cette spécification exécutable et l'utilisation d'un modèle générique d'agent nous permettent de proposer la méthodologie RIO facilitant la conception, la réalisation et le déploiement effectif de systèmes multi-agents.

[1] C'est-à-dire que l'interaction n'est pas un objet de premier ordre, se traduisant par une structure manipulable dynamiquement lors de l'exécution du système.

2 Conception orientée agents et conception orientée interactions

Plusieurs méthodologies de conception orientée agents ont été proposées, nous nous sommes attachés à l'étude de deux d'entre elles : la proposition de Ferber[4], puis celle de Jennings et Wooldridge[21]. Il est important de remarquer que ces deux méthodologies ne font aucune supposition sur le(s) modèle(s) d'agent(s) utilisé(s) et se concentrent sur la décomposition en terme de rôles d'un système complexe.

Ce point est fondamental, notamment à cause de la multiplicité de modèles d'agents et de systèmes multi-agents disponibles. Cette multiplicité rend la tâche du développeur ardue : quel modèle d'agent utiliser ? Quel modèle organisationnel choisir ? En effet, chaque plateforme impose bien trop souvent son propre modèle d'agent, ainsi que son modèle organisationnel. Dans un second temps, nous mettrons ces travaux en relation avec différentes approches orientées interactions, qui ont l'avantage de par leur nature d'être neutre vis-à-vis des modèles d'agent ou d'organisation.

2.1 Les méthodologies Aalaadin et GAIA

Le modèle proposé par Ferber, appelé AALAADIN, repose sur les notions de rôle, de groupe et d'agent. Un agent est une entité communicante qui joue un ou plusieurs rôles dans des groupes. Un groupe est un ensemble atomique d'agents. Chaque agent peut appartenir à différents groupes et les groupes peuvent se chevaucher. Enfin, un rôle dans Aalaadin est une représentation abstraite de la fonction, du service ou tout simplement l'identificateur d'un agent au sein d'un groupe. Chaque agent peut tenir plusieurs rôles, mais chaque rôle est local à un groupe. La communication entre les agents n'est possible qu'au travers des rôles qu'ils assument et par conséquent, le contrôle sur les communications est effectué par le groupe.

Ainsi, dans AALAADIN, une organisation est vue comme un cadre (c'est-à-dire un *framework*) pour les activités et les interactions grâce à la définition de groupes, de rôles et de leurs relations. La notion d'organisation correspond donc ici à une relation structurelle entre les rôles définis au sein des groupes. La notion d'interaction n'est pas réifiée (elle n'est pas explicitement représentée ni manipulable dans le système) et la méthode ne fournit pas de principes permettant de faciliter la décomposition d'un système complexe.

La deuxième méthodologie appelée GAIA, proposée par Jennings, Wooldridge et Kinny, est basée sur l'idée qu'un système multi-agents doit être vu comme une organisation "computationnelle" reposant sur les interactions entre les rôles. Le but de cette méthodologie est le même que celui d'AALAADIN : capturer la structure organisationnelle du système. Cependant, dans GAIA, cette structure est constituée d'un ensemble de rôles en relation, et qui interagissent selon des *motifs d'interaction*. Ainsi, cette méthode propose l'identification de deux modèles : le modèle de rôle et celui des interactions.

Un rôle dans GAIA est défini par quatre attributs :
Les responsabilités : ce sont "les invariants" du rôle, qui se décomposent en deux catégories : *liveness*, un comportement à déclencher selon certaines conditions et *safety*, qui consiste à s'assurer que certaines conditions restent valides.
Les permissions : les droits accordés à ce rôle. Ces droits concernent principalement les droits sur l'accés à l'information.
Les activités : qui correspondent au comportement "privé" de l'agent : c'est-à-dire des comportements pouvant être déclenchés sans que l'agent soit en interaction.
Les protocoles : ils définissent la structure des interactions entre les rôles.
Nous nous intéressons principalement au dernier point : la représentation des protocoles. Dans GAIA, les protocoles sont des motifs d'interaction *institutionnalisés*, c'est-à-dire décrits formellement et de manière abstraite.

La description d'un protocole dans GAIA est la suivante :

But	Description textuelle de la nature de l'interaction
Initiateur(s)	Le(s) rôle(s) responsable(s) de l'initiation de l'interaction.
Intervenant(s)	Le(s) rôle(s) impliqués dans l'interaction
Entrée(s)	Information(s) utilisée(s) par le rôle initiateur lors du déroulement du protocole
Sortie(s)	Information(s) fournie(s) ou produite(s) par le(s) rôle(s) non initiateur(s) lors de l'interaction
Effet(s) de bord	Description textuelle des traitements devant être effectués par l'initateur lors de l'interaction

Une interaction est représentée comme un protocole entre différents rôles, mais il n'y a pas de description précise des messages échangés ou encore de leur flux. La méthodologie proposée est intéressante sur bien des points, mais reste

trop générale pour pouvoir facilement passer de la phase de conception du système à sa réalisation.

De plus, les différents niveaux de la communication ne sont pas explicités dans la description des protocoles. En effet, les travaux sur les langages de communication entre agents (ACL)[5] identifient les différents niveaux constituants une conversation :

Sémantique	Description du contenu du message
Intention	Description de l'intention grâce aux performatifs
Interaction	Description de la coordination, la structure de la conversation

Pour les deux premiers niveaux, la sémantique et l'intention d'un message, on dispose d'un certain nombre de langages permettant de les exprimer, tel SL ou KIF pour les langages de contenu et KQML et FIPA-ACL pour les performatifs et la structure du message. Cependant, même en considérant des plateformes hétérogènes partageant la même ontologie, il reste difficile d'avoir des garanties sur le respect des protocoles d'interaction[2]. C'est pourquoi des travaux ont été menés sur la formalisation de cet aspect interactionnel avec plusieurs objectifs : décrire l'enchaînement des messages, avoir certaines garanties sur le déroulement d'une conversation et faciliter l'interopérabilité entre plateformes hétérogènes.

2.2 Les langages d'interaction

Pour illustrer ces approches basées sur une formalisation des interactions, nous en avons étudié trois : APRIL[13], AGENTALK[8] et COOL[1]. APRIL (AGENT PROCESS INTERACTION LANGAGE) est issu des travaux sur les langages d'acteurs[6]. C'est un langage symbolique de manipulation de processus facilitant la création de protocoles. Dans APRIL, le développeur doit concevoir un ensemble de *handlers* qui traitent chacun des messages spécifiques (la mise en relation est effectuée par *pattern matching*). Suivant les même principes, AGENTALK rajoute la possibilité de créer facilement de nouveaux protocoles par spécialisation de protocoles existants, en se reposant sur un mécanisme d'héritage. Un autre apport fondamental d'AGENTTALK est la description explicite du protocole : la conversation est représentée par un ensemble d'états et un ensemble de règles de transitions. Ce même principe a été employé dans COOL, qui propose de modéliser une conversation à l'aide d'un automate à états finis.

Dans COOL, la nécessité de l'introduction de *conventions* entre agents pour faciliter la coordination est mise en avant. Cette notion de *convention* doit être rapprochée des travaux de Shoham sur les *règles sociales*[19] et de leurs apports sur la performance globale du système. Ainsi, l'introduction de ce niveau de gestion des interactions tout en rigidifiant d'une certaine manière les interactions possibles entre agents, apporte des garanties sur la coordination et permet de réifier ces interactions. Le tableau ci-dessous, inspiré par les travaux de Singh[20], illustre les différents niveaux d'abstractions d'un système multi-agents :

Compétences applicatives	Connaissances métier
Modèles d'agents et compétences systèmes	Conception orientée agents
Gestionnaire de conversation	Conception orientée interactions
Transport des messages	Plateforme ou conteneur d'agents

Pour conclure, nous reprenons ici la définition proposée par Singh[20] de l'approche orientée interactions, qui caractérise nos objectifs :

We introduce interaction-oriented programming (IOP) as an approach to orchestrate the interactions among agents. IOP is more tractable and practical than general agent programming, especially in settings such as open information environments, where the internal details of autonomously developed agents are not available.

C'est le point de vue que nous adoptons, en proposant une méthode pragmatique de conception et de réalisation de systèmes multi-agents reposant sur la notion de spécification *exécutable* de protocoles d'interactions.

3 La conception orientée interactions

Le cœur de notre proposition est un modèle formel de description de protocole d'interaction, et un mécanisme de transformation générant le code nécessaire à la gestion de ces protocoles. Pour que les agents d'un système en cours d'exécution puissent exploiter ces nouvelles interactions, nous nous reposons sur un modèle générique minimal d'agent, qui autorise

[2]Nous faisons ici référence aux différents protocoles définis dans les spécifications FIPA (SC00026 à SC00036).

TAB. 1 – Les quatre couches d'abstraction de notre modèle d'agent

4	Compétences applicatives	Accés aux bases de données, interface utilisateur ...
3	Compétences liées au modèle d'agent	Moteur d'inférence, moteur comportemental, ...
2	Compétences intrinsèques à la notion d'agent	Base de connaissances, Gestion des protocoles d'interaction, Gestion des organisations
1	Compétences systèmes minimales	Communication et Gestion de compétences

la construction incrémentale d'agent par ajout de compétences.

Nous allons donc dans un premier temps présenter ce modèle générique d'agent, puis dans un second temps, étudier le modèle de spécification de protocoles d'interaction et le mécanisme de transformation.

3.1 Un modèle générique d'agent minimal

Le fondement de ce modèle est d'une part la création *interactive* d'agent, et d'autre part une recherche sur les fonctionnalités fondamentales d'un agent. Nous ne nous intéressons pas à la description du comportement individuel des agents, mais plutôt à l'identification des fonctions suffisantes et nécessaires à un agent. En effet, la gestion des interactions, la gestion des connaissances ou la gestion des organisations, ne sont pas liées au modèle d'agent, mais sont des caractéristiques intrinsèques à la notion d'agent.

Dans notre modèle, un agent est un réceptacle pouvant recevoir des compétences. Une compétence est un ensemble cohérent de fonctionnalités accessibles au travers d'une interface neutre. Cette notion de compétence est à rapprocher de la notion de capacité dans l'architecture BDI proposée par Padgham et Lambrix[15] en terme de décomposition et réutilisation. Cependant contrairement à cette approche, nous ne nous reposons pas sur un formalisme logique de description des capacités, mais sur un typage des messages émis et/ou reçu par la compétence. Ainsi, la notion de compétence s'apparente plus à celle de composant logiciel dans les technologies orientées objets. Ainsi, un agent est constitué d'un ensemble de compétences qui réalisent différentes parties de son comportement. Nous avons identifié quatre couches qui sont caractérisées par différents niveaux d'abstraction des fonctionnalités proposées (tableau 1).

Le premier niveau correspond aux compétences "systèmes", c'est-à-dire les fonctionnalités minimales permettant de bootstraper un agent : la communication (émission/réception de messages) et la gestion de compétences (acquisition/retrait dynamique de compétences)[10]. Le deuxième niveau identifie les compétences *agents* : la base de connaissances, un média d'interaction pour les compétences et le lieu de représentation des connaissances de l'agent, la gestion des protocoles d'interaction (cf. section suivante) et la gestion des organisations (cf. dernière section). Le troisième niveau correspond aux compétences définissant le modèle d'agent (réactif, BDI ...), tandis que le dernier niveau représente les compétences purement applicatives. Plus que les compétences réalisant ces différents niveaux, ce sont les fonctionnalités qu'elles représentent qui sont fondamentales : la gestion des communications, tout comme la base de connaissances peuvent être implémentées de différentes manières, par contre, il est nécessaire de disposer de ces fonctions au sein de l'agent.

Ainsi, le premier et le deuxième niveau caractérisent notre modèle générique d'agent minimal. Ce modèle est générique vis-à-vis des modèles d'agents qu'il peut réaliser, et minimal en ce sens qu'il n'est pas possible de retirer l'une des fonctionnalités sans perdre un aspect fondamental de la notion d'agent. Précisons que notre modèle est destiné aux agents coopératifs à gros grains, c'est-à-dire des agents logiciels non situés.

Une compétence est constituée de deux parties : son *interface* et son *implémentation*. L'interface spécifie les messages entrants et sortants, tandis que l'implémentation réalise les traitements de ces messages. Cette séparation découple la spécification de la réalisation, et permet ainsi d'avoir plusieurs implémentations pour une interface donnée. L'interface de compétence est définie par un ensemble de motifs de messages qu'elle accepte et produit. Ces messages doivent être discriminés, il est donc nécessaire de les typer.

```
interface := ((m_in)*, (m_out)*)*

    m_x = motif de message
```

Le typage des motifs de messages peut prendre plusieurs formes : un typage fort, qui a l'avantage de spécifier complétement les interfaces,

tandis qu'un typage faible offre plus de souplesse en ce qui concerne l'évolution de l'interface. Ainsi, si le contenu des messages est exprimé en KIF ou en DAML+OIL, un typage fort consistera en une vérification de l'intégralité du message, tandis qu'un typage faible ne le vérifiera que partiellement.

3.2 Un modèle de spécification exécutable de protocole d'interaction

De nombreux travaux ont été effectués pour spécifier des protocoles d'interaction. Récemment, AgentUML[14] a été défini comme une extension d'UML, pour spécifier les conversations entre agents, notamment en spécialisant les diagrammes de séquence. Cependant, ces spécifications nécessitent l'interprétation de développeurs, qui doivent ensuite les traduire dans leur propre système. Les travaux de Labrou et Finin[17] explorent l'utilisation des Réseaux de Petri Colorés[7] (RdPC) pour modéliser les conversations entre agents. Dans [12], la même approche est utilisée, mais la notion de Réseau de Pétri Colorés Récursifs est introduite pour faciliter la composition de conversations.

Nos travaux suivent le même principe : représenter l'interaction de manière globale, et utiliser un formalisme reconnu et établi. Cependant, contrairement aux travaux précédants, notre objectif est de produire une *spécification exécutable*. C'est-à-dire, une description du protocole d'interaction qui peut être ensuite directement intégrée dans un système s'exécutant. De plus, les RdPC sont indéniablement adaptés à la modélisation de processus concurrents, et fournissent un intéressant formalisme graphique, mais ils ne s'utilisent malheureusement pas très aisément. C'est pourquoi il nous paraît préférable de définir un langage adapté à la modélisation des protocoles d'interactions, et d'utiliser ensuite un mécanisme de projection permettant de traduire ce langage vers les RdPC.

3.3 Le modèle de spécification de protocoles d'interactions

Ce modèle a pour but de faciliter la spécification, la vérification et le déploiement de protocoles d'interactions au sein de systèmes multi-agents. Sur l'ensemble de ces phases, le concepteur à pour tâche de réaliser la spécification, les autres phases étant *automatisées*. Pour cela, nous avons défini un formalisme représentant la vue globale d'un protocole d'interaction, et un mécanisme de projection qui transforme cette vue globale en un ensemble de vues locales à chaque agent. Nous décrirons dans un premier temps la spécification de la vue globale, avant de présenter le mécanisme de projection qui génére les vues locales qu'ont les agents de l'interaction lors de l'exécution du protocole d'interaction. Les protocoles d'interactions sont ici considérés comme des lois sociales au sens de Shoham[19], cela signifie que les agents perdent une partie de leur autonomie, mais en contrepartie le système gagne en déterminisme et en fiabilité. En effet, le code qui gére le déroulement du protocole étant généré, le développeur n'a plus à interpréter le schéma d'interaction pour l'implémenter. De même, un certain nombre de cas *exceptionnels* peuvent être automatiquement gérés (abandon d'une interaction, refus de participation ou non respect du protocole par l'un des participants).

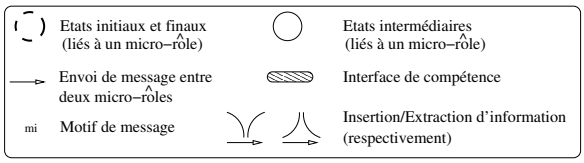

FIG. 1 – Définition des éléments syntaxiques constituant un protocole d'interaction

Notre modèle repose sur la notion de compétence, de micro-rôle et de graphe d'état de conversation (figure 1 et 2). Un protocole d'interaction spécifie formellement le déroulement d'une conversation (considérée comme une loi sociale) entre différentes entités, c'est-à-dire la nature des messages échangés, le flux de ces messages et les compétences que doivent mettre en œuvre les entités à chaque étape de la conversation. Ces entités correspondent à des micro-rôles, et sont caractérisées par leur nom et leurs compétences. On utilise un graphe pour représenter le déroulement de la conversation : les nœuds représentent les micro-rôles qui peuvent être associés à une interface de compétence, et les arcs à un envoi de message entre deux micro-rôles (typés par un motif de message). Un état dans le graphe caractérise donc partiellement un micro-rôle, en indiquant le(s) message(s) qu'il traite/génère, ainsi que la compétence qu'il doit utiliser pour traiter/produire ce(s) message(s), à un instant donné de la conversation.

Un protocole d'interaction est ainsi défini à partir des éléments suivants (figure 3) : le nom du protocole d'interaction, une description textuelle sommaire du but de ce protocole, la liste

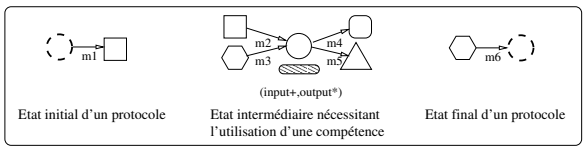

FIG. 2 – Les trois agrégats de base utilisés dans les graphes d'interactions

des micro-rôles intervenant dans l'interaction et le symbole géométrique qui leur est associé, les ontologies des messages échangés, une liste d'informations nécessaires en entrée et produites en sortie, un graphe d'interaction regroupant les informations comme le déroulement temporel de la conversation, la synchronisation et la nature des messages échangés, les compétences nécessaires aux différents micro-rôles.

Le graphe d'interaction est annoté en différents endroits, le détail de ces annotations est donné dans la dernière section. Ce qu'il est important de comprendre est que le concepteur dispose ainsi d'une vue globale de l'interaction : le flux des messages, leur nature (le typage de ces messages correspond aux annotations attachées aux arcs), les compétences nécessaires et les informations utilisées ou produites. D'autre part, toute l'information nécessaire à la gestion du protocole d'interaction est ici centralisée et peut être utilisée pour effectuer la génération de code nécessaire à la gestion de cette interaction pour chacun des micro-rôles. Le concepteur n'a donc qu'à définir le protocole d'interaction en utilisant un outil graphique, le mécanisme de projection se charge de la génération des descriptions pour chacun des micro-rôles, et le modèle générique d'agent peut ensuite exécuter ces descriptions.

3.4 Le mécanisme de projection

La section précédente a décrit le formalisme représentant les protocoles d'interactions, nous allons maintenant expliquer la transformation permettant de rendre cette spécification *exécutable*. La spécification des protocoles d'interaction donne au concepteur une vue globale de l'interaction. Notre objectif est de générer pour chacun des micro-rôles une vue locale à partir de cette vue globale, celle-ci pourra être ensuite distribuée dynamiquement aux agents du système.

Le mécanisme de projection transforme la spécification en un ensemble d'automates. Plus précisément, un automate est créé pour chaque micro-rôle. Cet automate gère le déroulement du protocole : cohérence du protocole (ordonnancement des messages), typage des messages, effets de bords (invocation de compétence). L'implémentation de ce mécanisme est réalisé en utilisant les Réseaux de Pétri Colorés. En effet, nous utilisons le typage des jetons pour représenter les motifs de messages, d'autre part nous disposons d'une bibliothèque facilitant les interactions entre les réseaux générés et les compétences des agents. En partant du graphe d'interaction, nous créons une description de Réseau de Pétri Coloré correspondant à chaque micro-rôle, et nous transformons cette description textuelle en classe Java. Cette classe est ensuite intégrée au sein d'une compétence, qui est elle même utilisée par la compétence de gestion des interactions (compétence de niveau 2 dans le tableau 1).

L'intérêt de cette approche est que le concepteur spécifie graphiquement la vue globale de l'interaction, le mécanisme de projection génére ensuite les compétences nécessaires à la gestion de cette interaction. De plus, grâce à l'acquisition dynamique de compétence, il est possible d'ajouter de nouveaux protocoles d'interaction aux agents du système en cours d'exécution.

3.5 La gestion des connaissances et des organisations

La gestion des connaissances et la gestion des organisations font aussi partie des fonctionnalités nécessaires et ne devant pas être lié au modèle d'agent. La gestion des connaissances regroupe à la fois leur représentation, le stockage des informations, le moyen d'y accéder et de les manipuler. L'implémentation de ces fonctionnalités est fortement dépendante du modèle d'agent utilisé (niveau 3 du tableau 1).

La notion d'organisation est nécessaire pour structurer les interactions qui interviennent entre les différentes entités du système. Cette notion apporte d'importants bénéfices : un moyen d'organiser logiquement les agents, un réseau de communication par défaut et un média de recherche d'agent, de rôle ou de compétence. De plus, sa réification fournit un point d'entrée dans le système permettant de visualiser et d'améliorer les interactions entre agents[11]. Nous ne détaillons pas plus ces deux aspects car ils ne sont pas fondamentaux vis-à-vis de la gestion des interactions (sauf lors de l'exécution du système, mais nous reviendrons sur ce point).

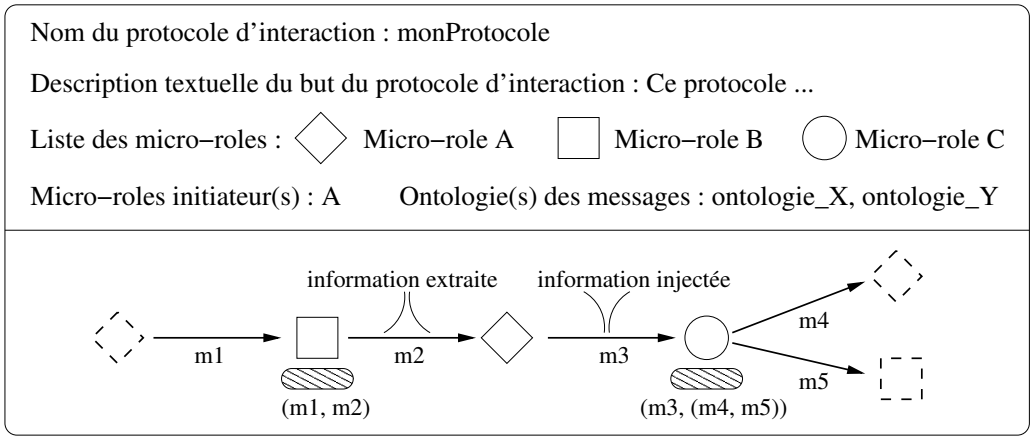

FIG. 3 – Cartouche spécificiant les protocoles d'interactions

4 RIO : vers une méthodologie orientée interactions

Dans cette section, nous allons présenter la méthodologie que nous développons, et qui repose sur la notion de spécification *exécutable* présentée précédemment. Cette méthodologie s'inscrit dans la lignée de GAIA[21], et vise donc les mêmes domaines d'applications. Cependant, GAIA reste trop générale pour pouvoir facilement passer de la phase de conception du système à sa réalisation. Notre proposition a pour but de faciliter cette transition.

La méthodologie RIO repose sur quatre phases, les deux premières représentent des spécifications réutilisables, tandis que les deux dernières sont singulières à l'application réalisée (figure 4). D'ailleurs, nous ne parlerons pas d'application, mais de société d'agents. En effet, la méthodologie RIO propose une construction incrémentale et interactive de systèmes multi-agents. Par analogie avec notre modèle minimal générique d'agent, où un agent est un réceptable pouvant accueillir des compétences, nous voyons un système multi-agents comme un réceptable devant être enrichi par des interactions. Nous allons détailler cette approche en étudiant les quatre phases de notre méthodologie.

Il est à noter que nous ne présentons ici que les phases de conception du système, nous supposons donc que le développeur a déjà effectué la nécessaire phase d'analyse. GAIA peut être utilisée pour cette première phase, et RIO intervient ensuite pour passer à la conception *effective* du système.

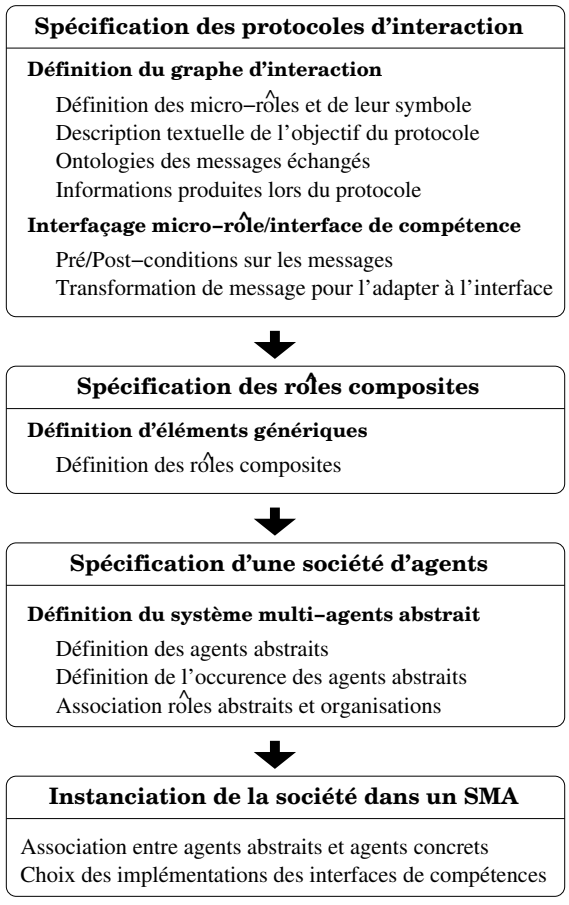

FIG. 4 – Les différentes étapes de la méthodologie RIO

4.1 La spécification des protocoles interactions

La première phase consiste à identifier les interactions et les différents rôles intervenant dans le système. Ensuite, il faut déterminer la granularité de ces interactions. En effet, pour des raisons de réutilisation de protocoles existants, il est important de trouver un équilibre entre des protocoles faisant intervenir un trop grand nombre de rôles, ces protocoles devenant ainsi trop spécifiques, et des protocoles où il n'y a que deux rôles, dans ce cas la vue de l'interaction n'est plus globale.

La spécification des protocoles d'interaction peut ensuite être effectuée de trois manières soit *ex-nihilo*, soit par spécialisation, soit par composition. La création *ex-nihilo* consiste à spécifier le protocole d'interaction en détaillant son cartouche (figure 3). La spécialisation permet d'annoter un cartouche existant. Ainsi, il est possible de spécifier le cartouche d'un protocole d'interaction tel que FIPA CONTRACT-NET, sa spécialisation consistera à changer les noms des micro-rôles pour les adapter à l'application, à affiner le typage des motifs de messages, et éventuellement à modifier les insertions/extractions d'informations. Finalement, la composition consiste à assembler des protocoles existants en spécifiant les associations de micro-rôles et les transferts d'informations.

A la fin de cette phase, le concepteur dispose d'un ensemble de protocoles d'interaction. Il peut ensuite passer à la description des rôles composites, qui va permettre l'agrégation des micro-rôles intervenant dans des interactions de même nature.

4.2 Spécification des rôles composites

Cette deuxième phase spécifie des modèles de rôle. Ces modèles sont des descriptions abstraites pouvant être réutilisées. Les rôles composites correspondent à un regroupement logique de micro-rôles. Ces patrons définissent ainsi des rôles *abstraits*, qui regroupent un ensemble de protocoles d'interaction cohérents. Par exemple, un rôle composite GESTION DE FOURNITURES regroupera les micro-rôle ACHETEUR du protocole d'interaction RECHERCHE DE FOURNISSEURS et le micro-rôle MAGASINIER de l'interaction DISTRIBUTION DE FOURNITURES.

En effet, un rôle se compose généralement d'un

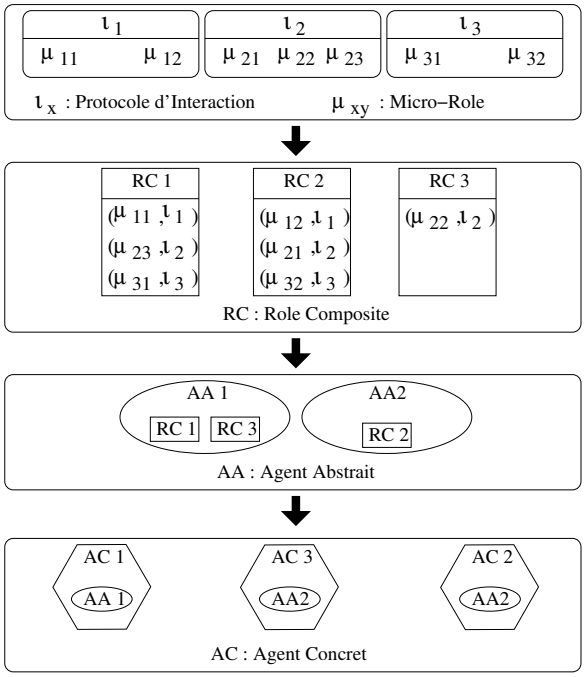

FIG. 5 – Illustration synthétique des différentes étapes de RIO

ensemble de tâches pouvant être, ou devant être effectuées par l'agent jouant ce rôle. Chacune de ces tâches peut elle même être décomposée et correspondre à un ensemble d'interactions avec d'autres rôles. La notion de rôle composite sert donc à donner une cohérence logique entre les micro-rôle représentant les différentes facettes d'un rôle.

4.3 Spécification d'une société d'agents

Cette troisième phase peut être considérée comme une spécification d'une société d'agents abstraite, c'est-à-dire une description des agents abstraits et de leur occurrence (i.e. le nombre *d'instanciation* possible de l'agent abstrait dans le système), ainsi que la liaison des rôles composites avec les organisations. Une fois que l'ensemble des rôles composites est créé, il est possible de définir les agents abstraits (patrons d'agent au sens d'*agent template*), qui sont constitués par un ensemble de rôles composites. Ces agents abstraits décrivent des modèles d'agent. Ces modèles sont spécifiques, car ils introduisent des dépendances fortes entre les rôles composites. Par exemple, le rôle composite GESTION DE FOURNITURES pourrait être associé au rôle composite de GESTION FRAIS DE MISSION, pour caractériser un agent abstrait SECRÉTAIRE. Une fois les agents abstraits définis, il faut préciser leur occurrence dans le sys-

tème. Cela revient pour chaque agent abstrait à préciser le nombre *d'instances* qui pourront être créées dans le système (exactement n agents, 1 ou plus, *, ou encore [m..n]).

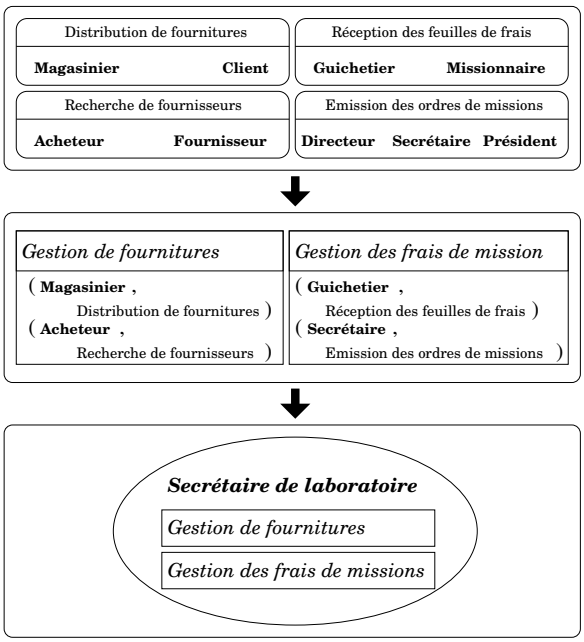

FIG. 6 – Illustration des différentes étapes de RIO

La deuxième partie de cette phase consiste à préciser pour chacun des agents abstraits, et éventuellement pour chacun des rôles composites de ces agents, quelle organisation utiliser pour trouver leurs accointances. En effet, lorsque les agents s'exécutent, pour qu'ils puissent initier des protocoles d'interactions, ils ont dans un premier temps à trouver leurs interlocuteurs. L'organisation sert de média à cette recherche. Cette association permet d'utiliser différentes organisations pour chacun des protocoles d'interaction.

4.4 Instanciation d'une société d'agents dans un système multi-agent

Cette dernière phase spécifie les règles de déploiement des rôles abstraits et des organisations sur les agents d'un système en cours d'exécution. Nous disposons d'une spécification complète de la société d'agents, qu'il faut maintenant projeter sur les agents concrets du système multi-agents. Pour cela, il faut indiquer les affectations des agents abstraits aux agents concrets. C'est lors de cette phase, que la liaison entre une interface de compétence et son implémentation se réalise. Le concepteur doit en effet, en fonction de critères propres à la plateforme d'accueil ou sur des critères applicatifs, lier les implémentation avec les interfaces de compétences avant que le déploiement ne puisse s'effectuer.

C'est lors du déploiement que le modèle générique d'agent se justifie comme support à l'ajout dynamique des nouvelles compétences liées à l'interaction. En effet, l'interaction, une fois transformée par le mécanisme de projection, est representée pour chacun des rôles par un Réseau de Pétri Coloré et des compétences associées. L'ensemble de ces informations est envoyé à l'agent, qui ajoute les compétences applicatives et délégue au gestionnaire de protocole le RdPC.

Lorsqu'un agent reçoit un message, le gestionnaire de protocole vérifie si ce message correspond à une conversation en cours (grâce à l'identificateur de conversation inclus dans le message), si c'est le cas, il délégue le traitement du message au RdPC concerné, sinon il recherche le motif de message correspondant au message reçu et instancie le RdPC correspondant. S'il n'en trouve pas, le message devra être traité par le modèle d'agent.

5 Conclusion

Nous avons présenté dans cet article une méthodologie s'inscrivant dans la lignée de GAIA, mais reposant sur les concepts de la programmation orientée interaction. Le fondement de cette méthodologie est l'ingénierie des protocoles d'interaction, et plus précisément l'ingénierie de spécifications exécutables.

Pour cela, nous utilisons un outil facilitant la conception graphique de protocoles d'interaction, et un mécanisme de transformation générant le code correspondant à la vision que chaque participant a de l'interaction. En utilisant ces spécifications, il est possible de créer des abstractions caractérisant les différents rôles et agents d'un système multi-agents : les rôles composites, qui regroupent un ensemble de micro-rôles, et les agents abstraits, qui regroupent un ensemble de rôles composites.

Une implémentation de cette approche est en cours de réalisation, et se repose sur les technologies suivantes : les Réseaux de Pétri Colorés, DAML+OIL[16] pour les ontologies de messages et la représentation des connaissances, OSGi[3] pour le modèle de composant, et le langage Java pour la plateforme multi-agents.

Références

[1] M. Barbuceanu and M. S. Fox. Cool : A language for describing coordination in multiagent systems. In *Proceedings of the First International Conference oil Multi-Agent Systems (ICMAS-95)*, pages 17–24, San Francisco, CA, 1995.

[2] F. M. T. Brazier, B. M. Dunin-Keplicz, N. R. Jennings, and J. Treur. DESIRE : Modelling multi-agent systems in a compositional formal framework. *Int Journal of Cooperative Information Systems*, 6(1) :67–94, 1997.

[3] H. Cervantes and R.S Hall. Beanome : A component model for the osgi framework. In *Workshop on Software Infrastructures for Component-Based Applications on Consumer Devices held in Lausanne in September 2002*.

[4] J. Ferber and O. Gutknecht. Operational semantics of a role-based agent architecture. In *Proceedings of ATAL'99*, jan 1999.

[5] T. Finin, R. Fritzson, D. McKay, and R. McEntire. KQML as an Agent Communication Language. In *Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM'94)*, pages 456–463, Gaithersburg, Maryland, 1994. ACM Press.

[6] C. Hewitt. Viewing control structures as patterns of passing messages. In *Artificial Intelligence : An MIT Perspective*. MIT Press, Cambridge, Massachusetts, 1979.

[7] Kurt Jensen. Coloured petri nets - basic concepts, analysis methods and practical use, vol. 1 : Basic concepts. In *EATCS Monographs on Theoretical Computer Science*, pages 1–234. Springer-Verlag : Berlin, Germany, 1992.

[8] Nobuyashu Osato Kazuhiro Kuwabara, Toru Ishida. Agentalk : Describing multiagent coordination protocols with inheritance. In *Proc. 7th International Conference on Tools with Artificial Intelligence (ICTAI'95)*, pages pp. 460–465, 1995.

[9] E. A. Kendall, M. T. Malkoun, and C. H. Jiang. A methodology for developing agent based systems. In Chengqi Zhang and Dickson Lukose, editors, *First Australian Workshop on Distributed Artificial Intelligence*, Canberra, Australia, 1995.

[10] P. Mathieu, J.C. Routier, and Y. Secq. Dynamic skill learning : A support to agent evolution. In *Proceedings of the AISB'01 Symposium on Adaptive Agents and Multi-Agent Systems*, pages 25–32, 2001.

[11] P. Mathieu, J.C. Routier, and Y. Secq. Principles for dynamic multi-agent organisations. In *Proceedings of Fifth Pacific Rim International Workshop on Multi-Agents (PRIMA2002)*, August 2002.

[12] Hamza Mazouzi, Amal El Fallah Seghrouchni, and Serge Haddad. Open protocol design for complex interactions in multi-agent systems. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 517–526. ACM Press, 2002.

[13] Frank G. McCabe and Keith L. Clark. April – agent process interaction language. In M. Wooldridge and N. R. Jennings, editors, *Intelligent Agents : Theories, Architectures, and Languages (LNAI volume 890)*, pages 324–340. Springer-Verlag : Heidelberg, Germany, 1995.

[14] J. Odell, H. Parunak, and B. Bauer. Extending uml for agents, 2000.

[15] L. Padgham and P. Lambrix. Agent capabilities : Extending bdi theory. In *Proceedings of Seventeenth National Conference on Artificial Intelligence - AAAI 2000*, pages 68–73, 2000.

[16] Filip Perich, Lalana Kagal, Harry Chen, ovrin STolia, Youyong Zou, Tim Finin, Anupam Joshi, Yun Peng, R. Scott Cost, and Charles Nicholas. ITTALKS : An application of agents in the Semantic Web. In *Engineering Societies in the Agents World II*, volume 2203 of *LNAI*, pages 175–193. Springer-Verlag, December 2001. 2nd International Workshop (ESAW'01), Prague, Czech Republic, 7 July 2001, Revised Papers.

[17] Tim Finin Yannis Labrou R. Scott Cost, Ye Chen and Yun Peng. Modeling agent conversations with colored petri nets. In *Third Conference on Autonomous Agents (Agents-99), Workshop on Agent Conversation Policies*, Seattle, May 1999. ACM Press.

[18] Y. Shoham. Agent-oriented programming. *Artificial Intelligence*, 60 :51–92, 1993.

[19] Yoav Shoham and Moshe Tennenholtz. On social laws for artificial agent societies : Off-line design. *Artificial Intelligence*, 73(1-2) :231–252, 1995.

[20] Munindar P. Singh. Toward interaction-oriented programming. In *Poster at International Conference on Multiagent Systems (ICMAS), Kyoto, Japan*, pages pp. 460–465, December 1996.

[21] M. Wooldridge, NR. Jennings, and D. Kinny. The GAIA methodology for agent-oriented analysis and design. *Journal of Autonomous Agents and Multi-Agent Systems*, 2000.