Conception par agent orientée compétences.

HABILITATION À DIRIGER DES RECHERCHES

en

Informatique

par

Jean-Christophe ROUTIER

25 novembre 2005

Jury :

Mireille CLERBOUT, Université de Lille 1, examinatrice

Guillaume DEFFUANT, CEMAGREF, rapporteur

Jacques FERBER, Université de Montpellier 2, rapporteur

Michel OCCELLO, Université de Grenoble 2, rapporteur

Christophe KOLSKI, Université de Valenciennes, examinateur

Philippe MATHIEU, Université de Lille 1, examinateur

UNIVERSITÉ DES SCIENCES ET TECHNOLOGIES DE LILLE
LABORATOIRE D'INFORMATIQUE FONDAMENTALE DE LILLE - UMR CNRS 8022
Cité Scientifique 59655 Villeneuve d'Ascq cedex

à mes parents

Le Jury

Rapporteurs

Guillaume DEFFUANT *Chercheur, Habilité*	Cemagref - LISC 24, rue des landais BP 50085 63172 Aubière Cedex.
Jacques FERBER *Professeur*	LIRMM – Université Montpellier II 161, rue ADA 34392 Montpellier Cedex 5
Michel OCCELLO *Professeur*	IUT de Valence – UPMF Département Informatique 51 rue Barthelemy de Laffemas BP 29 26901 Valence cedex 9

Examinateurs

Mireille CLERBOUT *Professeur*	LIFL – Université des Sciences et Technologies de Lille Bât M3, Cité Scientifique 59655 Villeneuve d'Ascq cedex
Christophe KOLSKI *Professeur*	LAMIH - Université de Valenciennes et du Hainaut Cambrésis Le Mont Houy 59313 Valenciennes cedex 9
Philippe MATHIEU *Professeur*	LIFL – Université des Sciences et Technologies de Lille Bât M3, Cité Scientifique 59655 Villeneuve d'Ascq cedex

Remerciements

Le travail que je vais présenter dans ce document est une synthèse de mes activités de recherche au cours de ces dernières années. Bien évidemment un tel travail n'est jamais réalisé seul. Tout un environnement y contribue directement ou indirectement. Il est donc certainement impossible de désigner nominativement toutes ces personnes, je tiens cependant à plus particulièrement adresser mes remerciements à un certain nombre d'entre elles.

En premier lieu, j'adresse mes sincères remerciements aux rapporteurs, Guillaume DEFFUANT, Jacques FERBER et Michel OCCELLO qui m'ont fait l'honneur d'accepter d'évaluer mon travail. La rédaction d'un rapport est un travail difficile, conséquent et important. Je les remercie donc sincèrement de m'avoir consacrer une partie de leur emploi du temps chargé ainsi que d'avoir jugé favorablement mon travail.

Un grand merci également aux examinateurs, Mireille CLERBOUT et Christophe KOLSKI pour l'intérêt qu'ils ont porté à mon travail et de l'honneur qu'ils m'ont fait en acceptant de participer à mon jury.

J'adresse des remerciements particuliers à Philippe MATHIEU envers qui mon estime dépasse largement son statut directeur de recherches. En m'acceptant au sein de l'équipe SMAC qu'il venait de créer il a évidemment eu un impact très fort sur ces travaux. Par son remarquable travail de direction, il facilite énormément les activités des différents agents de l'équipe et le mien particulier, nous permettant d'être aussi bien réactifs que proactifs, tout en restant aussi cognitifs que possible. Nos points de vue sur le métier d'enseignant-chercheur en général convergent le plus souvent, ce qui ne nous empêche pas d'avoir des discussions passionnées sur ce sujet. Je lui adresse donc à nouveau de très sincères et amicaux remerciements.

Les étudiants ayant participé à ces recherches sont également à remercier. Je citerai particulièrement Yann SECQ, maintenant maître de conférences, avec qui j'espère que nous continuerons d'avoir ces discussions acharnées et passionnantes. Merci également à Patrick TESSIER, Julien ACROUTE et enfin Damien DEVIGNE avec qui je collabore le plus actuellement.

Mes remerciements vont également aux autres agents de l'équipe SMAC : Bruno, Jean-Paul, Sébastien, Julien, Marie-Hélène, Cédric et Laetitia, pour leur bonne humeur et les repas chaleureux du jeudi midi.

Enfin, il est bien évident que je ne pourrai jamais suffisamment remercier mes proches pour ce qu'ils ont fait et font pour moi : mes parents pour leur éducation ; Clélie, Flore et Flavien qui acceptent que je ne leur consacre pas toujours autant de temps qu'ils le souhaiteraient ; et enfin Karine dont la présence calme et patiente est pour moi un soutien inestimable, sans les sacrifices qu'elle accepte, en particulier ces derniers mois, la tâche aurait été autrement difficile, qu'elle en soit donc ici infiniment remerciée.

Table des matières

Préambule **1**

I Synthèse **5**

1 Introduction **7**
1.1 Thèse et après-thèse 7
1.2 MAGIQUE, Rio et Cocoa 8
1.3 Cohérence et complémentarité 10
1.3.1 Cohérence de l'approche 10
1.3.2 Complémentarité des thèmes 11
1.4 Conclusion 12
Bibliographie 13

2 De MAGIQUE à RIO **17**
2.1 Le modèle d'agent minimal 18
2.1.1 Définitions 18
2.1.2 Compétences 19
2.1.3 Avantages 20
2.2 Le modèle d'organisation hiérarchique 20
2.3 API et applications 23
2.3.1 Mise en place du modèle 23
2.3.2 Délégation, accointances et échange de compétences 25
2.3.3 IDE 25
2.3.4 Exploitation 26
2.3.5 Applications 26
2.4 RIO 32
2.4.1 Spécification exécutable de protocoles d'interaction 33
2.4.2 RIO : démarche méthodologique de conception de systèmes multi-agents 34
Bibliographie 35

3 CoCoA : simulation de comportements rationnels. **41**
3.1 Simulation de comportements et jeux vidéos 42
3.2 Approche et contexte 43
3.3 Des agents définis par leurs compétences et des simulations orientées interaction 46
3.4 Les interactions 47

3.5 Les agents . . . 50
3.6 L'environnement de conception de simulations . . . 52
3.7 Equipes d'agents . . . 53
Bibliographie . . . 54

II Sélection d'articles 57

4 Dynamic Skills Learning : a Support to Agent Evolution 59
4.1 Introduction . . . 59
4.2 Agent from Atomic Agent . . . 61
4.2.1 Definitions . . . 61
4.2.2 Agent Education . . . 62
4.2.3 Advantages . . . 63
4.2.4 Role evolution . . . 63
4.2.5 Conclusion . . . 64
4.3 MAGIQUE . . . 64
4.3.1 Hierarchies . . . 64
4.3.2 Mechanism for skill delegation . . . 65
4.3.3 MAGIQUE and skills . . . 65
4.3.4 Dynamicity in MAS . . . 66
4.4 API . . . 67
4.4.1 Agent creation . . . 67
4.4.2 Skill creation . . . 68
4.4.3 Skill invokation and delegation . . . 68
4.4.4 Dynamic skill acquisition . . . 69
4.4.5 Graphical Environment . . . 69
4.4.6 Conclusion . . . 69
4.5 An application to dynamic skill learning . . . 70
4.6 Conclusion . . . 71
Bibliographie . . . 71

5 Principles for Dynamic Multi-Agent Organizations 73
5.1 Introduction . . . 73
5.2 Adapting the Architecture of the Organization . . . 75
5.2.1 Some problems with static organizations. . . . 75
5.2.2 How does social organizations manage these problems? . . . 76
5.2.3 The three principles applied to multi-agent systems . . . 77
5.3 Experiments . . . 78
5.3.1 MAGIQUE . . . 79
5.3.2 Three experiments for three principles. . . . 81
5.4 Conclusion . . . 85
Bibliographie . . . 85

6 RIO : Roles, Interactions and Organizations 87
6.1 Introduction . . . 87
6.2 Agent methodologies and interaction languages . . . 88
6.3 Interaction oriented design . . . 89
6.4 RIO : towards an interaction based methodology . . . 93
6.5 Conclusion . . . 95
Bibliographie . . . 95

7 Interaction-Based Approach for Game Agents 97
7.1 Introduction . . . 98
7.2 Knowledge Representation . . . 99
7.2.1 Environment . . . 99
7.2.2 Agents . . . 100
7.2.3 Interactions . . . 102
7.3 The Agent Engine . . . 104
7.3.1 The Agent Structure . . . 104
7.3.2 The Planning . . . 106
7.4 Conclusion . . . 112
Bibliographie . . . 112

8 Teams of cognitive agents with leader: how to let them some autonomy. 115
8.1 Introduction . . . 115
8.2 Simulations with cognitive agents . . . 116
8.2.1 Environment . . . 116
8.2.2 Interactions . . . 117
8.2.3 Agents . . . 118
8.3 Teams . . . 121
8.3.1 Teams of cognitive agents with leader . . . 122
8.3.2 Description of a team . . . 123
8.3.3 Our proposition . . . 124
8.3.4 Team of teams . . . 126
8.4 Conclusion . . . 127
Bibliographie . . . 127

III Annexe 129

Mes activités d'enseignant-chercheur 131

Préambule

Après une thèse en Programmation Logique, j'ai effectué une reconversion thématique pour travailler ces dernières années dans le domaine des Systèmes Multi-Agents au sein de l'équipe SMAC du LIFL créée en 1998 par le Professeur Philippe Mathieu. Mes travaux m'ont permis d'aborder, et de combiner, les deux thématiques qui me tiennent à cœur : le problème du développement d'applications et l'intelligence artificielle. Au sein de cette équipe, j'ai pu collaborer avec des personnes, et notamment Philippe Mathieu, avec lesquelles je partage une conception de l'activité de recherche, et plus largement du métier d'enseignant-chercheur en général.

Ainsi, dans le cadre de nos recherches, nous voulons proposer des modèles et des concepts mais sans perdre de vue la faisabilité de leur mise en œuvre. Ainsi il ne suffit pas d'avoir un discours sur ce que l'on veut obtenir et d'y apporter des propositions de solutions, il est important selon moi qu'il n'y ait pas de dichotomie avec ce que l'on peut réaliser et donc d'expérimenter ces solutions. Entendons nous bien, il ne s'agit pas de faire de l'ingénierie, fut elle de haut niveau, mais bien de proposer d'abord des nouveaux concepts ou des nouvelles approches et ensuite de les expérimenter et de les valider concrètement.

C'est cette approche que nous avons appliquée dans les deux projets principaux auxquels j'ai participé. Le premier s'appelle MAGIQUE. Il s'inscrit dans la thématique de la programmation d'applications orientée agents. MAGIQUE propose à la fois un modèle organisationnel pour les systèmes multi-agents et un modèle minimal d'agent permettant la construction incrémentale d'agents plus complexes par acquisition dynamique de compétences. Le second est le projet CoCoA. Plus orienté Intelligence Artificielle, ce projet s'intéresse à la simulation de comportements rationnels d'agents évoluant dans un environnement situé. A la différence de MAGIQUE où nous proposons un framework général ne visant pas d'aspect applicatif particulier ni de modèle comportemental des agents, CoCoA propose un moteur de comportements basé sur la notion d'interactions et spécifiquement dédié aux simulations ciblées. Le point commun entre ces différentes recherches est une approche similaire de la construction de nos agents à partir de leurs capacités, que nous appelons *compétences* dans le premier cas et *interactions* dans le second.

Ce document est une synthèse de mon activité de recherche[1] au sein de ces projets. Il se décompose en deux parties. La première est un résumé de mes travaux et la seconde est constituée d'une sélection d'articles sur ces mêmes travaux.

Il n'est pas simple dans une synthèse, notamment lorsqu'elle couvre environ six années de recherche, de choisir le niveau de détail à appliquer. J'ai donc essayé de couvrir l'ensemble de mes travaux en essayant de dégager les idées et concepts que nous avons mis en œuvre ainsi que leurs apports. L'ordre de présentation de ce document ne respecte pas nécessairement exactement

[1] Un résumé de l'ensemble de mes activités d'enseignant-chercheur est présenté en annexe à la fin de ce document.

l'historique des travaux, adapté dans un souci de progression de la présentation. De même tous les aspects de nos travaux ne reçoivent pas nécessairement une place équivalente, j'ai plus particulièrement insisté sur ceux dont la contribution me semblait la plus significative. L'existence de publications sur les différents points présentés devrait permettre au lecteur souhaitant plus de détails, de les y trouver. De plus, la partie consacrée à cette synthèse proposera deux niveaux de granularité.

Le premier chapitre propose en effet un parcours assez rapide de mes activités. J'en profite pour positionner l'ensemble des publications auxquelles j'ai contribuées ainsi que les encadrements de DEA ou thèse que j'ai effectués ces dernières années. Je m'attache également dans ce chapitre à montrer la cohérence qui existe entre les sujets de recherche que j'ai abordés, ainsi que leur complémentarité dans la thématique multi-agent.

Le second chapitre est consacré au projet MAGIQUE. J'y présente plus en détail les concepts que nous avons développés au sein de ce projet : le modèle d'agent minimal et les *compétences*, la structure organisationnelle hiérarchique du système multi-agent avec délégation automatique de requêtes, ainsi que la dynamicité de l'organisation. La mise en œuvre de ces concepts est également présentée, ainsi que la démarche méthodologique RIO pour la conception d'applications par agents. Le mémoire de DEA et la thèse de Yann Secq, soutenue en décembre 2003, s'inscrivent dans cette thématique.

Le troisième chapitre concerne le projet CoCoA. Ce projet propose un modèle pour la conception de simulations par agents situés dans lesquelles des comportements "de type humain" peuvent être simulés. Un champ d'applications naturel d'une telle recherche est le domaine des simulations comportementales dont les jeux vidéos sont un exemple. Notre proposition principale consiste à baser la description des simulations et leurs dynamiques sur la notion d'*interactions*. Dans ce projet nous nous intéressons également à la mise en place de stratégies d'équipe en nous basant sur le même formalisme. Les mémoires de DEA de Patrick Tessier, Damien Devigne et Julien Acroute, ainsi que la thèse en cours de Damien Devigne s'inscrivent dans ce projet.

La seconde partie est constituée d'une sélection de cinq articles, trois concernent le projet MAGIQUE et deux le projet CoCoA. Il a été nécessaire de faire un choix, d'abord sur le nombre de publications à placer dans cette partie et ensuite sur ces publications elles-mêmes. J'ai pris le parti d'un faible nombre de publications en orientant mon choix sur celles qui selon moi illustrent le mieux nos propositions et contributions dans chacune des thématiques. Toutes mes autres publications sont énumérées dans le paragraphe *Bibliographie* du chapitre 1, le lecteur qui le souhaite pourra donc y trouver les autres points abordés dans nos recherches mais non nécessairement présents dans les cinq publications retenues.

J'ai également décidé de laisser le texte des articles tels qu'ils ont été publiés, en langue anglaise en l'occurrence. Seule la forme a été retravaillée dans un souci d'homogénéité de ce document et j'ai donc simplement modifié la feuille de style.

Le chapitre 4 correspond à l'article présenté à la conférence AISB en 2001. C'est dans cet article que nous présentons pour la première fois notre notion d'agent atomique et d'agents construits par acquisition dynamique de compétences.

Le chapitre 5 est l'article que nous avons présenté à la conférence PRIMA2002. Nous y exposons trois principes à appliquer pour construire des organisations multi-agents dynamiques.

Le chapitre 6 a été retenu pour la conférence CEEMAS 2003. Nous y introduisons la démarche méthodologique RIO, pour "Rôles Interactions et Organisation".

Le chapitre 7 a été présenté à la 19^{e} conférence européenne sur la modélisation et la simulation, ECSM'05. Nous y proposons notre modèle d'interactions et d'agents pour des simulations par agents situés tels que les jeux vidéos.

Le chapitre 8 correspond à l'article de la conférence CIG'05. Nous y présentons comment notre modèle basé sur les interactions peut être appliqué pour modéliser des comportements d'équipes avec chef dans lesquelles les équipiers gardent une certaine autonomie de décision.

Les travaux que je présenterai dans ce document résultent d'un travail de collaboration : avec les étudiants que j'ai encadrés en DEA ou en thèse, ou avec les membres de l'équipe et en particulier Philippe Mathieu, directeur de l'équipe. C'est pourquoi j'utiliserai par la suite le "nous". La recherche est un travail coopératif : au sein d'une équipe de recherche par des échanges directs ou au sein d'une communauté à travers les publications et différentes rencontres. Il est donc bien souvent difficile de déterminer avec certitude l'exacte paternité d'une idée. Il en reste néanmoins que ces travaux sont exprimés ici selon mon point de vue qui n'engage que moi.

Partie I

Synthèse

Chapitre 1

Introduction

1.1 Thèse et après-thèse

Après avoir obtenu un diplôme d'ingénieur de l'ENSI de Caen, option "Informatique et Intelligence Artificielle", et le DEA "Intelligence Artificielle" de l'Université de Caen en 1990, c'est pendant mon service militaire en tant que VFI[1] que j'ai noué mes premiers contacts avec le LIFL[2] en 1991. J'ai ainsi rejoint pour une thèse financée par une allocation du ministère de l'enseignement supérieur et de la recherche l'équipe Méthéol[3] de ce laboratoire en septembre 1991. Cette équipe était dirigée par le Professeur Jean-Paul Delahaye et c'est sous la responsabilité de Philippe Devienne et Patrick Lebègue que s'est déroulée ma thèse.

Celle-ci concernait l'étude théorique des programmes logiques minimaux non triviaux, c'est-à-dire constitués d'une seule clause binaire récursive, d'un fait et d'un but :

$$\begin{cases} p(fait) \leftarrow \quad . \\ p(gauche) \leftarrow p(droit) \quad . \\ \leftarrow p(but) \quad . \end{cases}$$

où *fait*, *gauche*, *droit* et *but* sont des termes arbitraires. Au cours de ma thèse, nous avons résolu les problèmes, jusque là ouverts, de terminaison (Devienne, Lebègue, and Routier 1992b, Devienne, Lebègue, and Routier 1992a, Devienne, Lebègue, and Routier 1993b) et de satisfiabilité (Devienne, Lebègue, and Routier 1993a) de ces programmes. Ensuite, nous avons établi que ces programmes avaient la puissance de calcul des machines de Turing (Devienne, Patrick Lebègue, and Würtz 1994).

J'ai soutenu cette thèse (Routier 1994) le 1er février 2004 sous le titre :

Terminaison, Satisfiabilité et Pouvoir Calculatoire d'une Clause de Horn Binaire

J'ai alors été nommé en septembre 1994 à un poste de Maître de Conférences à l'UFR d'IEEA de l'Université des Sciences et Technologies de Lille, restant au sein de l'équipe Méthéol et du LIFL.

Mon activité de recherche est alors entrée dans une période délicate. Plusieurs facteurs ont contribué à son ralentissement. D'abord la charge naturelle liée aux tâches d'enseignement

[1] Volontaire Formateur Informatique
[2] Laboratoire d'Informatique Fondamentale de Lille
[3] Méthode et outils théoriques pour la programmation Logique

de tout nouvel enseignant-chercheur handicape bien souvent les premières années en poste. Il en est de même des charges administratives. Ma nomination a coïncidé à la mise en place du DEUG MIAS à l'Université de Lille 1, il fallait construire un nouvel enseignement et coordonner les équipes enseignantes, ce dont je me suis chargé avec Eric Wegrzynowski. Ce travail a débouché sur la rédaction d'un livre (Routier and Wegrzynowski 1997 - 2003 (2nde édition)) d'enseignement qui a naturellement monopolisé beaucoup de mon temps pendant une année. Même si je considère que la diffusion de connaissance et la rédaction de tels ouvrages est un des rôles des enseignants du supérieur, et si je ne regrette en rien l'expérience vécue à l'occasion de ce travail, il n'en reste pas moins que cela se fait au détriment de la recherche. Ensuite, les résultats de ma thèse offraient peu de perspectives de poursuite puisqu'ils concluaient sur des propriétés d'indécidabilité, me privant ainsi d'une dynamique immédiatement issue du travail de thèse. Cette période s'est soldée par la publication de (Devienne, Lebègue, Parrain, and Würtz 1996). Enfin, la dissolution de l'équipe de recherche à laquelle j'appartenais m'a obligé à une reconversion thématique. C'est donc ce que j'ai entrepris en 1998 en rejoignant l'équipe SMAC, "Systèmes Multi-Agents et Comportements", que venait de créer le Professeur Philippe Mathieu, également issu de l'équipe Méthéol. Je suis à l'heure actuelle toujours membre de cette équipe. Mon travail s'est depuis cette époque essentiellement réparti sur deux thématiques : d'abord le développement d'applications multi-agents physiquement distribués et ensuite la simulation de comportements rationnels à base d'agents cognitifs situés.

1.2 MAGIQUE, Rio et Cocoa

Lorsque je l'ai rejointe, un axe de travail de l'équipe SMAC portait sur le modèle d'organisation de systèmes multi-agents MAGIQUE (pour Multi-AGent hiérarchIQUE), notamment dans le cadre de la thèse de Nour-Eddine Bensaïd. Je me suis en particulier attelé à la réalisation d'une API JAVA mettant en œuvre les concepts de MAGIQUE (Mathieu and Routier 2000-2001). Ce travail a amené une réflexion sur la notion de programmation orientée agents (*Agent Oriented Software Engineering*). Une première étape a concerné l'étude de la notion de services dans les systèmes multi-agents notamment dans le cadre du mémoire de DEA de Yann Secq (Secq 1999). Ces travaux ont amené à un enrichissement du modèle MAGIQUE auquel, outre le modèle organisationnel hiérarchique existant, nous avons ajouté un modèle d'agent. En effet notre réflexion sur la notion de services nous a conduits à envisager la construction d'un agent par un enrichissement dynamique et incrémental de *compétences* qui lui permettent de rendre des *services*. Nous avons ainsi établi qu'il était possible d'obtenir n'importe quel agent à partir d'un agent élémentaire, ou minimal, simplement doté de deux compétences initiales : une compétence de communication sans laquelle il serait autiste et une compétence d'acquisition/administration de ses compétences lui permettant d'évoluer (Routier, Mathieu, and Secq 2001).

Dans le même temps nous avons étudié la dynamicité de l'organisation du système multi-agents en énonçant et expérimentant différents principes (Mathieu, Routier, and Secq 2002). Ceux-ci s'appuient à la fois sur l'architecture de MAGIQUE, dont le principal avantage est de fournir un mécanisme de délégation de services par défaut, et sur l'évolution dynamique des agents. Nous obtenons ainsi des systèmes multi-agents souples et dynamiques tant du point de vue structurel que du point de vue individuel. Ces avantages renforcent à la fois la facilité de conception des applications en soulageant le concepteur d'un certain nombre de préoccupations, et la fiabilité du système en permettant une réorganisation dynamique de celui-ci. Nous avons appliqué ces

différents principes et notre approche orientée agent de la conception d'applications à différents domaines, tels que le calcul distribué (Mathieu, Routier, and Secq 2002c, Mathieu, Routier, and Secq 2002d) et une application de travail co-opératif (Mathieu and Routier 2001, Mathieu and Routier 2002). Ces expériences nous ont permis de faire progresser notre réflexion sur une démarche méthodologique de conception des applications à base d'agents.

Nous avons ainsi proposé la méthode RIO pour "Rôles, Interactions et Organisations" (Mathieu, Routier, and Secq 2002a, Mathieu, Routier, and Secq 2003c, Mathieu, Routier, and Secq 2003e, Mathieu, Routier, and Secq 2003a) qui a constitué le cœur de la thèse de Yann Secq, que j'ai co-encadrée avec le Professeur Philipe Mathieu, et qu'il a soutenue en décembre 2003 (Secq 2 décembre 2003). Cette méthode propose un formalisme graphique de représentation des protocoles d'interaction qui en constitue une spécification exécutable (Mathieu, Routier, and Secq 2003a, Mathieu, Routier, and Secq 2003b). On y exprime notamment les rôles impliqués dans ces interactions, appelés *micro-rôles*, ainsi que les messages échangés entre les micro-rôles participants. Les micro-rôles sont ensuite regroupés pour former des *rôles composites* qui peuvent finalement être attribués aux agents effectivement impliqués dans l'application et qui sont intégrés à l'organisation choisie.

La seconde thématique que j'ai abordée concerne la simulation de comportement pour agents situés. Ce projet s'appelle COCOA pour "Cognitive Collaborative Agents". Il a été amorcé à l'occasion d'une coopération avec la société de jeu Cryo Interactive dans le cadre d'un projet PRIAMM[4] qui s'est déroulé de juin 2000 à janvier 2001 (Mathieu, Routier, and Urro 2001). Le premier objectif de cette recherche est de proposer un cadre permettant de réaliser des simulations centrées individus nécessitant la mise en œuvre de comportements rationnels. L'article (Mathieu, Picault, and Routier 2005) publié dans la revue "*Pour la Science*" présente une synthèse rapide de cette problématique. Le second objectif est la réalisation d'un environnement destiné aux Game Designers afin de les aider dans la conception de comportements évolués. Les jeux vidéos constituent, avec la présence de personnages virtuels qui doivent manifester des comportements réalistes, un domaine d'application et un contexte d'expérimentation tout indiqué. Cependant l'approche utilisée dans les jeux actuels est, malgré ses nombreux défauts, essentiellement une approche réactive basée le plus généralement sur des scripts. Au contraire, dans notre recherche nous considérons des agents cognitifs motivés par des buts, capables de raisonner sur leurs connaissances et aptes à tenir compte du contexte pour tenter de réaliser ces buts. A cette fin l'agent dispose d'un moteur de comportement qui lui permet de déterminer l'action à effectuer afin de remplir au mieux ses objectifs. Ce moteur établit un plan d'actions en fonction des connaissances de l'agent et notamment de sa situation dans l'environnement (Devigne, Mathieu, and Routier 2004). On se place donc ici dans le cadre d'applications multi-agents spatialement situées. Les agents sont positionnés et évoluent dans un environnement muni d'un espace euclidien dont ils ont une vision partielle. Cet environnement est intrinsèquement dynamique et non monotone.

La principale contribution de ce travail consiste en une approche orientée *interaction* de la modélisation de simulations. Les interactions sont des éléments de connaissance qui définissent les règles qui régissent le monde simulé. Elles spécifient dans quel contexte une action peut être réalisée et ses conséquences. Les agents impliqués dans la simulation sont caractérisés par les interactions qu'ils peuvent subir, et pour les agents *animés* par les interactions qu'ils peuvent effectuer. La dynamique des simulations est basée sur cette dualité entre les agents qui peuvent effectuer une action et d'autres qui peuvent la subir (Mathieu, Picault, and Routier 2003, Devigne,

[4] Programme pour l'Innovation dans l'Audiovisuel et le Multimédia

Mathieu, and Routier 2005b). Cette thématique est plus orientée “Intelligence Artificielle”. Nous y proposons un modèle d’agent situé cognitif. Cela nous amène à être confrontés à de nombreux problèmes de l’IA : représentation de connaissance, planification, sélection d’actions, etc. Le regroupement de ces problèmes constitue une difficulté supplémentaire.

Ce travail a notamment a été développé à travers plusieurs mémoires de DEA que j’ai co-encadrés (Tessier 2002, Devigne 2003, Acroute 2005) ainsi que la thèse en cours de Damien Devigne (Devigne 2003-en cours). Dans le cadre de cette thèse, nous considérons également la modélisation de comportements d’équipes (Devigne, Mathieu, and Routier 2005c, Devigne, Mathieu, and Routier 2005a) se basant sur le même concept d’interaction. Nous avons notamment proposé une construction de plans d’équipe avec chef laissant une part d’autonomie aux différents équipiers.

Ces travaux m’ont amené à participer à différents groupes de travail et projets. Au niveau national je participe aux groupes du GdR I3 (“Information - Interaction - Intelligence”) ASA (Architecture des Systèmes multi-Agents) et MFI (Modèles Formels de l’Interaction). J’ai également participé aux projets de CPER (contrat de plan état-région) successifs dans lesquels l’équipe était ou est actuellement impliquée : d’abord le projet *Ganymède* puis ses successeurs *COLORS* (“Composants Logiciels Réutilisables et Sûrs”), *NIPO* (“Nouvelles Interactions Personnes-Organisations”) et *Formasciences* sur les nouveaux usages, et actuellement *MIAOU* (“Modèles d’Interaction et Architectures Orientées Usages”) et *MOSAIQUES* (“MOdèles et InfraStructures pour Applications ubIQUitairES”). J’ai également participé à un projet PRIAMM en collaboration avec la société Cryo Interactive.

1.3 Cohérence et complémentarité

Comme présentés dans le paragraphe précédent, mes travaux au sein de l’équipe SMAC concernent essentiellement deux thématiques : d’abord les plateformes multi-agents et la conception d’applications distribuées par une approche multi-agent, avec le projet MAGIQUE, ensuite la simulation de comportements rationnels à base d’agents à travers le projet CoCoA. Les points communs de ces recherches dépassent, évidemment, le simple terme *agent*. Les approches des problèmes sont similaires. En même temps, les thèmes se complètent pour couvrir de multiples problématiques du multi-agent.

1.3.1 Cohérence de l’approche

La notion centrale dans le projet MAGIQUE est celle de *compétence*. Dans le projet CoCoA, il s’agit des *interactions*. Les principes portés par ces deux notions sont similaires : elles permettent de définir les compétences et rôles des agents. Dans chacun des cas, nous soutenons une approche dans laquelle les construction et définition des différents agents se font à partir d’un noyau commun par attribution de capacités. Dans MAGIQUE, les agents sont construits dynamiquement à partir de notre noyau d’agent minimal par acquisition de compétences. Dans CoCoA, les agents impliqués dans les simulations sont caractérisés par les interactions qu’ils peuvent subir ou effectuer. Les agents animés sont tous dotés du même moteur de comportements et ce sont leurs buts ainsi que les différentes interactions dont ils sont dotés qui les amènent à se comporter différemment.

Dans les deux cas on obtient une construction incrémentale des agents. Les apports de cette approche sont les mêmes. Il s'agit essentiellement d'offrir les avantages de la modularité. Les compétences d'une part et les interactions d'autres part constituent des composants potentiellement réutilisables dans différentes applications. Ils représentent les éléments de connaissances et d'expertise des applications (ou simulations) réalisées. En fait il s'agit des notions centrales de nos approches, les agents n'étant à chaque fois que des réceptacles paramétrés par ces éléments. Compétences et interactions permettent, même implicitement, d'attribuer des rôles aux agents. Dans MAGIQUE c'est parce qu'un agent détient une compétence qu'il pourra se voir déléguer la réalisation d'une requête et jouera ainsi le rôle sous-jacent. Dans CoCoA, c'est parce qu'un agent peut effectuer une interaction qu'il pourra l'utiliser pour réaliser un but et ainsi tenir le rôle implicitement associé à cette interaction.

Si les approches "compétences" et "interactions" constituent la cohérence thématique des projets, les démarches des projets sont également similaires. En particulier, nous tenons à mettre en œuvre concrètement les concepts développés. Il s'agit pour nous d'un élément important que de ne pas se limiter à émettre des idées ou concepts mais de les confronter à la réalité d'une implémentation. Nous refusons donc une démarche qui serait purement théorique et littéraire. Dans les thèmes que nous étudions une telle démarche nous semble aberrante. Cette mise en œuvre concrète oblige à considérer la faisabilité des approches et se confronter à la réalisation de ses idées oblige à considérer les problèmes jusque dans leurs détails révélant des difficultés ou subtilités qui auraient pu ne pas être soulevées dans une étude purement conceptuelle.

Que ce soit avec MAGIQUE ou avec CoCoA nous avons donc développé un contexte reprenant les idées permettant ainsi de les valider expérimentalement. L'objectif est à chaque fois le même. Il s'agit d'offrir un cadre facilitant la réalisation d'applications à base d'agents. Avec MAGIQUE, nous avons réalisé une API[5] permettant de construire des applications multi-agents distribuées. Cette API met en œuvre notre modèle d'agent minimal et propose une organisation par défaut hiérarchique des systèmes multi-agents développés, tout en permettant la mise en place d'autres modèles. Avec CoCoA nous fournissons un environnement de programmation pour la réalisation de simulations centrées individus à base d'agents situés. Cet environnement permet de créer les différents éléments d'une simulation puis, lors de l'exécution de celle-ci, d'examiner l'évolution de différents éléments et notamment celle du moteur de planification des agents animés. A travers ces développements, outre la validation des idées, nous cherchons à mettre en avant la conception d'applications à base d'agents et une approche "agent oriented software engineering". Dans les deux cas nous visons à offrir un contexte générique, un *framework*, pour les applications cibles.

1.3.2 Complémentarité des thèmes

Si, comme nous venons de le voir, il y a une certaine cohérence dans les propositions apportées dans les deux thématiques, des différences existent néanmoins dans les sujets abordés ou leur traitement. Les thèmes abordés sont également complémentaires et couvrent des aspects différents du domaine du multi-agent.

Ainsi avec le projet MAGIQUE, nous nous sommes intéressés à la notion d'architecture et de plateforme multi-agent, cherchant à offrir un contexte de développement d'applications à

[5] Application Programming Interface

base d'agents sans nous intéresser précisément au modèle interne des agents ni aux aspects applicatifs. Avec MAGIQUE nous proposons donc un framework cherchant à faciliter la construction d'applications multi-agents sans nous attacher à un type d'application en particulier. Ce framework offre les outils permettant au concepteur de l'application de se dégager d'un certain nombre de préoccupations telles que la distribution ou les communications entre les agents. Il peut se concentrer sur les aspects applicatifs essentiels. Cependant nous ne proposons pas de modèle de comportement par défaut des agents. Ainsi dans MAGIQUE notre proposition se trouve en amont des préoccupations applicatives et des modèles comportementaux particuliers.

A la différence, dans CoCoA nous nous intéressons à un type d'applications ciblé : les simulations centrées individus par agents situés. Cette fois nous proposons un modèle d'agent particulier : le moteur comportemental des agents animés est fourni. Le travail du concepteur de la simulation consiste ici en la définition des interactions et des propriétés de ces agents. La démarche est ici plus orientée intelligence artificielle que dans MAGIQUE. Pour la mise en œuvre de nos simulations, nous avons nécessairement été confrontés à de nombreux problèmes d'IA qui, sans qu'ils soient les sujets centraux du projet, n'en ont pas moins dû être abordés. Il s'agit notamment de la représentation et de la manipulation de la connaissance et de la planification.

Une autre différence concerne les aspects multi-agents. Avec MAGIQUE nous nous sommes intéressés aux aspects organisationnels en proposant une structure hiérarchique du système multi-agent dotée d'un mécanisme de délégation de réalisation de services. Dans CoCoA, les aspects multi-agents se traduisent dans un premier temps essentiellement par une cohabitation des agents qui évoluent en concurrence dans le même environnement. Dans un second temps, nous nous sommes également intéressés aux comportements d'équipes dirigées ou non par un leader. Ainsi dans MAGIQUE le multi-agent est abordé du point de vue organisationnel alors que le point de vue est plus comportemental dans CoCoA.

Enfin, d'autres différences existent entre les problématiques de ces projets : dans CoCoA les agents sont situés ce qui n'est a priori pas le contexte des applications développées avec MAGIQUE. Cela implique également que la notion d'environnement doive être explicitement définie dans CoCoA alors qu'elle est implicite avec MAGIQUE. Ensuite, MAGIQUE vise des applications distribuées sur le réseau alors que les simulations construites avec CoCoA sont destinées à tourner "en local". Le parallélisme de l'exécution des différents agents est donc géré par du multi-threading dans MAGIQUE alors qu'il n'est que simulé dans CoCoA.

1.4 Conclusion

Ainsi à travers ces deux projets principaux, un large éventail des facettes du domaine du multi-agent a été abordé et étudié. La philosophie de base commune appliquée dans chacun des projets est de considérer que, plus que les agents, ce sont les capacités qu'ils portent qui sont importantes. Notre principe de base est donc de construire ces agents en leur affectant des compétences à partir d'une base commune. Dans le cadre de MAGIQUE, il s'agit de partir d'un noyau minimal élémentaire et de l'enrichir dynamiquement. Dans CoCoA, le modèle comportemental fourni est alimenté par l'attribution des interactions que l'agent peut effectuer. Dans le premier cas, les agents évoluent au sein d'une organisation multi-agent à la base hiérarchique permettant de construire des applications distribuées. Dans le second, les agents sont situés dans un

environnement et permettent la conception de simulations centrées individus où des comportements complexes peuvent être mis en œuvre.

Pour moi, la trame principale de ces recherches est la promotion d'une conception par agents favorisant une vision incrémentale de la construction des agents et plaçant au centre la notion de compétences. Cette approche est symbolisée par notre modèle d'agent minimal d'une part et l'approche orientée interaction des simulations d'autre part. Ce point de vue est complété par une réflexion sur les notions d'organisation multi-agent dans le cadre de MAGIQUE, et par une étude de la modélisation de comportements par agents situés dans le cadre de CoCoA.

A chaque fois, nous nous sommes attachés à mettre en œuvre ces aspects conceptuels de manière concrète : l'API MAGIQUE dans un cas et un environnement de conception de simulations dans l'autre. Ces réalisations nous ont permis de donner corps à nos idées et d'évaluer leur faisabilité et leur pertinence.

Actuellement, c'est au projet CoCoA que je consacre mes activités de recherche. Le travail en cours concerne l'amélioration du modèle comportemental de l'agent ainsi que les stratégies d'équipe. Un des objectifs à court terme est également la réalisation d'une simulation d'envergure valorisant ce modèle comportemental.

Bibliographie

Acroute, J.. 2005. "Apport réactif aux comportements cognitifs de la plateforme de simulation cocoA," Master's thesis, Master Recherche mention Informatique de l'Université de Lille 1, co-dirigé avec le Professeur Philippe Mathieu.

Devienne, P.; P. Lebègue; A. Parrain; and J.-C. R. J. Würtz. 1996. "Smallest Horn clause programs," *Journal of Logic Programming*, 27(3).

Devienne, P.; P. Lebègue; and J.-C. Routier. 1992a. "The halting problem of one binary Horn clause is undecidable," in *proceedings of "Workshop on termination" à l'occasion de JICSLP'92*, 5–14.

Devienne, P.; P. Lebègue; and J.-C. Routier. 1992b. "Weighted Sytems of Equations revisited," in *proceedings of Workshop on Static Aanalysis WSA'92*, 163–173.

Devienne, P.; P. Lebègue; and J.-C. Routier. 1993a. "The emptiness problem of one binary Horn clause is undecidable," in *Proceedings of 1993 International Symposium on Logic Programming, ILPS'93*, ed. by D. Miller, 250–265. MIT Press.

Devienne, P.; P. Lebègue; and J.-C. Routier. 1993b. "The halting problem of one binary Horn clause is undecidable," in *Proceedings of STACS'93*, ed. by P. Enjalbert, A. Finkel, and K. Wagner, no. 665 in LNCS, 48–57. Springer-Verlag.

Devienne, P.; J.-C. R. Patrick Lebègue; and J. Würtz. 1994. "One Binary Horn Clause is Enough," in *Proceedings of STACS'94*, ed. by P. Enjalbert, E. Mayr, and K. Wagner, no. 775 in LNCS, 21–32. Springer-Verlag.

Devigne, D.. 2003. "Simulation de comportements pour agents rationnels situés et étude du GraphPlan," Master's thesis, DEA d'Informatique de l'Université de Lille 1, co-dirigé avec le Professeur Philippe Mathieu.

Devigne, D.. 2003-en cours. “Modélisation de comportement d’équipes en environnement Multi-Agents situés,” Ph.D. thesis, Université des Sciences et Technologies de Lille, co-dirigée avec le Professeur Philippe Mathieu.

Devigne, D.; P. Mathieu; and J.-C. Routier. 2004. “Planning for Spatially Situated Agents,” in *Proceedings of IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT’04)*, ed. by I. Press, 385–388.

Devigne, D.; P. Mathieu; and J.-C. Routier. 2005a. “Gestion d’équipes et autonomie des agents,” in *actes des JFSMA2005*, ed. by Cépaduès.

Devigne, D.; P. Mathieu; and J.-C. Routier. 2005b. “Interaction-Based Approach For Game Agents,” in *Proceedings of ECMS/SCS/IEEE 19th European Conference on Modelling and Simulation. ECMS 2005*, ed. by Y. Merkuryev, R. Zobel, and E. Kerckhoffs, 7056714.

Devigne, D.; P. Mathieu; and J.-C. Routier. 2005c. “Teams of cognitive agents with leader: how to let them some autonomy,” in *Proceedings of IEEE Symposium on Computational Intelligence Games. CIG’05*, ed. by G. Kendall, and S. Lucas, 256–262.

Mathieu, P.; S. Picault; and J.-C. Routier. 2003. “Simulation de comportements pour agents rationnels situés,” in *Actes des Secondes Journées Francophones sur les Modèles Formels de l’Interaction, MFI’03*, ed. by A. Herzig, B. Chaib-draa, and P. Mathieu, 277–282.

Mathieu, P.; S. Picault; and J.-C. Routier. 2005. “Les agents intelligents,” *Pour La Science*, (332), 44–52.

Mathieu, P.; J. Routier; and Y. Secq. 2002a. “Dynamic Organization of Multi-Agent Systems,” in *Proceedings of the AAMAS’02 Conference*, 451–452.

Mathieu, P.; and J.-C. Routier. 2000-2001. “Tutoriel de Magique,” Equipe SMAC - LIFL.

Mathieu, P.; and J.-C. Routier. 2001. “Une contribution du multi-agent aux applications de travail coopératif,” *TSI Hermès Science Publication. Réseaux et Systèmes Répartis. Calculateurs Parallèles.*, 13. Numéro spécial télé-applications, 207–226.

Mathieu, P.; and J.-C. Routier. 2002. “A Multi-Agent Approach to Co-operative Work,” in *Proceedings of CADUI’02*, ed. by C. Kolski, and J. Vanderdonckt, 367–380. Kluwer Academics Press.

Mathieu, P.; J.-C. Routier; and Y. Secq. 2001. “Dynamic Skill Learning: A Support to Agent Evolution,” in *Proceedings of AISB’01*, 25–32.

Mathieu, P.; J.-C. Routier; and Y. Secq. 2002b. “Principles for dynamic multi-agent organizations,” in *Proceedings of 5th Pacific Rim International Workshop on Multi-Agents. PRICAI2002-PRIMA2002*, ed. by K. Kuwabara, and J. Lee, vol. 2413 of *LNAI*, 109–122. Springer-Verlag.

Mathieu, P.; J.-C. Routier; and Y. Secq. 2002c. “RAGE: An agent framework for easy distributed computing,” in *Proceedings of AISB’02*, 20–24.

Mathieu, P.; J.-C. Routier; and Y. Secq. 2002d. “Using agents to build a distributed calculus framework,” *The Interdisciplinary Journal of Artificial Intelligence and the Simulation of Behaviour*, 1(2), 197–208.

Mathieu, P.; J.-C. Routier; and Y. Secq. 2003a. "Bridging the gap between semantic and pragmatic," in *Proceedings of The 2003 International Conference on Information and Knowledge Engineering, IKE'03*, ed. by H. Arabnia, vol. I, 308–314.

Mathieu, P.; J.-C. Routier; and Y. Secq. 2003b. "RIO: Rôles, Interactions et Organisations," in *Actes des Secondes Journées Francophones sur les Modèles Formels de l'Interaction, MFI'03*, ed. by A. Herzig, B. Chaib-draa, and P. Mathieu, 179–188.

Mathieu, P.; J.-C. Routier; and Y. Secq. 2003c. "Runnable specifications of interactions protocols for open multi-agent systems," in *Proceedings of the 2003 International Conference on Information and Knowledge Engineering, IKE'03*, ed. by N. Goharian, vol. II, 431–437.

Mathieu, P.; J.-C. Routier; and Y. Secq. 2003d. "RIO : Roles, Interactions and Organizations," in *Proceedings of the 3rd International/Central and Eastern European Conference on Multi-Agent Systems, CEEMAS 2003*, ed. by V. Marik, J. Müller, and M. Pechoucek, 147–157.

Mathieu, P.; J.-C. Routier; and Y. Secq. 2003e. "Towards a Pragmatic Methodology for Open Multi-agent Systems," in *Proceedings of 14th International Symposium on Methodologies for Intelligent Systems, ISMIS 2003*, ed. by N. Zhong, Z. W. Raś, S. Tsumoto, and E. Suzuki, no. 2871 in LNAI, 206–210. Springer-Verlag.

Mathieu, P.; J.-C. Routier; and P. Urro. 2001. "Un modèle de simulation agent basé sur les interactions," in *Actes des Secondes Journées Francophones sur les Modèles Formels de l'Interaction, MFI'01*.

Routier, J.-C.. 1994. "Terminaison, Satisfiabilité et Pouvoir Calculatoire d'une Clause de Horn Binaire," Ph.D. thesis, Université de Lille 1.

Routier, J.-C.; and E. Wegrzynowski. 1997 - 2003 (2nde édition). *Débuter la Programmation avec* SCHEME. International Thomson Publishing France.

Secq, Y.. 1999. "Notion de service et d'échange de services dans les systèmes multi-agents," Master's thesis, DEA d'Informatique de l'Université de Lille 1, co-dirigé avec le Professeur Philippe Mathieu.

Secq, Y.. 2 décembre 2003. "RIO : Rôles, Interactions et Organisations, une méthodologie pour les systèmes multi-agents ouverts," Ph.D. thesis, Université des Sciences et Technologies de Lille, co-dirigée avec le Professeur Philippe Mathieu.

Tessier, P.. 2002. "Planification et exécution dans un environnement non monotone," Master's thesis, DEA d'Informatique de l'Université de Lille 1, co-dirigé avec Professeur Philippe Mathieu.

Chapitre 2

De MAGIQUE à RIO

Le premier projet auquel j'ai contribué au sein de l'équipe SMAC est le projet MAGIQUE. Le point de départ de ce projet est la proposition par Philippe Mathieu d'une organisation hiérarchique de systèmes multi-agents, MAGIQUE signifiant "Multi AGent hiérarchIQUE" (Bensaid and Mathieu 1997, Bensaid and Mathieu 1997b, Bensaid 1999). Ce projet a évolué pour proposer un modèle d'agent minimal, développer les aspects dynamiques des organisations multi-agents et, enfin, amener à la démarche méthodologique RIO basée sur les notions de rôles, d'interactions et d'organisation.

Ces recherches rejoignent les travaux sur les modèles organisationnels, les plate-formes multi-agents et les méthodes de développement d'applications à l'aide de systèmes multi-agents. Cette thématique correspondait notamment aux préoccupations du groupe ASA, pour "Architecture des Systèmes multi-Agents", du GdR I3. Plus largement le projet MAGIQUE s'inscrit dans le thème "Agent Oriented Software Engineering" (Jennings and Wooldridge 2000, Occello and Koning 2000, Petrie 2001) puisque l'objectif principal du projet est de fournir un contexte facilitant le développement d'applications grâce aux agents.

Les notions développées dans le projet MAGIQUE s'intègrent ainsi à différentes autres études :

- celles menées sur les organisations : à base de groupes (Ferber and Gutknecht 1998a), holonique (Gerber, Siekmann, and Vierke 1999, Adam, Mandiau, and Kolski 2001) ou le pair à pair par exemple.
- celles menées sur les plate-formes : Madkit (Ferber, Gutknecht, and Michel 2000, J.Ferber, Gutknecht, and Michel 2000) du LIRMM, Volcano (Ricordel and Demazeau 2002), Dima (Guessoum 1996),
- celles sur les méthodes : Voyelles (Ricordel 2001, da Silva and Demazeau 2002), Gaia (Wooldridge, Jennings, and Kinny 2000).

Les paragraphes suivants vont tenter de synthétiser les différents résultats de nos recherches sans nécessairement respecter la progression temporelle. Ainsi si le modèle organisationnel a été le moteur fondateur de MAGIQUE, je présenterai en premier lieu le modèle d'agent minimal, puis l'organisation hiérarchique et ses apports pour enfin m'intéresser à leur mise en œuvre dans l'API MAGIQUE. Je finirai par une brève présentation de RIO.

2.1 Le modèle d'agent minimal

Le caractère "agent" d'une application se situe le plus souvent dans l'esprit du programmeur (Shoham 1993, Travers 1996). De ce fait, de nombreuses définitions du terme *agent* sont apparues (Franklin and Grasser 1996), et il n'est pas difficile de regrouper un grand nombre de définitions différentes. L'intersection entre toutes ces définitions n'est le plus souvent pas vide : elles diffèrent, parfois très légèrement, à cause de quelques fonctionnalités considérées comme basiques pour chacun des modèles. Partant de ce constat, nous avons essayé de proposer une base sur laquelle pourraient s'appuyer toutes les autres définitions. Notre proposition consiste en la définition d'un *agent atomique* susceptible d'évoluer dans différentes directions pour correspondre aux différentes notions introduites par chacun.

La notion centrale de notre proposition est celle de *compétence*. Elle correspond à un ensemble cohérent de capacités qui peuvent être attribuées à un agent. Le principe est de partir d'un agent atomique et de lui *enseigner* les compétences nécessaires à l'obtention de l'agent recherché. Le résultat dépend des compétences enseignées et il est donc possible d'obtenir différents types d'agents. De plus, l'éducation doit pouvoir être réalisée alors que l'agent est actif. Celui-ci peut donc évoluer dynamiquement.

Du point de vue du programmeur, cette approche favorise la réutilisabilité et la modularité, puisqu'une fois qu'une compétence a été développée, elle peut être exploitée dans différents contextes. Dans cette vision, une compétence peut être considérée comme un composant logiciel (pour une autre approche de la construction d'agents à partir de composants, voir par exemple (Horling and Lesser 1998)).

Ces travaux ont été publié dans (Routier, Mathieu, and Secq 2001).

2.1.1 Définitions

Dans la mesure où un agent est *quelqu'un qui agit*, à partir du moment où l'on convient qu'une *compétence* désigne un "ensemble cohérent de capacités", la définition suivante devrait pouvoir être relativement consensuelle :

Définition 1 *Un* agent *est une entité douée de compétences.*

Toute propriété communément attachée à la notion d'agent – telle que proactivité, interactivité, intelligence, etc. – peut en effet s'exprimer en terme de compétences. Il semble donc raisonnable de dire que toutes les définitions d'agents, comme celle citées dans (Franklin and Grasser 1996), peuvent être obtenues à partir de celle-ci : les différences entre deux définitions d'agent proviennent en effet des fonctionnalités de base exigées des agents, c'est-à-dire de leurs compétences. Ainsi un agent dont le cycle de décision s'appuie sur un réseau de Pétri est un agent qui dispose d'une compétence d'interprétation de ce réseau ; un autre qui peut crypter ses messages est un agent doué d'une compétence de codage/décodage, etc. En se plaçant plus au niveau des concepts, les notions de rôles et de groupes, qui sont au centre du modèle Aalaadin (Ferber and Gutknecht 1998), peuvent également se traduire en termes de compétences et donc ce modèle pourrait être décrit en de tels termes. Cependant, pour cette même raison, la définition est probablement trop vague puisqu'elle permet trop de liberté d'interprétation selon les compétences attachées à l'agent. C'est pourquoi nous allons la préciser en fixant un ensemble minimal de compétences.

Nous affirmons que simplement deux compétences initiales sont nécessaires et suffisantes pour définir cet *agent atomique* à partir duquel toutes les autres définitions d'agent peuvent

être établies. Ces compétences sont : pour la première, une compétence qui permet à l'agent d'apprendre de nouvelles compétences, et pour la seconde, une compétence de communication (avec les autres agents – qui pourraient être humains ou logiciels).

Ces compétences sont en effet *nécessaires*. Sans la "compétence d'acquisition", un tel agent ne serait qu'une coquille vide incapable de faire quoique ce soit. Sans la "compétence de communication", un agent est isolé du "reste du monde" et perd de ce fait tout intérêt. De plus la communication est nécessaire à l'acquisition de nouvelles compétences.

Et elles sont *suffisantes* puisqu'il suffit à un agent d'utiliser sa compétence d'interaction pour entrer en contact avec un agent compétent et d'utiliser sa compétence d'acquisition pour apprendre de nouveaux talents auprès de celui-ci. Ainsi, n'importe quelle capacité peut être "donnée" à un agent par apprentissage auprès d'un "enseignant".

En conséquence nous proposons la définition d'agent suivante :

Définition 2 *Un* agent atomique *est une entité douée de deux compétences : une pour interagir et une pour apprendre de nouvelles compétences. Un* agent *est un agent atomique qui a appris des compétences au travers de communications.*

Nous soutenons que les types d'agents proposés dans les différentes définitions existantes s'intègrent dans cette définition.

Notons que ce ne sont pas les compétences elles-mêmes qui sont importantes mais plutôt les fonctionnalités qu'elles représentent. Ainsi on peut imaginer que la compétence de communication utilisée par un agent évolue au cours de son cycle de vie, parce qu'il en a appris une nouvelle par exemple. Ce qui est important ce n'est pas qu'il ait telle compétence d'interaction particulière, mais plutôt qu'il ait toujours la capacité de communiquer (et il en est de même avec la compétence d'apprentissage).

2.1.2 Compétences

Les compétences constituent la substantifique moelle des agents dans notre vision. Une compétence correspond à un ensemble de fonctionnalités qui peuvent être exploitées par un agent. D'un point de vue plus pragmatique, une compétence doit être vue comme un composant dont l'interface publique désigne les capacités qu'un autre agent peut exploiter.

La granularité et le degré de complexité d'une compétence ne peuvent pas être définitivement énoncés. Les capacités d'analyser un message XML ou d'additionner deux entiers peuvent chacune représenter une compétence bien que leurs complexités soient très probablement considérées comme se situant à des niveaux différents. De plus, savoir si il faut grouper dans une seule compétence les quatre opérations arithmétiques de base (addition, soustraction, multiplication et division) ou les séparer en quatre compétences ne peut pas être clairement établi. Cependant, il devrait être possible d'avoir un assentiment général sur le fait que les capacités d'analyse XML et l'addition doivent se situer dans des compétences différentes. Une compétence doit en effet représenter un ensemble *cohérent* de capacités.

Convenir qu'à une compétence doit correspondre une et une seule capacité (ou réciproquement) pourrait sembler raisonnable, mais la réponse n'est pas aussi simple, notamment car il existe des dépendances entre les capacités, et dans tous les cas cela risque de ne pas résister à la réalité des programmeurs... Les problèmes qui apparaissent ici sont les mêmes que ceux habituellement (et universellement) rencontrés en génie logiciel, et en conception objet en particulier, concernant la décomposition en objets.

2.1.3 Avantages

Ce paradigme offrant une construction dynamique d'agents à partir de compétences procure de nombreux avantages. D'abord, l'autonomie des agents est potentiellement accrue puisqu'ils peuvent acquérir les compétences qui leur manquent. L'agent peut de même faire évoluer et améliorer les compétences dont il est déjà doté. Remarquons que d'un point de vue opérationnel cette évolutivité dynamique apporte une souplesse supplémentaire. Quand une compétence d'un agent doit être changée (a priori pour l'améliorer), il n'est plus nécessaire d'accomplir le cycle classique (et pénible) : "l'arrêter, modifier le source, compiler et redémarrer". La nouvelle compétence peut être dynamiquement enseignée à l'agent. Cela peut être particulièrement important pour un agent dont la durée de vie est longue et qui est dédié à un rôle qui ne peut tolérer la moindre interruption.

Ensuite, le système multi-agent tire profit de ce principe. Il y gagne en robustesse puisqu'un agent détenteur d'une compétence critique qui doit quitter le système peut céder à un autre agent cette compétence et ainsi garantir la pérennité et la cohérence du système. L'efficacité est également potentiellement améliorée puisqu'un agent submergé par des requêtes pour exploiter une de ses compétences peut choisir de l'enseigner à d'autres agents (qu'il peut éventuellement créer et éduquer dans ce but spécifique) afin d'alléger sa charge.

Avec ce principe d'évolution dynamique d'un agent, il n'est plus possible d'utiliser le terme de "classe" d'agents. Même si pour différents agents vous partez d'un base commune, dans la mesure où ils peuvent, et vont probablement, recevoir une éducation différente due à leurs "expériences" individuelles, ils vont bientôt diverger. Il sera de ce fait impossible de les considérer comme appartenant à une même "classe". Cette notion n'a définitivement plus de sens dans ce contexte. Cela constitue une différence forte entre une telle programmation orientée agents et la programmation orientée objets.

2.2 Le modèle d'organisation hiérarchique

L'origine du projet MAGIQUE est sa proposition d'un modèle organisationnel pour systèmes multi-agents qui s'appuie sur la notion de hiérarchie évolutive (Bensaid and Mathieu 1995, Bensaid and Mathieu 1997) et d'"appel à la cantonnade". Les agents mis en œuvre correspondent au modèle présenté dans la section précédente.

Dans MAGIQUE une société d'agents est définie à la base comme une hiérarchie, c'est-à-dire un arbre dont chaque nœud est un agent et dont les branches représentent les liens d'accointances par défaut. MAGIQUE correspond donc à la notion d'"acquaintance model" telle que donnée dans (Wooldridge, Jennings, and Kinny 2000) ou (Kinny, Georgeff, and Rao 1996). Cependant cette notion de "modèle d'accointances" ne fait que préciser que les agents interagissent entre eux et qu'ils doivent pour cela utiliser des chemins de communication. La notion d'organisation doit aller plus loin en fournissant un modèle de "construction" de ce réseau d'accointances. C'est pourquoi MAGIQUE accompagne la proposition de la structure hiérarchique d'un mécanisme par défaut de délégation d'une requête d'exécution de compétences : "l'appel à la cantonnade". Le principe en est le suivant, quand un agent souhaite utiliser une compétence :

- l'agent "possède" la compétence, il l'"invoque" directement,
- l'agent ne "possède" pas la compétence, plusieurs cas de figure possibles :

- il a une accointance particulière pour cette compétence, il lui demande alors de la réaliser pour lui,
- sinon, il est racine d'une hiérarchie et un membre de sa hiérarchie possède cette compétence, il transmet (récursivement *via* la hiérarchie) la délégation de réalisation de cette compétence à qui de droit,
- sinon, il demande à son supérieur hiérarchique de trouver quelqu'un de compétent pour lui et celui-ci réapplique ce même mécanisme récursivement.

Du point de vue de la programmation, l'invocation de compétence est très facile à réaliser. Là où en programmation objet vous écrivez :

```
object.ability(arg...);
```

pour faire un appel à une méthode `ability`, vous devez maintenant écrire :

```
perform("ability",arg...);
```

Cela a pour effet que la compétence nommée "`ability`" sera "invoquée" (sans avoir à savoir par qui[1]).

La primitive `perform` est dédiée à l'invocation de compétence pour lesquelles on n'attend pas de réponse. Il existe principalement trois autre primitives : `ask` quand une réponse asynchrone est désirée, `askNow` pour une réponse immédiate et `concurrentAsk` pour une invocation concurrente.

L'avantage de cette délégation du point de vue du programmeur, en comparaison des appels nominatifs, est qu'il n'a pas besoin de connaître explicitement les agents qui seront présents dans l'application au moment du codage. Les références se situent en effet au niveau des compétences. Il suffit pour le concepteur de savoir que la compétence est présente dans le système multi-agents, sans nécessairement connaître l'agent compétent. Le code produit gagne alors en réutilisabilité. L'application multi-agent gagne, quant à elle, en robustesse car l'agent réalisateur d'une compétence n'étant pas nécessairement prédéfini, il peut éventuellement varier au fil du temps, si un agent devient indisponible ou surchargé par exemple.

Ainsi quand un agent souhaite exploiter une compétence, peu importe qu'il la possède ou non. Dans les deux cas, la manière d'invoquer la compétence est la même. Si la réalisation de la compétence doit être déléguée à un autre, cela est fait de manière transparente pour lui, même si il n'a pas de lien d'accointance direct pour celle-ci. Un tel mécanisme facilite le développement en découplant la compétence de tout agent ou système multi-agent. De plus il favorise la fiabilité du système en améliorant la tolérance aux pannes.

Un autre point fondamental dans MAGIQUE est la possibilité pour un système multi-agents d'évoluer dynamiquement. En effet avec seulement les communications hiérarchiques, le modèle serait probablement trop rigide. C'est pourquoi MAGIQUE offre la possibilité de créer des liens directs (i.e. en dehors de la hiérarchie) entre deux agents. Nous les appelons *liens d'accointances* (par opposition aux liens *hiérarchiques* même si ceux-ci désignent également des accointances).

[1] Bien évidemment, MAGIQUE offre également la possibilité de préciser un destinataire si besoin est, ceci est d'ailleurs indispensable lorsque deux agents n'ont pas de relation d'accointance à travers une hiérarchie mais sont en relation directe.

La décision de la création de tels liens dépend de la politique de l'agent. L'idée visée est cependant la suivante : après quelques temps d'évolution du système multi-agents, si il apparaît des requêtes privilégiées (fréquentes) entre deux agents pour une compétence, la décision peut être prise de créer dynamiquent un lien d'accointance entre ces deux agents pour la compétence concernée. L'intérêt est évidemment de diminuer les communications et de mettre en évidence des liens "naturels" d'interaction entre les agents. Ces liens correspondent aux communications *horizontales*. Une stratégie de création de liens d'accointance peut donc être spécifiée au niveau des agents. Cette approche reprend le mode de fonctionnement des interactions au sein d'une entreprise.

Ainsi, dans MAGIQUE, l'organisation hiérarchique n'est proposée que comme support par défaut. Cette structure est destinée à évoluer en accord avec le comportement dynamique du système multi-agents en favorisant les relations les plus fréquentes. Il en résulte, qu'après un certain temps, le système multi-agents devrait plus ressembler à un graphe. L'intérêt est que, la structuration étant dynamique et auto-adaptative, le concepteur d'un système multi-agents peut se reposer en partie sur ce mécanisme lors de la conception de l'organisation de son système multi-agents.

La dynamicité dans MAGIQUE opère à plusieurs niveaux. Elle est d'abord *individuelle* puisqu'un agent peut acquérir ou oublier des compétences. Dans MAGIQUE, cela se traduit par un échange effectif de compétences entre agents, éventuellement distants, à l'initiative des agents eux-mêmes. La dynamicité est également *relationnelle* puisque des liens d'accointances peuvent être créés lorsque des relations favorisées apparaissent entre deux agents de la hiérarchie. Cela a pour effet de supprimer les communications récurrentes le long de l'arborescence puisque alors la communication entre les agents devient directe. Couplée avec le mécanisme de délégation, cette création dynamique de relations de communication contribue à rendre *non déterministe* le fonctionnement d'une application multi-agent : deux exécutions successives ne produiront pas nécessairement la même structure d'accointances et donc de communications et donc les compétences ne seront pas réalisées nécessairement par les mêmes agents. La dynamicité est enfin *organisationnelle* ou *architecturale* puisque des agents peuvent être créés ou détruits afin d'adapter le système multi-agents à certaines contraintes.

Pour caractériser cette dynamicité nous avons identifié trois principes simples qui peuvent être utilisés pour améliorer le comportement global et qui entraînent une organisation dynamique de la structure sociale :

1. *avoir un bon carnet d'adresses*,
2. *partager la connaissance*,
3. *recruter de nouveaux collaborateurs compétents*.

Afin de pouvoir appliquer ces principes les agents doivent d'abord avoir la possibliité de créer dynamiquement de nouveaux liens d'accointances afin d'auto-adapter l'organisation (Ghanea-Hercock 2000). Cependant, ils 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. Ensuite il faut que les agents puissent apprendre de nouvelles

compétences auprès des agents (et donc les agents doivent être capables d'enseigner aux autres, voir (Clement 2000)). Un mécanisme qui permet cela doit être fourni tout en tenant compte de l'aspect distribué. Enfin, les agents doivent pouvoir créer dynamiquement de nouveaux agents, et en utilisant la capacité d'acquérir des compétences, ces agents peuvent être adaptés aux besoins, il suffit de leur inculquer les bonnes compétences.

Ces différents critères sont offerts par les propriétés de dynamicité offertes par les modèles d'agent et d'organisation de MAGIQUE. Nous avons mené différentes expérimentations concernant ces trois principes d'évolution dynamique des systèmes multi-agents. Ce travail a été publié dans (Mathieu, Routier, and Secq 2002).

2.3 API et applications

Les deux sections précédentes ont présenté les principaux concepts et principes mis en œuvre dans MAGIQUE : des agents construits par acquisition dynamique de compétences s'intégrant dans une organisation par hiérarchique offrant un mécanisme de délégation automatique de services et pouvant évoluer dynamiquement.

Comme je l'ai indiqué dans la première partie, nous partageons au sein de l'équipe SMAC le souci de mettre en pratique les concepts que nous proposons. MAGIQUE a donc été concrétisé sous le forme d'une API Java dont l'objectif est de permettre le développement d'applications réparties à base d'agents. Nous avons voulu cette API comme un framework général sans orientation applicative. Elle fournit donc les éléments de base pour des applications agents, les aspects particuliers étant à développer comme des bibliothèques constituées de compétences. En plus de l'implémentation de nos concepts, une de nos préoccupations lors de l'écriture de cette API a été de la rendre aussi facile à appréhender que possible pour quelqu'un ayant une connaissance raisonnable du langage Java. Par différentes expériences avec des étudiants et le retour de différents utilisateurs, il semblerait que cet objectif ait été raisonnablement atteint.

2.3.1 Mise en place du modèle

Après avoir défini les notions de compétence et de message, l'étape suivante a consisté en la création de notre agent atomique (classe `AtomicAgent`) doté des deux compétences minimales lui permettant d'acquérir des nouvelles compétences et de communiquer (cf. figure 2.1).

La notion d'agent MAGIQUE (classe `Agent`) a ensuite été mise en place conformément au modèle. C'est-à-dire que nous avons développé les différentes compétences liées au modèle organisationnel MAGIQUE. Il s'agit essentiellement des compétences permettant la mise en œuvre de la hiérarchie (connexion à un supérieur, notion d'équipes, etc.) et de la gestion de la délégation de services. Ces compétences sont ensuite enseignées dynamiquement à l'agent atomique :

```
public class Agent extends AtomicAgent {
...
   protected void initBasicSkills() throws SkillAlreadyAcquiredException {
      addSkill(new fr.lifl.magique.skill.system.DisplaySkill());
      addSkill(new fr.lifl.magique.skill.system.AddSkillSkill(this));
      addSkill(new fr.lifl.magique.skill.system.LearnSkill(this));
      addSkill(new fr.lifl.magique.skill.system.ConnectionSkill(this));
      addSkill(new fr.lifl.magique.skill.magique.BossTeamSkill(this));
      addSkill(new fr.lifl.magique.skill.magique.ConnectionToBossSkill(this));
      addSkill(new fr.lifl.magique.skill.magique.KillSkill(this));
   }
```

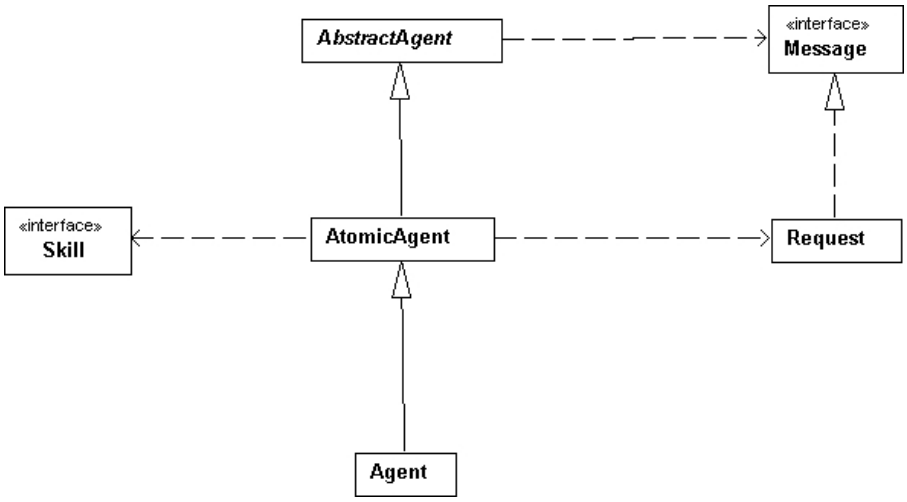

Figure 2.1: Les dépendances entre les principales classes de l'API MAGIQUE. La classe `Agent` correspond aux agents MAGIQUE.

```
}
```

L'extrait de code ci-dessus est directement issu du source de l'API. La méthode `initBasicSkills` est appelée dès la fin de la construction d'une instance de type `Agent` (en fait `AtomicAgent`). Comme on peut le constater (et le comprendre facilement) les compétences propres à l'API MAGIQUE (en gras) sont dynamiquement ajoutées à l'agent via les appels à la méthode `addSkill`. Les agents MAGIQUE sont donc bien obtenus à partir de l'agent atomique par enrichissement dynamique de compétences. L'existence d'une classe spécifique aux agents MAGIQUE ne se justifie en effet que pour offrir des méthodes facilitant le développement. La mise en place de tout autre modèle d'agent se ferait de la même manière, une fois les compétences propres à ces modèles développés il faut les enseigner à l'agent atomique.

Il en est de même des agents impliqués dans une application. Leur construction est simple une fois les compétences développées puisqu'il suffit de lui enseigner celles-ci. Il peut ensuite être intégré à une hiérarchie existante :

```
import fr.lifl.magique.*;
...
   // création d'un agent atomique
   Agent myAgent = createAgent("myName");
   // l'agent apprend dynamiquement des compétences
   myAgent.addSkill("SkillOne");
   myAgent.addSkill("SkillTwo");
   myAgent.addSkill("SkillThree", params);
   // il rejoint une hiérarchie (= SMA)
   myAgent.connectToBoss("bossName@host.dom.XX:4444");
   ...
```

La création de compétence ne cache pas de difficulté particulière. Il s'agit de développer une classe Java dont les méthodes publiques seront les capacités que pourra exploiter l'agent compétent. La signature de la méthode sert de sémantique pour les messages.

```
import fr.lifl.magique.*;
import fr.lifl.magique.skill.*;
...
public class ASkill implements Skill {
   public ASkill() {...}

   // l'agent pourra utiliser <ability>
   public void ability(...)  {
      ...
   }
}
```

S'appuyant sur notre modèle d'agent minimal, la structure que nous proposons permet la construction d'agents participant à des structures organisationnelles différentes de celle de MAGIQUE. René Mandiau et Emmanuel Adam de l'Université de Valenciennces ont ainsi utilisé MAGIQUE et développé les compétences permettant la mise en place d'une organisation holonique (Adam and Mandiau 2005).

2.3.2 Délégation, accointances et échange de compétences

Les primitives d'invocation de compétences précédemment évoquées sont évidemment mises en place: `perform`, `ask`, `askNow`. Elles permettent les invocations de services sont le destinataire est nommé ou non, ce second cas mettant en œuvre le mécanisme de délégation automatique.

Les connexions entre agents sont réalisées par les primitives `connectTo` pour le connexions "directes" et `connectToBoss` pour la mise en place des liens hiérarchiques. Enfin, l'API offre les primitives permettant l'échange dynamique de compétences entre les agents, y compris si ceux-ci sont physiquement distribués. Cet échange ne nécessite pas que soit réalisée une distribution de bibliothèques applicatives sur l'ensemble des machines impliquées dans l'application. Le cas échéant, lors de l'échange d'une compétence entre deux agents, le bytecode nécessaire est automatiquement transféré entre les plate-formes MAGIQUE (cf. Figure 2.2). Ce mécanisme est rendu possible par la présence d'"agents plate-forme" qui sont des agents atomiques auxquels nous avons ajouté les compétences nécessaires. Le déploiement d'applications multi-agents se trouve ainsi simplifié facilitant le travail du développeur.

L'API et son utilisation sont présentées plus en détail dans le tutoriel MAGIQUE (Mathieu and Routier 2000-2001).

2.3.3 IDE

Pour encore faciliter la mise en place de systèmes multi-agents applicatifs, nous avons conçu un environnement de développement et de déploiement (cf. Figure 2.3). Celui-ci permet de constituer les agents en leur assignant leurs compétences, de construire la hiérarchie puis de déployer automatiquement les agents, sans qu'il soit nécessaire d'avoir préalablement déployé du code applicatif sur les machines distantes et ce grâce à l'existence des agents plate-formes et à l'échange dynamique de compétences. Cet environnement est en effet développé à base d'agents atomiques (non spécifiquement MAGIQUE) dotés des compétences propres à cet environnement. Enfin une console d'administration permet d'interagir directement avec les agents.

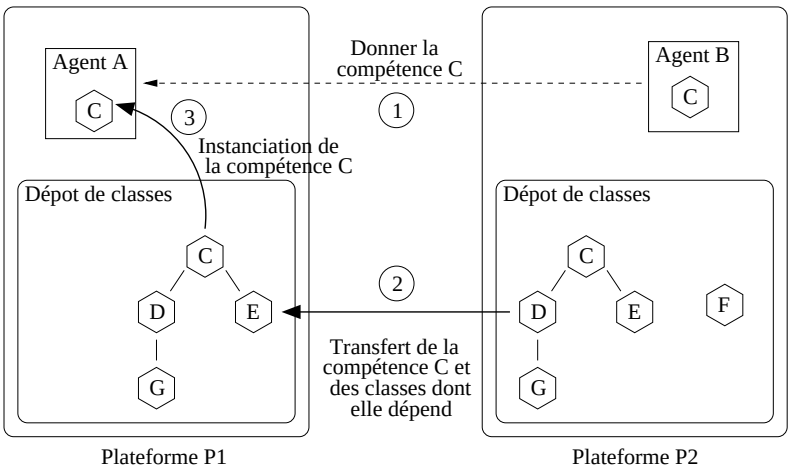

Figure 2.2: Echange dynamique de code entre plate-formes

2.3.4 Exploitation

Les agents proposés par l'API MAGIQUE n'ont aucune capacité applicative particulière. Cependant, avec cette base, le constructeur d'applications se voit dégager des soucis liées à la distribution, la communication entre les agents et, au moins partiellement, à la mise en place de son organisation. Il peut s'appuyer sur la hiérarchie de base et son mécanisme de délégation et laisser le système multi-agents évoluer en utilisant les possibilités de réorganisation dynamique.

Le concepteur peut alors se consacrer à l'essentiel : le développement des aspects applicatifs. Cela correspond à l'écriture de compétences. Le concepteur doit donc déterminer les fonctionnalités nécessaires à son application, les regrouper en ensemble cohérent pour construire les compétences puis affecter ces compétences à des agents MAGIQUE pour les faire évoluer vers les agents applicatifs souhaités. L'attribution de compétences à des agents revient à leur attribuer des rôles : celui induit par la compétence. Les agents sont ensuite intégrés à une hiérarchie. Si le concepteur a tout intérêt à réfléchir à l'organisation logique de celle-ci, il peut néanmoins se reposer partiellement sur le mécanisme de délégation automatique qui garantit qu'une requête d'exploitation atteindra l'agent compétent dès que la compétence est présente dans le système multi-agents,

2.3.5 Applications

Nous avons utilisé les concepts de MAGIQUE et l'API qui les met en œuvre pour le développement de différentes applications. A travers ces expériences nous avons pu vérifier les avantages apportés par le modèle notamment au niveau de la souplesse de conception. Nous avons ainsi pu valider notre approche par compétences. La démarche de conception par agent est toujours la même. D'abord, le travail d'analyse consiste à identifier les compétences mises en œuvre dans l'application et les rôles qu'elles induisent. Ensuite vient le développement de ces compétences. La construction des agents est ensuite aisée puisqu'il suffit de leur "ajouter" les compétences qui leur sont dues. Le travail de distribution sur le réseau et d'échange de messages est pris en charge par les fonctionnalités offertes de base par MAGIQUE.

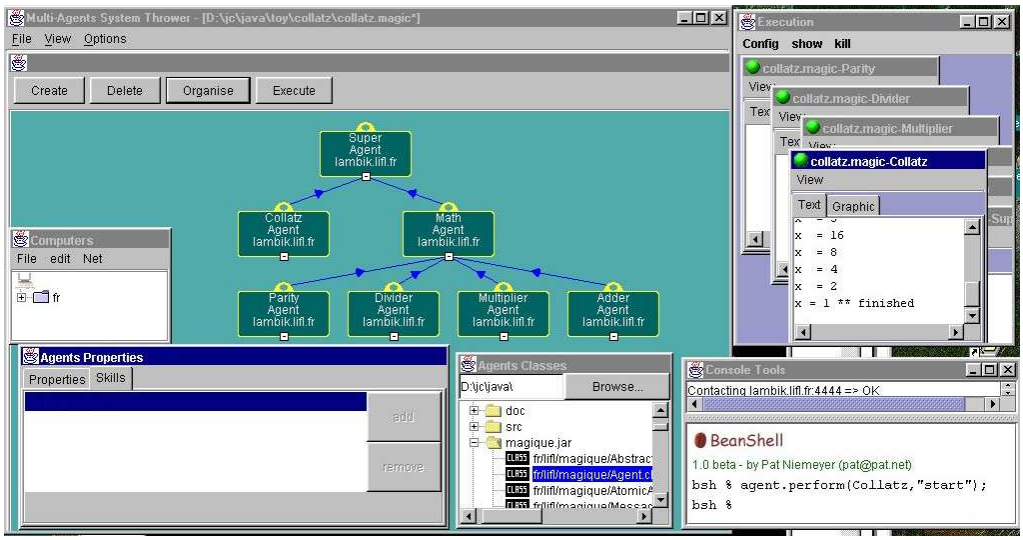

Figure 2.3: Environnement de développement et de déploiement MAGIQUE.

Parmi ces applications, j'en citerai deux en particulier. La première se situe dans le domaine du travail coopératif. Elle fournit un support permettant la tenue d'une conférence entre intervenants physiquements distribués. La seconde s'inscrit dans le calcul distribué. Il s'agit du framework RAGE permettant la définition de tâches de calcul et leur distribution sur un réseau de machines.

L'apport de la première application a été pour nous de tester la conception d'application avec MAGIQUE avec la création d'agents applicatifs par ajout de compétences à partir de l'agent atomique. Mais surtout, cette application offrait un exemple pour l'échange dynamique de compétence entre les agents permettant une évolution dynamique de rôles. La seconde application a été l'occasion à nouveau de valider l'approche par compétences et aussi d'aborder un cas faisant intervenir plus de rôles et une hiérarchie plus complexe. De plus cette application nécessitait des créations dynamiques d'agents en fonction des tâches de calcul à réaliser.

Remarquons que dans les deux cas, la présence d'agents dans l'application n'est pas nécessairement perceptibles par l'utilisateur de l'application. C'est essentiellement l'approche et la conception de l'application qui sont "orientées agents".

Travail coopératif

L'application "diapo-conférence". L'objectif de cette application de travail coopératif est de fournir un support permettant la tenue d'une conférence entre des intervenants physiquement distribués. Chacun des intervenants dispose d'un ensemble de ressources documentaires ("*diapositives*"). Dans une première version de cette application, ces ressources sont décrites par des documents HTML. L'application doit permettre à l'utilisateur de diffuser ces documents auprès des autres. Il ne peut cependant le faire qu'à la seule condition qu'il soit détenteur de la *télécommande* (unique) associée à l'application. Cette télécommande lui permet de parcourir et de diffuser ses ressources. Mais elle tient également lieu de "pointeur" que le conférencier peut utiliser pour

attirer l'attention des autres utilisateurs sur un point précis du document diffusé, ce pointeur étant visible par tous les utilisateurs en temps réel.

On distingue donc deux *rôles* dans cette application : celui de *conférencier* pour celui qui détient la télécommande, et celui d'*auditeur* pour les autres participants. Comme dans une conférence réelle, le conférencier peut à tout moment céder la télécommande à un auditeur et les rôles respectifs tenus par les intervenants évoluent alors dynamiquement. Une attention particulière a été portée au respect de la conformité avec la tenue des conférences réelles.

Les figures 2.4 et 2.5 donnent un aperçu de l'application du point de vue du conférencier et du point de vue d'un auditeur quelconque. On peut distinguer différents outils sur les deux vues : la visionneuse sur la gauche des écrans (on peut remarquer que celle de l'auditeur a la même taille que celle du conférencier), le pointeur de souris que l'on peut apercevoir dans la visionneuse du conférencier est visualisée par un point dans la fenêtre de l'auditeur. D'autres outils comme la télécommande (active chez le conférencier et passive chez l'auditeur), les fenêtres d'équipe et de messagerie, ainsi que l'assistant en haut de chaque écran (il fait une annonce dans la fenêtre du conférencier) sont également visibles.

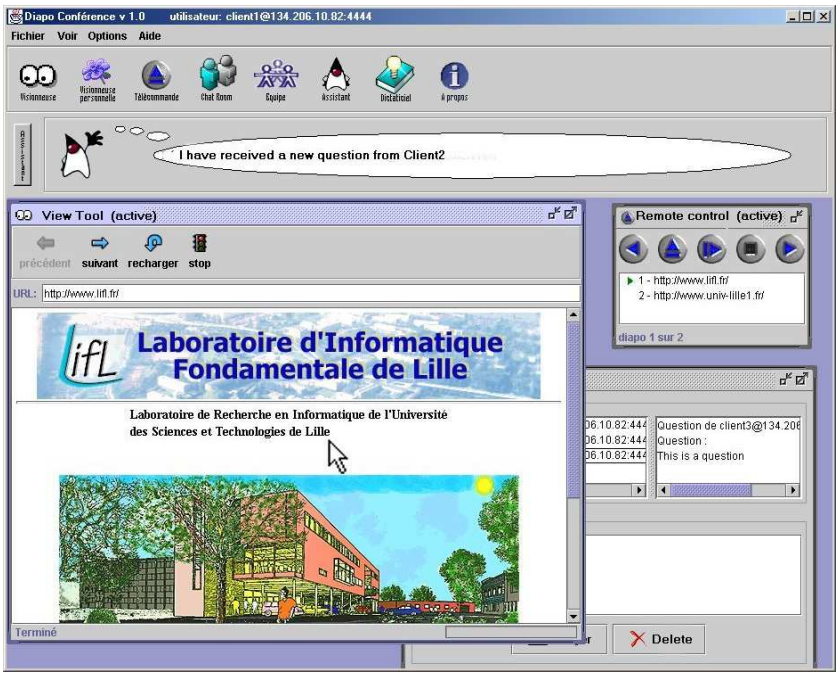

Figure 2.4: L'application du côté *conférencier*. On peut entre autre voir la fenêtre de diffusion des diapositives, l'assistant qui fait part d'un évènement.

Il est possible pour le conférencier de céder cette télécommande à un auditeur et d'induire ainsi un changement de rôle pour chacun des participants. C'est cet aspect qui nous intéressait le plus dans le cadre expérimental offert par cette application. Cet échange fonctionnel se traduit en effet par un transfert effectif de la compétence de gestion de la télécommande entre les agents et induit un changement de rôles des agents impliqués dans l'échange de compétence. Nous reviendrons sur ce point un peu plus loin.

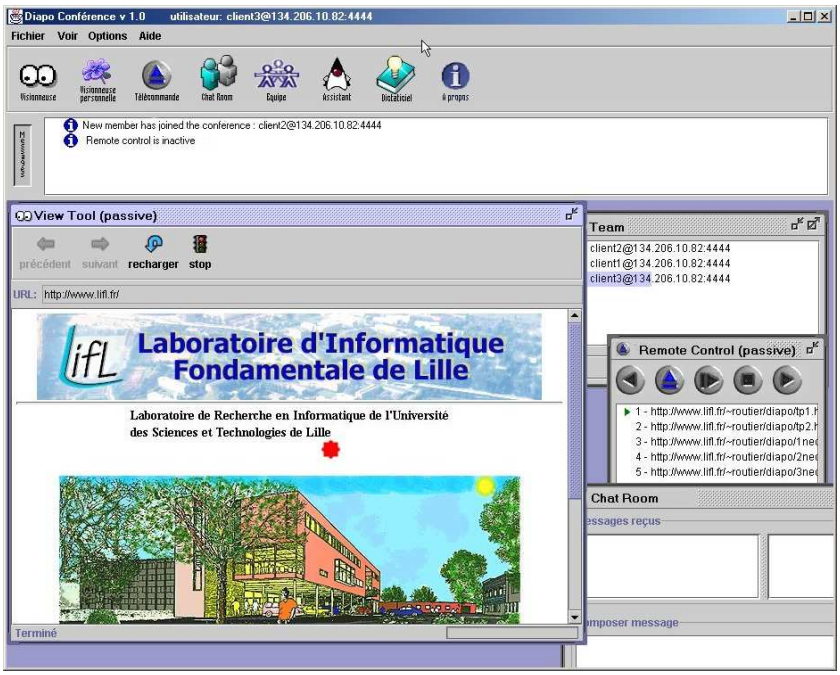

Figure 2.5: L'application du côté *auditeur*, on peut remarquer le pointeur rouge représentant la souris du conférencier et dont il reproduit les déplacements.

Les autres fonctionnalités offertes par l'application sont des outils qui favorisent la coopération et la prise de conscience par chacun des autres intervenants. Citons principalement : la "visualisation" de l'assistance, un système de messagerie et un assistant. Ce dernier a trois fonctions principales : il permet une mise en évidence d'événements, fournit une aide pour contribuer à la coopération et offre une modélisation des autres intervenants en vue de futures collaborations.

Nous allons aborder maintenant rapidement la conception de cette application avec MAGIQUE.

La réalisation avec MAGIQUE Il convient de déterminer les agents intervenants et les compétences de chacun, puis de définir l'organisation du système et les interactions possibles.

Nous avons déjà clairement identifié deux de ces rôles : ceux de *conférencier* et d'*auditeur*. Il existe cependant un troisième rôle qui est celui du *coordinateur*. Il peut être vu comme le représentant de l'*environnement* et constitue le support de l'interaction et concrétise par son existence la cohabitation de l'ensemble des participants à la conférence.

Les compétences correspondant aux différentes fonctionnalités brièvement décrites précédemment doivent ensuite être développées. La compétence de gestion de la *télécommande* est la plus intéressante car c'est elle qui crée la différence entre les deux rôles *conférencier* et *auditeur*. L'évolution dynamique des rôles en cours de session (de *conférencier* à *auditeur* et réciproquement) implique une évolution dynamique de la compétence connue par les agents. Il est en effet nécessaire d'accorder au nouveau conférencier les fonctions pour mener la conférence et de les retirer à l'ancien conférencier. L'évolution dynamique des rôles *conférencier*$\longleftrightarrow$*auditeur* est facilement prise en compte par MAGIQUE. Notre plate-forme a en effet l'avantage de permet-

tre l'évolution dynamique des compétences des agents par échanges entre eux. Les agents apprennent et oublient *effectivement* des compétences. Ces échanges de compétences sont réels, indépendamment de toute hypothèse sur le code de ces compétences et de la distribution sur le réseau des agents. Le *conférencier* peut donc, comme cela se passe lors du passage de micro dans une conférence réelle, donner la compétence de gestion de la télécommande au nouveau *conférencier* et de ce fait la *perdre* (cf. figure 2.6). Il ne s'agit pas ici de la simple modification de la valeur d'un attribut quelconque, mais d'une mutation concrète de l'agent et du rôle qu'il tient et donc en quelque sorte de ses droits dans l'application.

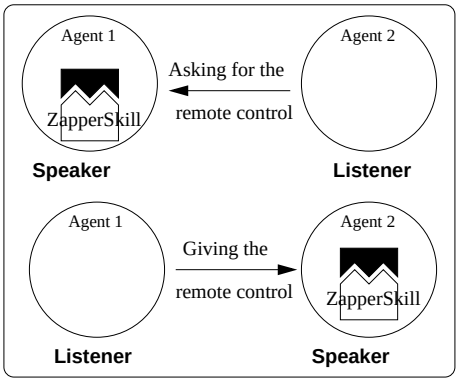

Figure 2.6: Evolution des rôles par échange de la compétence "Télécommande". Il s'agit d'un échange effectif de la télécommande. L'agent conférencier donne la compétence Télécommande à l'auditeur et n'a plus possibilité d'accéder aux services qu'elle offre.

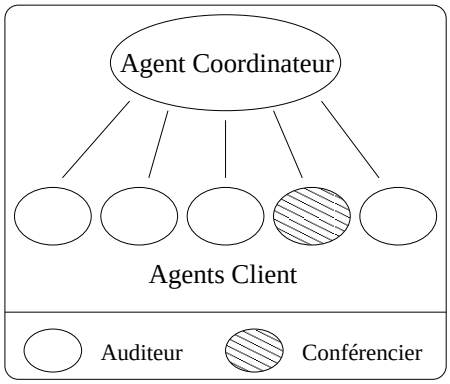

Figure 2.7: L'organisation hiérarchique du système multi-agents

Il faut bien sûr ensuite faire le choix de la structure organisationnelle pour le système multi-agents. Avec MAGIQUE cette structure est par défaut hiérarchique mais dans le cas de cette application, la hiérarchie est réduite à sa plus simple expression. Les arités d'occurrence des différents rôles sont assez triviale également : il n'y a qu'un coordinateur, qu'un conférencier (même si l'agent tenant ce rôle peut évoluer) et un nombre quelconque d'auditeurs (cf. Figure 2.7).

Enfin, la construction des agents se fait très simplement, il suffit d'"enseigner" à l'agent de base les compétences requises une par une. La couche MAGIQUE se charge ensuite de gérer les connexions des agents et le transport des messages dus aux interactions entre agents. Ces communications n'apparaissent pas explicitement au niveau du code, puisqu'elles sont induites par les invocations de compétences et donc prises en charge par MAGIQUE.

Ces travaux ont été publiés dans (Mathieu and Routier 2001, Mathieu and Routier 2002).

Rage

Le second exemple d'application que je présenterai est un framework de calcul distribué, RAGE, qui a été implémenté avec notre plate-forme MAGIQUE. Cet exemple illustre à nouveau

la facilité de conception et d'implémentation de telles classes d'applications, lorsque l'on analyse ces systèmes en terme de rôles et de compétences.

Le framework RAGE, acronyme de Reckoner AGEnt, est un cadre de développement destiné aux scientifiques non-informaticiens et dédié à la réalisation de calculs distribués. Ce système supporte l'exécution de plusieurs calculs simultanément et peut voir sa puissance de calcul augmenter ou diminuer dynamiquement en fonction de la disponibilité des plate-formes hébergeant les calculs. Dans RAGE, il y a deux types de clients : ceux qui fournissent des calculs à effectuer, et ceux qui proposent leur puissance de calcul. Le premier type de client correspond au scientifique désirant effectuer des calculs, et nécessite donc l'écriture d'un programme. Tandis que le second client est fourni par le framework, et correspond à un programme que les utilisateurs n'ont qu'à exécuter pour augmenter la puissance de calcul de RAGE, à la manière du Grid Computing.

La *simplicité* d'utilisation est le critère principal de ce framework. Cet environnement étant destiné à des non-informaticiens, nous avons défini un faible nombre de concepts que l'utilisateur devra s'approprier pour pouvoir alimenter RAGE en calculs. Ces concepts sont principalement celui de *tâche* et de *résultat*. Une *tâche* est un algorithme qui pourra être distribué dans le système, tandis qu'un *résultat* représente les données produites par la réalisation d'une *tâche* et devant être sauvegardées.

Les principales étapes que nous avons suivies lors de l'analyse et la conception de RAGE sont les mêmes que pour l'application précédente. On commence par la définition des rôles impliqués dans le système, de leurs interactions et de leurs compétences, puis vient la définition de l'organisation des rôles au sein du système et enfin l'association des rôles aux agents concrets.

Identifier les différentes entités qui interviennent dans le système revient à déterminer les fonctions que le système doit fournir, il faut ensuite regrouper ces fonctions de manière homogène. Dans le cas de notre framework de calcul distribué, nous avons identifié les rôles suivants : *Boss* qui est responsable des interactions avec les utilisateurs, *Task Dispatcher* qui se charge de la répartition des *tâches* et gère aussi la tolérance aux pannes, *Platform Manager* gère les agents qui ont à effectuer les calculs des *tâches*, et doit s'assurer que l'approvisionnement en *tâche* est toujours suffisant, *Reckoner Agent* qui calcule les *tâches* et retourne les *résultats*, *Repositories Manager* qui gère le stockage et l'accès aux *résultats* et enfin *Result Repository* qui réalise un miroir de la base de données des *résultats*.

Dans la mesure où nous avons implémenté ce système avec MAGIQUE, nous avons concrétisé ces rôles en développant les compétences nécessaires. Par exemple, le rôle *PlatformManager* est défini par une compétence qui définit son comportement : maintenir un *pool* de tâches et gérer leur distribution aux *Reckoner Agent*s. Un rôle est ainsi défini dans MAGIQUE par un ensemble de compétences. Les interactions ne sont pas réifiées et se trouvent enfouies dans le code des compétences.

Après avoir décrit les rôles, il est nécessaire de définir l'architecture du système à savoir l'organisation des agents dans le système multi-agents. Lorsque l'on travaille avec MAGIQUE, cela signifie que l'organisation doit satisfaire deux contraintes : d'abord un groupement logique des rôles, mais aussi un schéma de communication par défaut facilitant la délégation de requête.

La figure 2.8 représente la hiérarchie des rôles choisie dans RAGE : elle ne définit pas comment les rôles sont associés aux agents concrets, ni comment ces agents sont répartis sur le réseau, mais décrit la topologie des liens hiérarchiques. Cette topologie est l'organisation initiale, qui pourra évoluer au gré des interactions entre les agents.

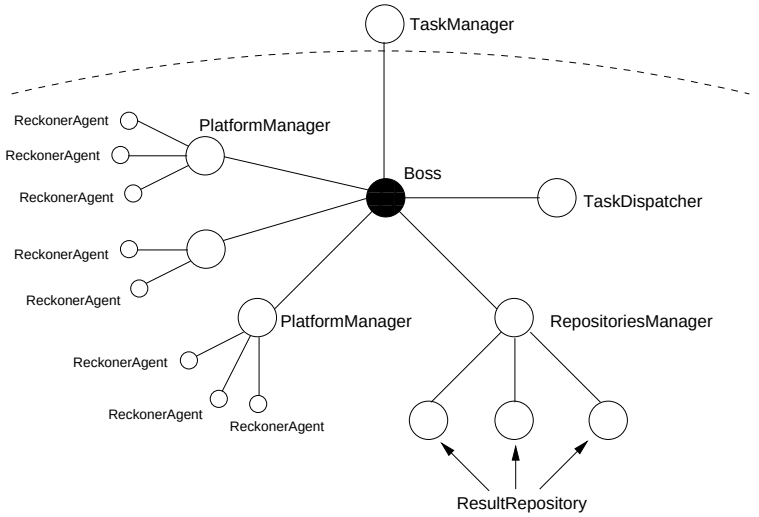

Figure 2.8: La hiérarchie d'agent dans Rage

Cette insfrastucture étant mise en place, l'exploitation de ce framework est simple. On y distingue deux types d'utilisateurs. Celui qui définit le calcul et celui qui prête de la puissance de calcul pour réaliser le calcul. Le premier se contente de définir ce qu'est une tâche élémentaire de son calcul, il s'appuie sur la structure `Task` proposée dans le framework et doit simplement programmer le calcul réalisé par la tâche, sa condition de terminaison/interruption et le résultat qu'elle produit une fois terminée. Par exemple, dans le cadre d'une application de factorisation d'entiers pour casser des clés en cryptographie, une telle tâche pourrait être un simple test de primalité d'un entier. La distribution d'un ensemble de telles tâches entraine la parallélisation massive du calcul. C'est le framework qui prend en charge automatiquement cette distribution et la récolte des résultats. Le client utilisateur n'a donc qu'à lancer un `ReconerAgent` sur sa machine sans être conscient de l'existence des autres agents. Les calculs qu'il réalisera (les tâches) lui seront automatiquement envoyées par la plate-forme.

Ces travaux ont été publiés dans (Mathieu, Routier, and Secq 2002c, Mathieu, Routier, and Secq 2002d).

2.4 RIO

La réalisation de ces différentes applications nous a amenés à une réflexion sur la démarche de conception d'applications multi-agents. Ce travail a été publié dans (Mathieu, Routier, and Secq 2002a, Mathieu, Routier, and Secq 2003c, Mathieu, Routier, and Secq 2003a) et a constitué le cœur de la thèse de Yann Secq (Secq 2 décembre 2003) qu'il a soutenue en décembre 2003 et que j'ai co-encadrée avec le professeur Philippe Mathieu. RIO adopte les principes de la programmation orientée interactions (Singh 1996). Notre approche est similaire à celle proposée dans Gaïa (Wooldridge, Jennings, and Kinny 2000) dans la mesure où nous modélisons les notions de Rôles (abstraits et concrets), de protocoles d'Interaction et d'Organisation, d'où le nom RIO donnée cette démarche. Cependant, à notre sens, Gaïa propose une analyse trop

générale qui ne permet pas un passage facile à la phase de conception. Une de nos préoccupations est de faciliter cette transition. C'est pourquoi RIO propose une description graphique des protocoles d'interaction qui constitue une spécification exécutable de ces interactions à travers un mécanisme automatique de génération des codes associés. Le déploiement est quant à lui facilité par l'utilisation de nos agents atomiques qui peuvent être dynamiquement enrichis de nouvelles compétences.

2.4.1 Spécification exécutable de protocoles d'interaction

Des travaux tels que AGENTUML (Odell, Parunak, and Bauer 2000, Huget 2001) qui étendent les diagrammes de séquence d'UML ou ceux de (Labrou and Finin 1994) à base de réseaux de Pétri colorés ont pour objectif de permettre la spécification de protocoles d'interaction entre agents.

Notre approche emprunte les mêmes principes : représenter l'interaction de manière globale. Cependant, à la différence des travaux précédents, notre objectif est la production de *spécification exécutable* des interactions. C'est-à-dire que, non seulement nous souhaitons décrire le protocole, mais aussi que cette description permette grâce à une génération du code gérant l'interaction, son intégration directe dans le système. Notre modèle a pour but de faciliter la spécification et le déploiement de protocoles d'interactions au sein de systèmes multi-agents. Le concepteur décrit à travers un formalisme graphique la spécification de son interaction (cf. Figure 2.9), les autres étapes étant automatisées.

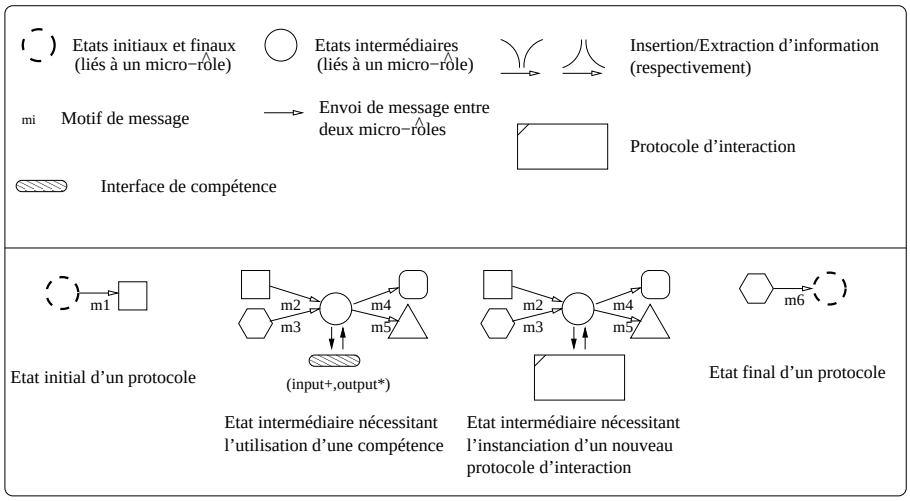

Figure 2.9: Éléments permettant la définition d'un protocole d'interaction.

La description graphique du protocole d'interaction spécifie le déroulement d'une conversation entre les différentes entités impliquées. Celles-ci sont identifiées par ceux que nous avons appelé des *micro-rôles*. Un micro-rôle correspond à l'abstraction d'un participant dans la conversation modélisée. Ils sont représentés par des nœuds du graphe dans notre formalisme. Ils peuvent être associés à un moment de l'interaction à une interface de compétence ou à l'initialisation d'un nouveau protocole. Les arcs qui relient ces micro-rôles correspondent aux envois de messages.

Ils sont étiquetés par des éléments permettant le typage des messages échangés. Le concepteur dispose ainsi d'une vue globale de l'interaction : participants, flux des messages et leur nature, compétences nécessaires.

Une fois le protocole d'interaction décrit dans son ensemble, nous proposons un processus de transformation permettant de produire du code exécutable (cf. Figure 2.10). La description du protocole étant globale, il est nécessaire de générer pour chacun des micro-rôles la vue locale du protocole d'interaction. C'est cette vue locale, une fois traduite en code exécutable, qui peut ensuite être dynamiquement distribuée aux agents du système assumant le micro-rôle concerné.

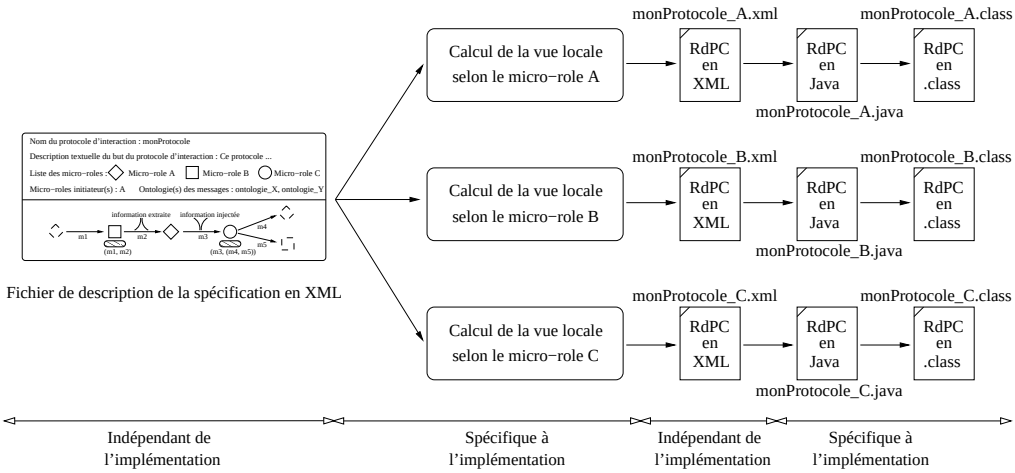

Figure 2.10: Spécification exécutable : de la description graphique au code

Ces travaux ont été publiés dans (Mathieu, Routier, and Secq 2003b)

2.4.2 RIO : démarche méthodologique de conception de systèmes multi-agents

La démarche que nous proposons avec RIO se décompose en quatre phases (cf. Figure 2.11). Les deux premières concernent des spécifications qui peuvent être réutilisées entre différentes applications alors que les deux dernières sont a priori propres à l'application considérée.

La première phase consiste en l'identification des interactions et des micro-rôles qui y prennent part. Le choix de la granularité de ces protocoles a un impact sur leur réutilisabilité. A la fin de cette phase, le concepteur dispose donc d'un ensemble de protocoles d'interaction. De plus, à chacun des micro-rôles sont associées les interfaces de compétences nécessaires à l'interaction.

La phase suivante consiste en la définition des *rôles composites* obtenus par agrégation de micro-rôles de différentes interactions. Il s'agit ici d'un regroupement permettant d'exprimer qu'un rôle donné est généralement impliqué dans plusieurs tâches. La notion de rôle composite donne une cohérence logique entre les micro-rôles qui représentent les différentes facettes d'un code. Un micro-rôle définit donc un rôle abstrait regroupant un ensemble de micro-rôles cohérents.

La troisième phase correspond en la réunion des rôles composites au sein d'*agents abstraits* introduisant ainsi des dépendances entre ces rôles. Il s'agit ici en fait de définir une société

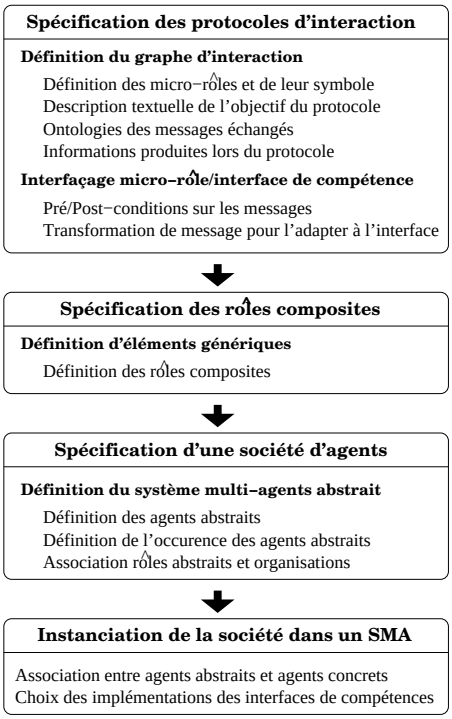

Figure 2.11: La démarche méthodologique RIO

abstraite d'agents ou, en plus de ces patrons d'agents, on précise la cardinalité de ces agents et la structure organisationnelle retenue, ce qui revient à choisir le modèle d'accointance.

Enfin, il faut déployer la société d'agents. Les agents abstraits sont alors projetés sur les agents concrets du système en respectant les arités définies. C'est également lors de cette quatrième phase qu'est réalisée la liaison entre une interface de compétences et sa réalisation.

La Figure 2.12 reprend l'ensemble de ce processus illustré par un rapide exemple.

En adoptant les principes de la programmation orientée interactions et en proposant un modèle de spécifications exécutables des protocoles d'interaction, RIO propose une méthodologie pour la réalisation d'applications multi-agents distribuées. Le principal apport est de donner au concepteur une vue globale de chaque interaction tout en laissant au système le travail de projection de cette vue globale sur l'ensemble des vues locales des différentes entités. La démarche proposée permet également une construction incrémentale des systèmes multi-agents applicatifs puisque, de même que notre agent générique peut se voir enrichi de nouvelles compétences, le système peut être enrichi par des interactions.

Bibliographie

Adam, E.; and R. Mandiau. 2005. "Roles and hierarchy in multi-agent organizations," in *Proceedings of Multi-Agent Systems and Applications IV, 4th International Central and Eastern European Conference on Multi-Agent Systems, CEEMAS 2005*, ed. by M. Pechoucek, P. Petta, and L. Varga, no. 3690 in LNAI, 539–542. Springer-Verlag.

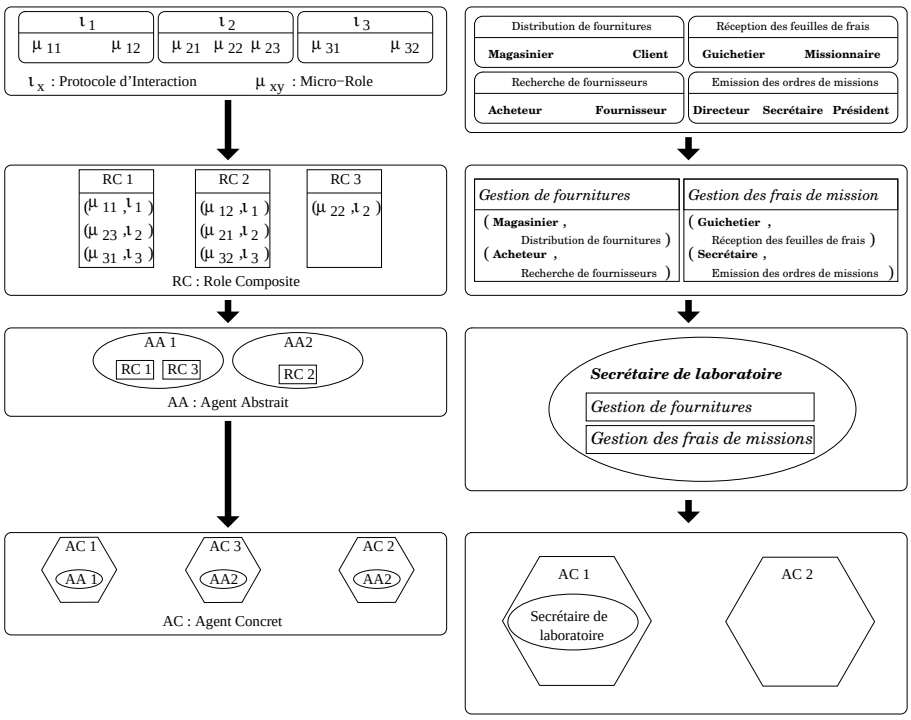

Figure 2.12: Synthèse de la démarche RIO mise en parallèle avec l'exemple d'un agent "secrétaire de laboratoire" impliqué dans la gestion de fourniture et celle des frais de mission.

Adam, E.; R. Mandiau; and C. Kolski. 2001. "Application of a holonic multi-agent system for cooperative work to administrative processes," *Journal of Applied Systems Studies*, 2, 110–115.

Bensaid, N.. 1999. "MAGIQUE, une architecture multi-agents hiérarchique," Ph.D. thesis, Université des Sciences et Technologies de Lille.

Bensaid, N.; and P. Mathieu. 1995. "Un modèle d'architecture multi-agents entièrement écrit en Prolog," in *IV Journées Francophones de Programmation Logique, JFPL'95*, 381–385, Dijon-France. teknea, Toulouse-France.

Bensaid, N.; and P. Mathieu. 1997a. "A Hybrid and Hierarchical Multi-Agent Architecture Model," in *Proceedings of PAAM'97*, 145–155.

Bensaid, N.; and P. Mathieu. 1997b. "A Hybrid Architecture for Hierarchical Agents," in *Proceedings of International Conference on Computationnal Intelligence and Multimedia applications, ICCIMA'97*, 91–95.

Clement, R.. 2000. "To Buy or to Contract Out: Self-Extending Agents in Multi-Agent Systems," in *SOMAS: A Workshop on Self Organisation in Multi Agent Systems*.

da Silva, J. T.; and Y. Demazeau. 2002. "Vowels Co-ordination Model," in *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'02)*, 1129–1136.

Ferber, J.; and O. Gutknecht. 1998a. “A meta-model for the analysis and design of organizations in multi-agent systems,” in *Proceedings of ICMAS’98*.

Ferber, J.; and O. Gutknecht. 1998b. “A meta-model for the analysis and design of organizations in multi-agent systems,” in *Proceedings of ICMAS’98*.

Ferber, J.; O. Gutknecht; and F. Michel. 2000. “The MadKit Agent Platform Architecture,” Discussion paper, W3C.

Franklin, S.; and A. Grasser. 1996. “Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agent,” in *Proceedings of the 3rd International Workshop on Agent Theories, Architectures, and Languages*. Springer-Verlag.

Gerber, C.; J. Siekmann; and G. Vierke. 1999. “Holonic Multi-Agent Systems,” Discussion paper, DFKI GmbH.

Ghanea-Hercock, R.. 2000. “Spontaneous Group Formation in Multi-Agent Systems,” in *SOMAS: A Workshop on Self Organisation in Multi Agent Systems*.

Guessoum, Z.. 1996. “Environnement de développement et de conception de systèmes multi-agents,” Ph.D. thesis, Université de Paris 6.

Horling, B.; and V. Lesser. 1998. “A Reusable Component Architecture for Agent Construction,” Discussion paper, UMass Computer Science.

Huget, M.-P.. 2001. “Une ingénierie des protocoles d‘interaction pour les SMA,” Ph.D. thesis, Université de Paris 9.

Jennings, N.; and M. Wooldridge. 2000. *Handbook of Agent Technology*chap. Agent-Oriented Software Engineering. AAAI/MIT Press.

J.Ferber; O. Gutknecht; and F. Michel. 2000. “MadKit: une plate-forme multi-agent générique,” Discussion paper, FIPA.

Kinny, D.; M. Georgeff; and A. Rao. 1996. “A Methodology and Modelling Technique for Systems of BDI Agents,” Discussion paper, Australian AI Institute.

Labrou, Y.; and T. W. Finin. 1994. “A Semantics Approach for KQML - A General Purpose Communication Language for Software Agents,” in *CIKM*, 447–455.

Mathieu, P.; J. Routier; and Y. Secq. 2002a. “Dynamic Organization of Multi-Agent Systems,” in *Proceedings of the AAMAS’02 Conference*, 451–452.

Mathieu, P.; and J.-C. Routier. 2000-2001. “Tutoriel de Magique,” Equipe SMAC - LIFL.

Mathieu, P.; and J.-C. Routier. 2001. “Une contribution du multi-agent aux applications de travail coopératif,” *TSI Hermès Science Publication. Réseaux et Systèmes Répartis. Calculateurs Parallèles.*, 13. Numéro spécial télé-applications, 207–226.

Mathieu, P.; and J.-C. Routier. 2002. “A Multi-Agent Approach to Co-operative Work,” in *Proceedings of CADUI’02*, ed. by C. Kolski, and J. Vanderdonckt, 367–380. Kluwer Academics Press.

Mathieu, P.; J.-C. Routier; and Y. Secq. 2001. "Dynamic Skill Learning: A Support to Agent Evolution," in *Proceedings of AISB'01*, 25–32.

Mathieu, P.; J.-C. Routier; and Y. Secq. 2002b. "Principles for dynamic multi-agent organizations," in *Proceedings of 5th Pacific Rim International Workshop on Multi-Agents. PRICAI2002-PRIMA2002*, ed. by K. Kuwabara, and J. Lee, vol. 2413 of *LNAI*, 109–122. Springer-Verlag.

Mathieu, P.; J.-C. Routier; and Y. Secq. 2002c. "RAGE: An agent framework for easy distributed computing," in *Proceedings of AISB'02*, 20–24.

Mathieu, P.; J.-C. Routier; and Y. Secq. 2002d. "Using agents to build a distributed calculus framework," *The Interdisciplinary Journal of Artificial Intelligence and the Simulation of Behaviour*, 1(2), 197–208.

Mathieu, P.; J.-C. Routier; and Y. Secq. 2003a. "RIO: Rôles, Interactions et Organisations," in *Actes des Secondes Journées Francophones sur les Modèles Formels de l'Interaction, MFI'03*, ed. by A. Herzig, B. Chaib-draa, and P. Mathieu, 179–188.

Mathieu, P.; J.-C. Routier; and Y. Secq. 2003b. "Runnable specifications of interactions protocols for open multi-agent systems," in *Proceedings of the 2003 International Conference on Information and Knowledge Engineering, IKE'03*, ed. by N. Goharian, vol. II, 431–437.

Mathieu, P.; J.-C. Routier; and Y. Secq. 2003c. "RIO : Roles, Interactions and Organizations," in *Proceedings of the 3rd International/Central and Eastern European Conference on Multi-Agent Systems, CEEMAS 2003*, ed. by V. Marik, J. Müller, and M. Pechoucek, 147–157.

Occello, M.; and J.-L. Koning. 2000. "Multiagent oriented software engineering: An approach based on model and software reuse.," in *Proceedings of Second International Symposium "From Agent Theory to Agent Implementation"*, 25–28.

Odell, J.; H. Parunak; and B. Bauer. 2000. "Extending UML for Agents," .

Petrie, C.. 2001. "Agent-Oriented Software Engineering," *Lecture Notes in AI*.

Ricordel, P. M.. 2001. "Programmation Orientée Multi-Agents :Développement et Déploiement de Systèmes Multi-Agents Voyelles," Ph.D. thesis, Institut National Polytechnique de Grenoble.

Ricordel, P.-M.; and Y. Demazeau. 2002. "VOLCANO : a vowels-oriented multi-agent platform.," in *Proceedings of the International Conference of Central Eastern Europe on Multi-Agent Systems (CEEMAS'01), From Theory to Practice in Multi-Agent Systems*, 252–262.

Secq, Y.. 2 décembre 2003. "RIO : Rôles, Interactions et Organisations, une méthodologie pour les systèmes multi-agents ouverts," Ph.D. thesis, Université des Sciences et Technologies de Lille, Co-dirigée avec le Professeur Philippe Mathieu.

Shoham, Y.. 1993. "Agent-Oriented Programming," *Artificial Intelligence*, 60, 51–92.

Singh, M. P.. 1996. "Toward Interaction-Oriented Programming," Discussion Paper TR-96-15, Department of Computer Science, North Carolina State University.

Travers, M.. 1996. "Programming with Agents: New metaphors for thinking about computation," Ph.D. thesis, MIT.

Wooldridge, M.; N. Jennings; and D. Kinny. 2000. “The Gaia Methodology for Agent-Oriented Analysis and Design,” *Journal of Autonomous Agents and Multi-Agent Systems*.

Chapitre 3

CoCoA : simulation de comportements rationnels.

Mon second projet au sein de l'équipe SMAC est le projet CoCoA, pour Collaborative Cognitive Agents. Ce projet est né à l'occasion d'un contrat PRIAMM avec la société éditrice de jeux vidéos Cryo Interactive.

Son objectif est la proposition d'un modèle permettant la simulation de comportements rationnels pour des agents évoluant dans des environnements situés. Modéliser des comportements réalistes devient très délicat dès que les interactions deviennent complexes. Il est notamment difficile de réutiliser (au sens du Génie Logiciel) les interactions antérieurement définies et difficile d'adapter le comportement de l'agent aux changements externes (l'environnement) et internes (les capacités de l'agent). Il devient alors capital de séparer le modèle d'interactions des interactions elles-mêmes. L'intérêt du monde des jeux vidéo pour une telle thématique est évidemment d'accroitre le réalisme des univers de jeux. Les entités qui y apparaissent doivent manifester des comportements les plus réalistes possibles. Cependant, cette problématique ne se limite pas aux jeux vidéo mais concerne en fait les simulations dans lesquelles la notion de comportement individuel a un sens et les simulations centrées individus en général. Les domaines d'application sont multiples, que ce soit des applications éducatives, le cinéma, ou les simulations de situations d'urgence (Querrec, Reignier, and Chevaillier 2001).

D'autres projets internationaux effectuent des travaux similaires sur les comportements d'agents pour les jeux vidéos. On peut notamment citer les projets Excalibur (Nareyek 2000), Soar (Laird and Duchi 2000, Laird and Rosenbloom 2005) et Cog (Brooks and al. 1999). Cependant, l'arrêt du premier vient d'être annoncé, le second a une approche plus réactive et proche de celles des scripts, enfin l'objectif du troisième est autre, plus ambitieux et à nettement plus long terme, puisqu'il vise à construire un robot humanoïde intelligent. Notre approche est cognitive. L'objectif de nos travaux est d'offrir un outil de modélisation générique de comportements d'agents pour les concepteurs de simulations en insistant sur la séparation du déclaratif et du procédural pour une meilleure réutilisabilité. Nous nous attachons notamment à rendre le moteur comportemental de l'agent indépendant des interactions que celui-ci peut être amené à exécuter.

Dans le cadre de ce travail, nous avons conçu un modèle original basé sur une dualité des interactions possibles entre les agents. Celui-ci offre une grande diversité de type d'agents, et un comportement adaptatif en fonction de leur environnement et de leur capacité. Un système de planification est fourni à chaque agent pour décider de l'action à effectuer en fonction du contexte.

Notre travail a abouti à la réalisation d'une plate-forme logicielle opérationnelle qui nous permet maintenant de concevoir des applications de simulation avec des interactions complexes. Plus récemment nous avons également orienté nos travaux sur la gestion d'équipes d'agents rationnels.

Enfin, précisons que ce qui nous intéresse est l'exécution de la simulation et non son résultat. Ainsi, à la différence de simulations basées sur des équations mathématiques, nous ne cherchons pas à connaître l'état d'un système à l'issue de la simulation mais nous voulons examiner le déroulement de la simulation "pas à pas". Ce sont les comportements individuels des agents qui sont simulés et la simulation résulte de la somme de ces comportements observés. Il s'agit donc de simulations *centrées-individus*. Un telle approche permet des applications différentes des simulations orientées "population" ou de celles où ce sont les interactions sociales qui sont étudiées (Deffuant, Ferrand, Bernard, and Azembourg 1999). De plus, la progressivité de leur déroulement permet d'apporter un niveau explicatif au résultat de la simulation dans la mesure où la simulation peut être suivie action par action, et donc justifiée pas à pas.

Je commencerai par discuter de la position de cette thématique par rapport à sa motivation initiale qui est celle des jeux vidéos et des problématiques de ce domaine. Je présenterai ensuite notre proposition basée sur les interactions avant de décrire le moteur comportemental des agents impliqués dans les simulations. Je parlerai ensuite d'équipes d'agents qui s'appuient sur les mêmes concepts.

3.1 Simulation de comportements et jeux vidéos

Le monde du jeu vidéo est confronté aux problèmes posés par la simulation de comportements de types humains. Il est largement accepté que l'Intelligence Artificielle est le nouveau challenge des jeux vidéos après que le graphisme ait occupé ces dernières années l'essentiel des ressources R&D de cette industrie. Aux difficultés scientifiques et techniques dues à l'Intelligence Artificielle viennent s'ajouter des contraintes économiques. Il s'agit essentiellement du temps requis pour le développement des jeux. Celui-ci dépend dans une large mesure de la possibilité de réutiliser des éléments issus de réalisations précédentes. L'apparition de moteurs graphiques et physiques ont permis cela pour le développement des parties visuelles des jeux. Dans le cas de l'intelligence artificielle des personnages, les techniques employées jusque maintenant rendent difficiles, voire impossible, cette réutilisation.

La recherche en intelligence artificielle a, elle aussi, à gagner de cet intérêt du monde du jeu vidéo. Le poids économique de cette industrie en fait un domaine d'application pouvant justifier à lui seul cette recherche. Il suffit de regarder combien l'informatique graphique a pu en profiter ces dernières années pour s'en convaincre. L'intérêt pour l'intelligence artificielle n'est pas seulement économique, les jeux vidéos constituent en effet un champ d'expérimentation idéal pour la simulation de comportements rationnels. Selon John Laird, ils constituent même la "killer application" pour le développement d'une IA de type humain (Laird and van Lent 2000). Les personnnages non joueurs impliqués dans les jeux vidéos doivent en effet être perçus comme des entités autonomes offrant des comportements de plus en plus réalistes. Les comportements proposés doivent correspondre à ce qu'attend le partenaire ou adversaire humain. Les personnages virtuels doivent pouvoir s'adapter au contexte et aux nouvelles situations rencontrées. De plus, ceux-ci doivent tenir compte des autres entités du jeux et la mise en place de stratégies d'équipes est souvent utile. Dans cette optique, les agents doivent amener l'observateur humain à penser

qu'ils se comportent de manière intelligente et rationnelle, c'est-à-dire comme un humain aurait pu se comporter dans une situation similaire pour accomplir ses propres buts. L'objectif ultime est que dans un contexte où se mèlent les joueurs humains et virtuels, il ne soit pas possible de faire la différence entre eux. On arriverait ainsi à une satisfaction partielle du test de Turing. Notamment, un des avantages apportés par les jeux est qu'ils proposent des contextes dans lesquels la réalité est simplifiée. Les mondes simulés réduisent ainsi les comportements possibles et les choix offerts aux personnages à chaque instant simplifiant la tâche qui autrement serait inabordable à moyenne échéance. En ce sens, si test de Turing il y a, il s'agit bien d'un test dégradé.

Un certain nombre de propositions ont été faites concernant les jeux vidéos et les agents (Nareyek 2004) mais la plupart concernent des agents réactifs (Nareyek 1998). Si ils constituent une réponse possible à la simulation de comportements, les agents réactifs offrent cependant des comportements limités influencés essentiellement par le court terme plutôt que par des buts. Leur capacité à réaliser des tâches dépend de leur environnement immédiat plus que d'une réelle volonté. De plus, dans le cadre des jeux vidéos, ces comportements réactifs sont quasi systématiquement mis en place à l'aide de scripts. Les défauts de cette approche sont reconnus et ont été souvent soulignés (Tozour 2002). Les comportements qu'ils permettent sont le plus souvent trop figés et peu adaptatifs. Même si certains essaient de contourner leurs défauts (Ponsen and Spronck 2004), les améliorations apportées ne sont que partielles. De plus l'utilisation des scripts est une des raisons principales pour laquelle il est quasiment impossibles de réutiliser des éléments de simulation de comportements d'un jeu vers un autre. Au contraire, notre proposition, que je détaillerai par la suite, se base sur des agents proactifs, cognitifs, dirigés par des buts.

Le fait que cette thématique se trouve au confluent de plusieurs domaines dont elle cumule les problèmes constitue déjà en soi une difficulté. D'une part on trouve des préoccupations de l'IA "classique" telle que la représentation de la connaissance et sa manipulation, ainsi que la construction de plans d'actions. D'autre part, la naturelle concurrence des mondes simulés ainsi que la nécessité de mettre en place des situations de coordination entre agents et des stratégies d'équipes amène à considérer les problèmes de l'IA distribuée. Enfin, les préoccupations de génie logiciel doivent être présentes afin de permettre une réutilisabilité des différents éléments de comportements entre différentes simulations.

3.2 Approche et contexte

Comme je l'ai annoncé précédemment, les préoccupations sont multiples. Non seulement il s'agit de modéliser et simuler des comportements réalistes, mais il faut aussi que la proposition prenne en compte les contraintes de conception des simulations et de réutilisabilité des éléments d'une simulation à une autre.

Notre approche se veut donc générique. Nous stipulons qu'un même formalisme peut être utilisé pour modéliser et concevoir des comportements réalistes, c'est-à-dire crédibles, destinés à évoluer dans des mondes artificiels. Ainsi dans notre proposition, le moteur comportemental cognitif reste le même d'une simulation à l'aure et les composants comportementaux peuvent être, au moins partiellement, réutilisés. Le principe fondateur est de baser la dynamique des simulations et la représentation des connaissances sur les interactions entre les agent impliqués dans les simulations : certains agents peuvent effectuer des interactions que d'autres agents peuvent

subir. Ces interactions permettent de modéliser les règles qui régissent le monde simulé et constituent les éléments de connaissance réutilisables entre simulations. Le moteur comportemental des agents actifs est en effet le même et produit des comportements qui dépendent du contexte dans lequel l'agent évolue ainsi que des interactions dont il est doté. Ainsi donc, un changement dans les interactions d'un agent animé entraine une adaptation de son comportement pour prendre en comte ces nouvelles capacités.

Les agents mis en œuvre dans ces simulations sont donc situés dans leur environnement. Celui-ci est muni d'un espace euclidien. Il a donc une géographie et les notions de "position", "voisinage", "déplacement", etc. ont un sens et une influence. A tout moment ces agents ont une perception partielle de leur environnement. Cette perception leur permet de se construire une représentation de l'environnement. Cependant ce dernier est dynamique et concurrent, en conséquence les informations dont dispose un agent sur son environnement sont non monotones. Une information acquise par l'agent peut devenir fausse ou non pertinente après un certain temps. Il en résulte que la connaissance qu'a l'agent de l'environnement est généralement incomplète et potentiellement fausse. Le raisonnement de l'agent s'appuie cependant sur *sa* représentation de l'environnement, il peut donc être amené à commettre des erreurs ou à devoir réviser ses intentions en raison de ce décalage entre ses croyances et la réalité.

Nous distinguons dans nos simulations deux types d'agents. Les agents *inanimés* correspondent en fait aux objets qui peuplent le monde simulé. Plus intéressants, les agents *animés* ont des capacités et disposent d'un moteur comportemental. Ce sont les acteurs des simulations. C'est de ces agents dont il était question dans les paragraphes précédents et ce sont essentiellement d'eux dont nous parlerons par la suite. Ils ont généralement des buts et leur moteur comportemental se base sur leurs connaissances et capacités pour proposer un comportement permettant de satisfaire au mieux et "rationnellement" ces buts. Attardons nous quelques instants sur ce "rationnel". Il n'est pas évident de trouver le terme qui qualifie le mieux les comportements que nous cherchons à produire. Le terme "intelligent" semble trop prétentieux et ambitieux, il implique trop de capacités et entraine trop de débats pour que nous le retenions. On sait bien les discussions qui existent autour d'une définition (y en a-t-il "une" d'ailleurs ?) du terme *intelligence artificielle*. C'est pourquoi nous lui préférons les termes de "rationnels" ou "crédibles" pour qualifier les comportements de nos agents, "raisonnables" ou "plausibles" seraient également des possibilités. Ils sont moins connotés mais restent néanmoins subjectifs. Ainsi nous définissons comme *rationnel* un comportement si, à tout moment, la décision d'effectuer une action aurait raisonnablement pu être prise par un humain qui aurait eu les mêmes informations que l'agent et qui disposerait des mêmes capacités. L'évaluation de ce critère ne peut donc être faite que par des juges extérieurs à la simulation.

Une conséquence évidente de ce point de vue est que nous ne cherchons pas à obtenir des comportements optimaux, que ce soit en terme de déplacements ou de nombre d'actions effectuées par exemple. Ainsi nous ne visons pas l'obtention du "meilleur" comportement (et encore faudrait il définir le critère d'évaluation). Ce qui importe le plus c'est d'éviter les comportements aberrants, c'est-à-dire manifestement non raisonnables. Il est probablement important de rappeler maintenant que dans le type de simulations que nous visons, les comportements sont exécutés "pas à pas" et qu'il est donc possible de suivre en détail la démarche de l'agent. Ainsi l'état final de l'environnement à l'issue de la simulation n'est pas le plus important. Il importe plus de pouvoir observer et étudier le déroulement du comportement individuel des agents.

Ces derniers points : rationnalité des comportements observés et exécution pas à pas, prennent une importance particulière du fait que nos environnements de simulation sont *situés*. Pour pouvoir exécuter les actions commmandées par leur comportement et interagir avec les autres éléments de la simulation, les agents vont en effet être amenés à se déplacer dans l'environnement. Dans une exécution "pas à pas" ces déplacements vont constituer une part importante du comportement observé et représenteront donc un facteur majeur pour évaluer la rationnalité du comportement, d'autant plus qu'ils sont particulièrement visibles pour l'observateur extérieur. Le caractère situé de nos simulations va donc avoir un impact sur les plans d'action des agents. A la séquence d'actions calculée pour résoudre les objectifs de l'agent vont venir s'ajouter les déplacements nécessaires pour atteindre les positions de l'environnement concernées dans le cadre de la simulation. Evidemment pour un même objectif et une même séquence d'actions, les déplacements nécessaires varieront pour des environnements différents. D'autre part, il est possible que, pour pouvoir être exécuté, un déplacement requiert l'exécution d'autres actions qui devront être intégrées à la séquence initiale. Ainsi on peut distinguer ce que nous appelons le *plan abstrait* du *plan d'exécution*. Le premier correspond au plan d'actions dans lequel les positions relatives ou absolues des agents ne sont pas prises en compte, il est indépendant d'un environnement situé. Le second correspond au plan dû à l'exécution dans un environnement particulier.

Le plan abstrait fournit les actions permettant la résolution effective des objectifs. C'est lorsque l'action a que doit exécuter un agent se révèle être une interaction avec un autre agent de la simulation, et c'est souvent le cas, que le caractère situé intervient. Le plus souvent des contraintes de proximité entre les deux agents s'appliquent afin que l'interaction puisse effectivement avoir lieu. En conséquence il est fort possible qu'avant de pouvoir exécuter son (inter)action, l'acteur devra se déplacer pour s'approcher de sa cible. Ce déplacement d devra donc être intégré au plan pour permettre le déroulement de la simulation. Le plan initial "*exécuter a*", devient donc *exécuter d puis a* (Cf. Figure 3.1).

Mais cela soulève un nouveau problème lorsqu'un déplacement requiert à son tour une planification pour pouvoir être exécuté correctement. C'est par exemple le cas lorsque ce déplacement amène l'agent à franchir une porte fermée. L'ouverture de cette porte et la ou les actions qu'elle nécessite devront être également intégrées au plan. Le déplacement *d* est donc décomposé en deux sous-déplacements d_1 et d_2 interrompus par l'ouverture de la porte. On en arrive donc au plan d'exécution *exécuter* d_1, *ouvrir la porte, exécuter* d_2 *puis a*. Les choses se complexifie encore un peu plus lorsque l'ouverture de la porte n'est pas atomique et que la résolution de cet objectif nécessite à son tour une planification. Planification qui entraine la production d'un plan abstrait et de son plan d'exécution contenant ses propres déplacements qui ont un impact sur ceux déjà planifiés. C'est par exemple le cas lorsque la porte en question est cadenassée et que l'agent doit donc préalablement prendre la clé nécessaire et donc se déplacer jusque celle-ci avant d'aller à la porte. Le sous-plan préalable à l'ouverture de la porte est donc *exécuter le déplacement* d'_1, *prendre la clef, exécuter* d'_2, *décadenasser la porte*. Il remplace le déplacement d_1 qui n'a plus lieu d'être. Le schéma récursif de ce raisonnement est évident et l'on peut imaginer que la récupération de la clef engendre de nouvelles actions. On le voit sur ce petit exemple, un plan abstrait initial simple comme l'était *exécuter a* se voit fortement enrichi du fait du contexte situé. Ce point est également repris dans la publication (Devigne, Mathieu, and Routier 2004). Le mémoire de DEA de Damien Devigne s'intéressait déjà à ce problème (Devigne 2003).

Les travaux sur les robots mobiles font eux aussi intervenir une planification pour la résolution d'objectifs et un déplacement. Il est clair que ces robots évoluent en environnement situé. Dans ces travaux, planification et déplacement sont cependant traités séparément par des

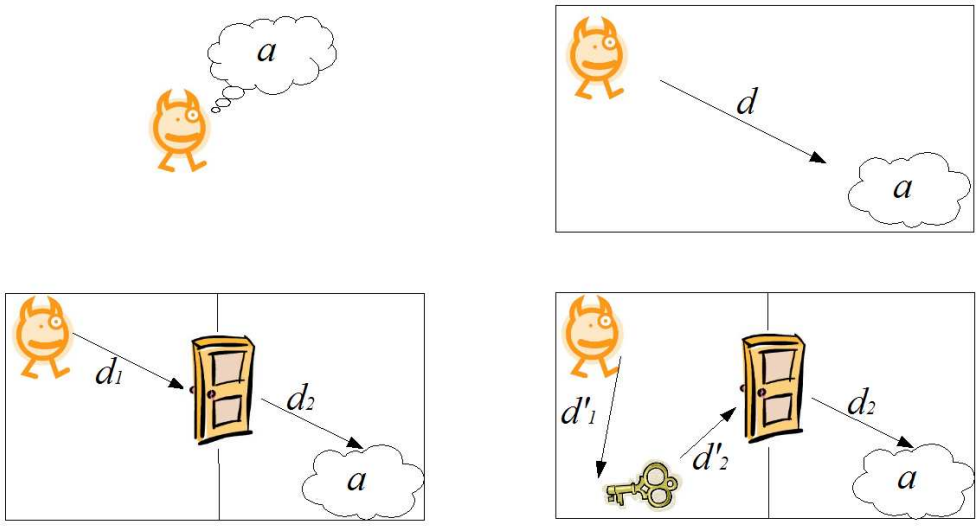

Figure 3.1: Influence du situé sur la planification. En haut à gauche, le plan abstrait. En haut à droite, dans un contexte situé le déplacement doit être prévu pour atteindre l'endroit d'exécution de l'action. En bas à gauche, l'environnement impose une planification, ici ouverture d'une porte. En bas à droite, cas où l'agent sait - ou croit - que la porte est cadenassée et doit donc prendre la clef avant de s'y rendre. (les actions *ouvrir* et *décadenasser* n'apparaissent pas sur les figures)

modules différents. On retrouve donc d'une certaine manière notre plan abstrait et le plan de d'exécution/déplacement. Cependant dans ces travaux, l'environnement, même si il est constitué d'obstacles inconnus qu'il faut détourner ou légèrement déplacer, n'amène pas à devoir planifier des actions permettant les déplacements comme je l'ai décrit ci-dessus. De plus dans ces travaux, la demande de rationnalité n'est pas la même que la notre, on demande que le robot accomplisse sa tâche, qu'il ait parfois eu passagèrement des choix d'actions "bizarres" importe peu. Pour ces raisons, si il y a une certaine proximité, la problématique de ces travaux diffère de la notre sur ces points et nous amène à faire des propositions différentes notamment avec l'intégration des déplacements dans la planification.

Après avoir positionné le contexte et les contraintes qui lui sont inhérentes, je vais maintenant détailler notre approche basée sur les interactions avant de présenter le modèle d'agent cognitif que nous proposons pour la réalisation de nos simulations.

3.3 Des agents définis par leurs compétences et des simulations orientées interaction

Avant de détailler nos travaux dans les sections suivantes, je vais en présenter les grandes lignes qui reposent essentiellement sur la notion d'*interaction*.

L'environnement mis à part, tous les éléments de la simulation sont appelés *agents*. On y distingue les objets du monde simulé de ses acteurs. Les premiers sont appelés *agents passifs* ou *inanimés* et les seconds *agent actifs* ou *animés*.

Mais le point central de notre proposition est la notion d'interactions. Celles-ci constituent la base de la représentation de connaissance dans la mesure où elles définissent les lois qui régissent le monde simulé. Les interactions décrivent en effet les actions qui peuvent être accomplies et leurs effets.

Ainsi, les propriétés principales des agents sont des listes d'interactions. Celles-ci sont appelées *peut-subir* et *peut-effectuer* (Cf. Figure 3.2), la seconde n'existant que pour les agents animés. Ces propriétés ont pour valeur des listes d'interactions. La première correspond aux interactions dont l'agent peut être la cible, la seconde à celles dont il peut être l'acteur.

```
agent        := agent-passif | agent-actif
agent-passif := { ( "nom", symbole),
                    ("peut-subir", {interaction* }),
                    propriété*}
agent-actif  := agent-passif ∪
                    {("peut-effectuer", {interaction*}),
                    ("buts", but*),
                    ("mémoire", environnement),
                    ("moteur", moteur)}
propriété    := (nom_propriété, Valeur)
```

Figure 3.2: Définition d'un agent

Les interactions que peut effectuer un agent animé représentent en fait les capacités ou compétences de cet agent. C'est bien la la valeur de sa propriété *peut-effectuer* qui détermine ce que l'agent peut accomplir et donc quel(s) rôle(s) il peut jouer dans la simulation. On retrouve ici une approche similaire à celle que nous avons développée dans le cadre du projet MAGIQUE présenté au chapitre précédent.

Ce sont les interactions qui sont à la base de la dynamique de nos simulations. Le principe général est la mise en correspondance d'un agent *acteur* et d'un agent *cible*, le premier exécutant l'interaction sur le second (cf. Figure 3.3). Ce principe peut être décrit ainsi :

$$a \in Actif, t \in Agent, \text{si } \exists i \in a.\textit{peut-effectuer}() \cap t.\textit{peut-subir}(), \text{alors } a.executer(i,t)$$

L'interaction à exécuter est en fait déterminée par le *moteur de comportements* de l'agent a afin qu'elle lui permette de réaliser son *but*. L'agent a doit alors rechercher la cible t qui convient effectivement.

3.4 Les interactions

Les interactions définissent les lois qui régissent le monde simulé. La connaissance y est exprimée de manière déclarative. De ce fait, le plus souvent, une interaction n'est pas attachée à une simulation en particulier dans la mesure où elle représente une connaissance "universelle". Cela constitue une contrainte au niveau de la représentation des interactions en obligeant le moteur

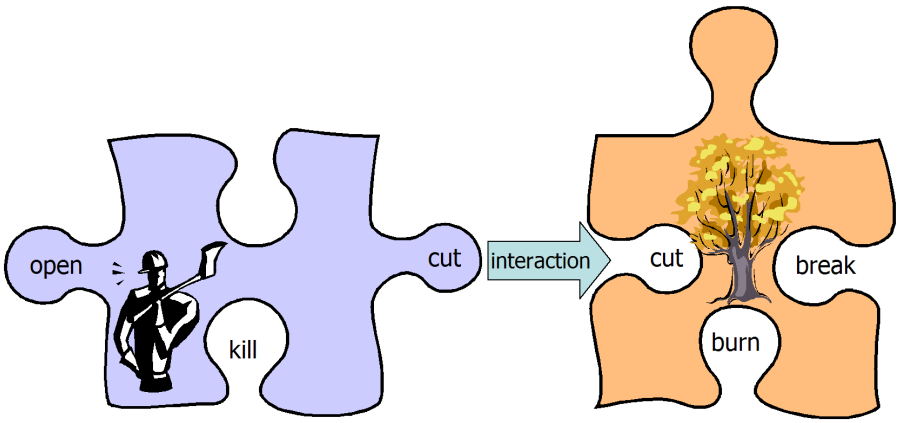

Figure 3.3: Les agents sont décrits en terme d'interactions qu'ils peuvent subir ou effectuer, ici un agent actif *bucheron* peut effectuer les interactions *open* et *cut* et subir *kill*, alors qu'un agent passif *arbre* peut subir *cut*, *break* et *burn*. Le principe central de la simulation est l'assocation d'un acteur effectuant une interaction et d'une cible la subissant, comme l'interaction *cut* ici.

de raisonnements qui les manipule à être générique. Mais de ce fait, la plupart de ces interactions peuvent être réutilisées dans différentes simulations et la généricité imposée au moteur des agents permet de satisfaire l'un des objectifs évoqués précédemment.

```
acteur          := l'agent qui réalise l'interaction
cible           := l'agent qui subit l'interaction

interaction     := (nom, garde, condition, action)
nom             := symbole
garde           := d op nombre
d               := la distance entre l'acteur et la cible
op              := = | > | <
condition       := test_propriete | primitive(args)
test_propriete  := { acteur | cible }.nom_propriété op Valeur
action          := affect_propriété | primitive
affect_propriété:= { acteur | cible }.nom_propriété = Valeur
primitive       := { acteur | cible }.nom_primitive(args)
nom_primitive   := Symbole
nom_propriété   := Symbole
```

Figure 3.4: Définition d'une interaction

Nous avons déjà évoqué il y a quelques lignes les agents *acteur* et *cible* qui caractérisent une interaction. Outre son nom, la description d'une interaction se fait classiquement en trois parties (cf. Figure 3.4) :

la condition Elle permet de tester le contexte d'exécution courant. Cette partie consiste essentiellement à tester les valeurs de propriétés de la cible et de l'acteur.

la garde Il s'agit ici d'exprimer des conditions plus générales d'applicabilité de l'interaction. En particulier, les aspects liés à l'aspect situé des simulations est exprimé dans cette garde et non pas dans la partie *condition*. Le plus souvent il s'agit d'exprimer que l'acteur doit être suffisamment proche de la cible pour que l'interaction puisse être effectuée. C'est pour cette raison que la *garde* est séparée de la *condition*. Dans un contexte non situé, on ne trouverait que cette dernière partie alors que les conditions de la *garde* sont à l'origine des déplacements dans le plan. Grâce à cette garde, il n'est pas nécessaire d'exprimer explicitement les déplacements dans l'interaction. L'expression déclarative des contraintes spatiales entrainera automatiquement la planification de ces déplacements si ils s'avèrent nécessaires.

l'action Elle décrit les conséquences de l'exécution de l'interaction : changement des valeurs de propriétés de l'acteur ou de la cible, effet de bord sur l'environnement, etc.

Illustrons cette description avec l'exemple de l'interaction *ouvrir* (cf. Figure 3.5). La *condition* exprime ici que la cible ne doit pas déjà être ouverte pour pouvoir subir *ouvrir*. Si elle semble triviale, cette connaissance est néanmoins nécessaire si l'on veut éviter des aberrations telles qu'un agent cherchant désespérément à ouvrir une porte ouverte... L'*action* exprime quant à elle le changement d'état de la cible une fois que l'interaction a été exécutée. Enfin, la garde exprime que l'acteur doit être à côté de la cible pour pouvoir exécuter celle-ci. Cette connaissance évite que nos agents deviennent télékynésiste... Elle n'aurait pas d'importance dans un contexte non située.

$$open: \begin{cases} \texttt{condition} = \text{“}\texttt{target.opened} = \texttt{false}\text{”} \\ \texttt{guard} = \text{“}\texttt{d} < 1\text{”} \\ \texttt{action} = \text{“}\texttt{target.opened} = \texttt{true}\text{”} \end{cases}$$

Figure 3.5: L'interaction *ouvrir*. L'expression de la connaissance est déclarative, en particulier la nature de la cible n'est pas explicite.

Comme on peut le constater à la lecture du code de cette interaction, aucune référence n'est faite à un type particulier de cible. Celle-ci peut aussi bien être une porte qu'une boîte de conserve ou une fenêtre ou tout autre agent “ouvrable”. C'est essentiellement ce découplage entre le code de l'interaction et les agents auxquels elle s'applique qui favorise la réutilisabilité de ces éléments de connaissance entre différentes simulations.

Cependant cette généricité est mise à mal lorsque l'on considère que certaines cibles peuvent imposer des conditions particulières pour une même interaction, ou plutôt pour un même concept exprimé par une interaction. Pour poursuivre l'exemple précédent, considérons le cas de portes qui peuvent être cadenassées. Celles-ci peuvent bien sûr toujours subir l'interaction *ouvrir* mais pour pouvoir exécuter cette interaction, il est nécessaire qu'une condition supplémentaire soit vérifiée : la porte ne doit pas être cadenassée. Cependant, conceptuellement, le plan d'un acteur devant franchir cette porte sera fondamentalement le même : il doit *ouvrir* la porte. Ce n'est que le traitement de cette ouverture qui changera avec la nature de la porte selon qu'elle est cadenassable ou pas. Afin de garder le découplage entre les interactions et les agents ainsi que la généricité du moteur, créer une nouvelle interaction comme *ouvrir-si-cadenassable* ne paraît pas pertinent, ni judicieux. C'est pourquoi notre proposition consiste en la possibilité pour les cibles de spécifier par extension des conditions spécifiques. Lors de la construction du plan de l'acteur, la

cible “annonce” à l’acteur les contraintes supplémentaires. Pour l’acteur il n’y a toujours qu’une seule interaction générique. Par exemple, dans le cas d’une *porte-cadenassable* l’interaction *open* voit ses conditions enrichies de la condition *target.isLocked = false*. Ainsi lorsqu’un acteur a pour objectif d’*ouvrir une porte*. Il récupère la condition qui apparaît à la figure 3.5 pour une porte “simple”, mais récupère la condition *target.isOpen = false et target.isLocked = false* dans le cas d’une *porte-cadenassable*. Cette condition amènera le cas échéant l’agent acteur à chercher à décadenasser cette porte avant de l’ouvrir.

Ce mécanisme offre aux concepteurs de la simulation la possibilité d’ajouter de nouvelles cibles qui en spécifient une existante, prenant ainsi en compte certaines particularités du monde simulé sans remettre en cause l’existant. Cette extension ne nécessite en effet pas non plus que soient modifiés les acteurs éventuels, ou plutôt leur moteur de comportements. Cette flexibilité facilite la conception des simulations et favorise la réutilisation des interactions et par conséquence des agents d’une simulation à une autre.

L’article (Mathieu, Picault, and Routier 2003) présentait cette notion qui était également le thème du mémoire de DEA de Patrick Tessier (Tessier 2002).

3.5 Les agents

Comme je l’ai écrit précédemment nous distinguons dans nos simulations les agents *inanimés*, qui correspondent aux objets des mondes simulés, des agent *animés* qui sont les acteurs de ces mondes. Dans les deux cas, ils sont définis par leurs caractéristiques (nom, couleur, énergie, etc.) et les interactions qu’ils peuvent subir. Ce qui les distinguent c’est que les acteurs animés sont en plus définis par les interactions qu’ils peuvent effectuer et disposent d’un moteur de comportements qui leur permet d’accomplir leurs objectifs. Le cycle de fonctionnement de l’agent est le suivant :

1. perception de son environnement, c’est la phase de prise d’information dans un environnement situé ;
2. mise à jour de la base de connaissance ;
3. choix d’une action exécutable, cela implique le calcul d’un plan et la sélection d’une action proposée par ce plan ;
4. exécution de l’action choisie dans l’environnement.

Différents éléments (cf. Figure 3.6) composent donc ce moteur de comportements.

Le *module de perception* permet à l’agent de percevoir son environnement. Ces informations sont transmises au *module de mise à jour* dont la fonction est de prendre en compte l’impact de ces informations. Notamment c’est ce module qui met à jour la base de connaissances de l’agent. Celle-ci comprend les interactions que l’agent peut effectuer, mais surtout sa *mémoire*. Cette mémoire correspond à l’image qu’a l’agent de son environnement. Il s’agit donc d’une version dégradée, car incomplète et non nécessairement à jour, de cet environnement, y compris des autres agents qu’il contient. C’est sur cette mémoire que se base le *moteur de planification* pour établir les plans d’action destinés à lui permettre d’accomplir ses objectifs. Le moteur assure que ces plans sont corrects en accord avec la mémoire de l’agent. Cependant, ces plans peuvent être remis en cause par les informations apportées par le module de perception. Dans ce cas une

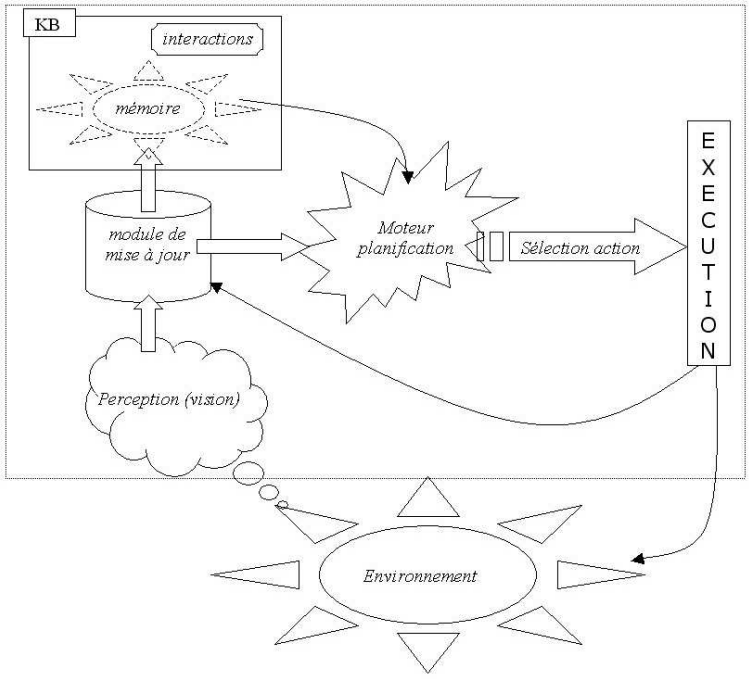

Figure 3.6: Les différents éléments du moteur comportemental.

replanification partielle des plans est réalisée allégeant le travail de planification. Un *mécanisme de sélection d'action* propose ensuite la prochaine action à exécuter par l'agent. Ce choix dépend de priorités de l'agent et du contexte dans lequel il évolue. Si nos agents sont proactifs, cela ne les empêchent en effet pas de manifester des comportements réactifs. L'agent essaie ensuite d'exécuter l'action proposée dans l'environnement. Cela peut s'avérer impossible si la mémoire de l'agent ne correspondait pas à l'état réel de l'environnement. Dans ce cas, le module de mise à jour prendra en compte la nouvelle information et la répercutera.

Le moteur de planification fonctionne de manière assez classique selon un mécanisme type chaînage arrière sur les interactions. Les difficultés principales naissent de la nécessité de prendre en compte l'aspect situé des simulations c'est-à-dire d'inclure les déplacements dans le plan. J'ai déjà évoqué cette problèmatique précédemment lors de la discussion sur les plans abstraits et concrets. Un problème supplémentaire à prendre en compte est la nécessité pour l'acteur de trouver les cibles qui lui permettront d'accomplir les actions que lui commande son plan. Ainsi, grâce aux interactions qu'il peut effectuer, il peut connaître l'existence a priori de telle cible sans savoir nécessairement où elles se trouvent dans l'environnement. A nouveau on voit apparaître une spécificité due au type des simulations visées et notamment à leur aspect situé.

Ces travaux ont été publiés dans (Devigne, Mathieu, and Routier 2005b).

3.6 L'environnement de conception de simulations

Toujours soucieux de mettre en application les concepts que nous proposons, et cela semble encore plus indispensable dans cette thématique, nous avons développé un environnement de conception ("IDE") et d'exécution de simulations (cf.Figure 3.7).

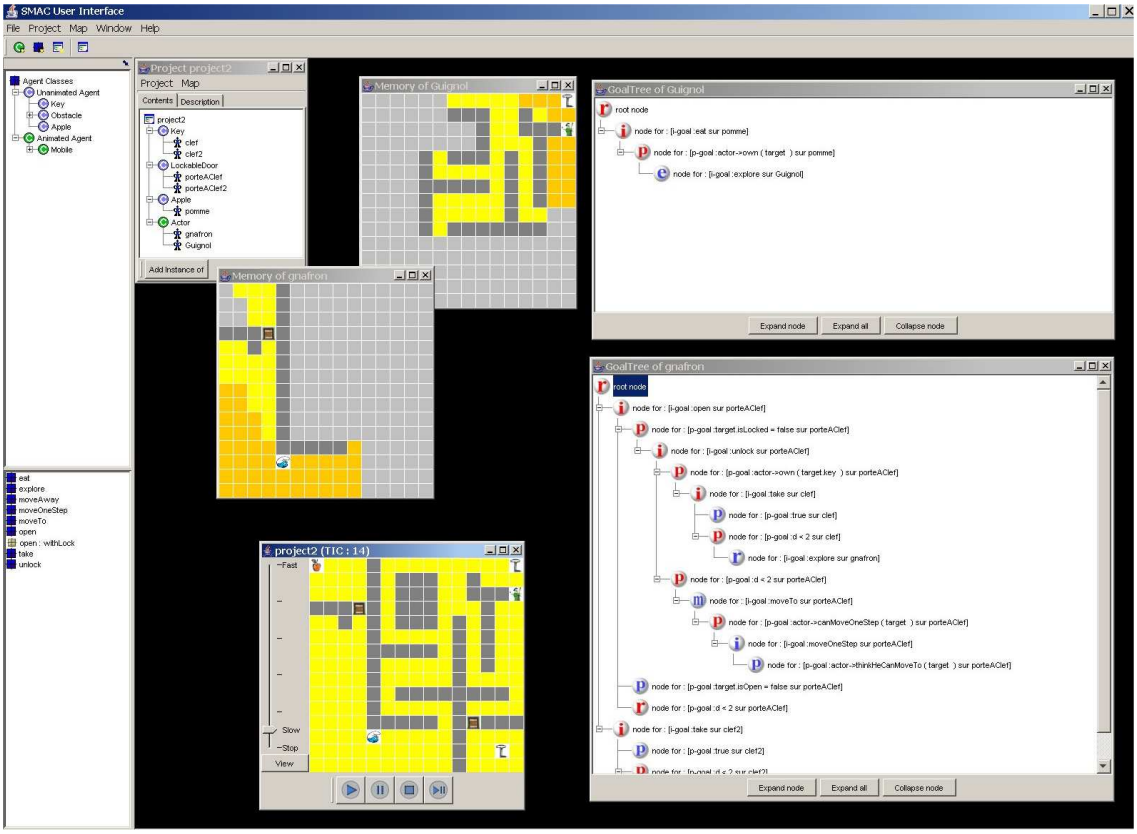

Figure 3.7: Environnement de conception d'applications de CoCoA : on y voit sur la gauche les classes d'agent (en haut) et interactions (en bas) déjà créées. La zone centrale montre une simulation en cours d'exécution. L'environnement y apparaît ainsi que les deux fenêtres montrant la mémoire de chacun des agents actifs avec sa vision partielle de cet environnement. Les arbres de raisonnement des agents sont également visibles.

Celui-ci permet aux concepteurs de simulations de créer des bibliothèques d'interactions et d'agents qui pourront être réutilisés de simulations en simulations. Pour les agents il s'agit plus de patrons (pour ne pas dire classes) partiellement initialisés, ceux-ci peuvent alors être "instanciés" pour créer les agents qui prennent effectivement part aux simulations. Il est également possible de concevoir l'environnement géographique de la simulation et d'y placer les agents auxquels on affecte des buts.

Une fois créée une telle simulation peut être exécutée. Il est alors possible pour chacun des agents animés de suivre la progression de son raisonnement. Ainsi, la mémoire de l'agent peut être observée et son plan d'action visualisé sous forme arborescente. Le concepteur de la simulation peut également intervenir pour modifier telle ou telle valeur dans l'environnement afin

d'étudier l'impact de cette modification sur le comportement des agents. Cet environnement constitue donc également un contexte de tests pour les simulations et les comportements créés.

Il existe d'autres plate-formes de réalisation de simulations centrées individus. On peut citer au niveau international SWARM (Swarm 1994-2005), REPAST (Repast 2003) et au niveau national MADKIT (Ferber, Gutknecht, and Michel 2000) et CORMAS (Bousquet, Bakam, Proton, and Page 1998). Cependant leurs objectifs ou approches diffèrent de la notre. En particulier, le plus souvent, ces plate-formes ne proposent pas (ou n'imposent pas) un modèle d'agents, c'est par exemple le cas de SWARM ou MADKIT. Elles fournissent plutôt un ensemble d'outils ou de primitives facilitant la mise en place de simulations mais laissent l'utilisateur définir son modèle d'agent. Leur approche est en ce sens proche de celle que nous avons adoptée dans le projet MAGIQUE. Ensuite CORMAS, par exemple, a des objectifs différents. Cet environnement a été conçu pour modéliser la gestion des ressources renouvelables et décrire les coordinations entre individus ou entre groupes exploitant des ressources communes. Ainsi notre plate-forme, et plus généralement notre projet, se différencie de ceux qui viennent d'être cités dans la mesure où il vise la modélisation de comportements et propose (et impose) pour cela un modèle d'agent.

3.7 Equipes d'agents

Les travaux décrits précédemment permettent la réalisation de simulations impliquant plusieurs agents évoluant en concurrence dans leur environnement, ces agents cohabitant sans coopérer. C'est pourquoi, de manière naturelle, nous avons été amenés à nous pencher sur la possibilité d'intégrer la notion d'équipe d'agents dans nos simulations. Ces travaux constituent l'un des axes de la thèse en cours de Damien Devigne que je co-encadre avec Philippe Mathieu.

Ayant toujours à l'esprit notre thématique principale d'application que sont les jeux vidéos, notre premier point de recherche a été de considérer des équipes avec chef. Cette configuration se retrouve en effet assez souvent dans les jeux vidéos. Cependant nous ne voulions pas modéliser des équipes au raisonnement totalement centralisé. En effet, les agents impliqués dans ces équipes correspondent à ceux décrits précédemment, c'est-à-dire à des agents proactifs, disposant de capacités de raisonnement. Ainsi il n'était pas question pour nous de réduire nos équipes à un chef qui décide de la moindre action de ses équipiers dont le rôle se réduirait à celui d'exécutants décérébrés.

De plus, nous souhaitions évidemment conserver la représentation des connaissances et notre modélisation à base d'interactions. Or dans ce cadre, nous disposons d'un moteur de comportements qui fonctionne de manière satisfaisante pour gérer les plans d'action individuel.

Notre problème a donc été : comment adapter notre moteur de planification pour produire des plans d'équipe dans lesquels les agents gardent une part d'autonomie.

La solution que nous avons trouvée découle principalement du principe que nous pourrions énoncer comme ceci : "Le chef a besoin de savoir ce que ses équipiers savent faire mais pas nécessairement de savoir comment ils le font". Ainsi le chef d'une équipe doit pouvoir affecter à ses agents des ordres d'assez haut niveau, charge à eux de les résoudre en fonction de leurs connaissances propres. Pour reprendre un exemple cher à Damien Devigne, dans le cadre de la construction d'une maison, un chef de chantier commandera à son électricien d'installer telle prise à tel endroit, mais ne lui dira pas qu'il doit dénuder les fils pour le faire. C'est en effet au chef de savoir où il faut installer les prises, mais comment celles-ci doivent être installées relève de la compétence de l'électricien. Le chef n'a même pas à avoir a priori cette connaissance.

Pour construire nos plans d'équipe, la réalisation de ce principe est obtenu en réutilisant exactement le moteur de planification individuel, mais en ne confiant au chef qu'une partie de la connaissance de ses équipiers. Cette connaissance correspond à la description en termes de fonctionnalités, et donc d'interactions, des rôles des équipiers. Ces interactions sont cependant un peu particulières, non pas dans leur structure puisqu'elles obéissent totalement au schéma présenté, mais dans leur contenu puisque leurs parties condition et garde sont masquées au chef, qui n'en voit donc que la partie action, c'est-à-dire le résultat. Cela revient au fait que le chef sait ce que l'interaction de son équipier peut amener (l'action) mais pas ce qu'il faut pour la réaliser (les conditions). Cette partie masquée est remplacée simplement par la valeur *vrai*, amenant naturellement le moteur de planification à arrêter son chaînage.

Ainsi lors de sa planification d'équipe, le plan du chef s'arrête sur des tâches correspondant à des compétences d'équipier. Le chef distribue alors les tâches aux équipiers qui les réalisent, ce qu'ils peuvent faire puisqu'ils disposent quant à eux de la connaissance dans son ensemble.

Cette solution permet dont la modélisation d'équipes dirigées par un chef qui est en charge d'établir les grandes lignes du plan de l'équipe et de distribuer les tâches à ses équipiers. Ceux-ci exploitent leurs capacités de raisonnement pour traiter de manière autonome ces objectifs. Ce travail a donné lieu aux publications (Devigne, Mathieu, and Routier 2005c, Devigne, Mathieu, and Routier 2005a). Il constitue la thématique principale de la thèse en cours de Damien Devigne (Devigne 2003-en cours) que je co-encadre.

Ce projet est encore en cours. Nous étudions notamment également la mise en place de comportements d'équipe sans chef. Au niveau du moteur comportemental, nous avons récemment travaillé sur le mécanisme de sélection d'action. Nous travaillons sur un modèle permettant à la fois d'introduire des comportements de type réactifs tenant compte de l'environnement immédiat sans perdre de vue les objectifs à plus long terme. Le mécanisme que nous cherchons à mettre en place permet également de simuler des caractères différents d'un agent à un autre. Ces travaux non encore publiés ont été commencés dans le cadre du Master Recherche de Julien Acroute (Acroute 2005).

Bibliographie

Acroute, J.. 2005. "Apport réactif aux comportements cognitifs de la plateforme de simulation CoCoA," Master's thesis, Master Recherche mention Informatique de l'Université de Lille 1, Co-dirigé avec le Professeur Philippe Mathieu.

Bousquet, F.; I. Bakam; H. Proton; and C. L. Page. 1998. "Cormas: common-pool resources and multi-agent Systems," in *Lecture Notes in Artificial Intelligence*, vol. 1416, 826–838.

Brooks, R. A.; and al.. 1999. "The COG Project: Building a Humanoid Robot," in *Computation for Metaphors, Analogy, and Agents*, vol. 1562 of *LNCS*, 52–87. Springer.

Deffuant, G.; N. Ferrand; S. Bernard; and D. Azembourg. 1999. "Modèles multi-agents de la diffusion de l'adoption de mesures agri-environnementales dans des réseaux sociaux d'agriculteurs : décision multicritère et abstraction décroissante des interactions sociales," in *Actes des JFI-ADSMA'99*.

Devigne, D.. 2003. “Simulation de comportements pour agents rationnels situés et étude du GraphPlan,” Master’s thesis, DEA d’Informatique de l’Université de Lille 1, Co-dirigé avec le Professeur Philippe Mathieu.

Devigne, D.. 2003-en cours. “Modélisation de comportement d’équipes en environnement Multi-Agents,” Ph.D. thesis, Université des Sciences et Technologies de Lille, Co-dirigée avec le Professeur Philippe Mathieu.

Devigne, D.; P. Mathieu; and J.-C. Routier. 2004. “Planning for Spatially Situated Agents,” in *Proceedings of IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT’04)*, ed. by I. Press, 385–388.

Devigne, D.; P. Mathieu; and J.-C. Routier. 2005a. “Gestion d’équipes et autonomie des agents,” in *actes des JFSMA2005*, ed. by Cépaduès.

Devigne, D.; P. Mathieu; and J.-C. Routier. 2005b. “Interaction-Based Approach For Game Agents,” in *Proceedings of ECMS/SCS/IEEE 19th European Conference on Modelling and Simulation. ECMS 2005*, ed. by Y. Merkuryev, R. Zobel, and E. Kerckhoffs, 7056714.

Devigne, D.; P. Mathieu; and J.-C. Routier. 2005c. “Teams of cognitive agents with leader: how to let them some autonomy,” in *Proceedings of IEEE Symposium on Computational Intelligence Games. CIG’05*, ed. by G. Kendall, and S. Lucas, 256–262.

Ferber, J.; O. Gutknecht; and F. Michel. 2000. “The MadKit Agent Platform Architecture,” Discussion paper, W3C.

Laird, J.; and J. Duchi. 2000. “Creating human-like synthetic characters with multiple skill levels: A case study using the soar quakebot,” .

Laird, J. E.; and M. van Lent. 2000. “Human-level AI’s Killer Application: Interactive Computer Games,” in *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence.*, 1171 – 1178. AAAI.

Laird, J. L. J.; and P. Rosenbloom. 2005. “A Gentle Introduction to Soar: 2005 Update,” unpublished, An update to an earlier paper (published in 1996).

Mathieu, P.; S. Picault; and J.-C. Routier. 2003. “Simulation de comportements pour agents rationnels situés,” in *Actes des Secondes Journées Francophones sur les Modèles Formels de l’Interaction, MFI’03*, ed. by A. Herzig, B. Chaib-draa, and P. Mathieu, 277–282.

Nareyek, A.. 1998. “Specification and development of reactive systems,” in *1998 AIPS Workshop*, 7–14, Menlo Park California. AAAI Press.

Nareyek, A.. 2000. “Intelligent Agents for Computer Games,” in *Computers and Games, Second International Conference, CG 2000. LNCS 2063.*, 414–422.

Nareyek, A.. 2004. “Artificial Intelligence in Computer Games - State of the Art and Future Directions,” *ACM Queue*, 1(10), 58–65.

Ponsen, M.; and P. Spronck. 2004. “Improving Adaptive Game AI with Evolutionary Learning.,” in *Proceedings of Computer Games: Artificial Intelligence, Design and Education (CGAIDE 2004)*, ed. by S. N. Quasim Mehdi, Norman Gough, and D. Al-Dabass, 389–396.

Querrec, R.; P. Reignier; and P. Chevaillier. 2001. “Humans and autonomous agents interactions in a virtual environment for fire-fighting training,” in *Proceedings of Virtual Reality International Conference (VIRC 2001)*, 57–64.

Repast. 2003. “The Recursive Porous Agent Simulation Toolkit,” `http://repast.sourceforge.net`.

Swarm. 1994-2005. “Swarm Software,” `http://www.swarm.org`.

Tessier, P.. 2002. “Planification et exécution dans un environnement non monotone,” Master’s thesis, DEA d’Informatique de l’Université de Lille 1, Co-dirigé avec Professeur Philippe Mathieu.

Tozour, P.. 2002. “The Perils of AI Scripting,” in *AI Game Programming Wisdom*, ed. by S. Rabin, 541–547. Charles River Media.

Partie II

Sélection d’articles

Partie III

Annexe

Mes activités d'enseignant-chercheur

Jean-Christophe Routier né le 4 juin 1968, marié, 3 enfants

Diplômes

– DEA Intelligence Artificielle et Applications, Université de Caen, 1990, mention bien.

– Ingénieur de l'ENSI de Caen, option Intelligence Artificielle, 1990.

– Docteur en Informatique de l'Université des Sciences et Technologies de Lille, thèse dirigée au LIFL par Philippe Devienne et Patrick Lebègue et soutenue le 1er février 1994, mention très honorable.
Terminaison, calculabilité, pouvoir calculatoire d'une clause de Horn binaire

Situation Actuelle

Maître de conférences depuis septembre 1994 à l'UFR d'IEEA de l'Université des Sciences et Technologies de Lille.

Membre de l'équipe SMAC (Systèmes Multi-Agents et Coopération) du Laboratoire d'Informatique Fondamentale de Lille – CNRS UMR 8022

Dans cette section je vais brièvement présenter mon activité dans les différentes facettes de mon travail d'enseignant-chercheur. J'essaierai ainsi de démontrer mon implication dans les trois domaines :

- *Enseignement* où je décrirai les enseignements effectués ainsi que la partie gestion de formation
- *Administration* où je citerai les charges collectives auquel j'ai pris ou prends part
- *Recherche* où, dans la mesure où les premières parties de ce document constituent une synthèse de mes activités depuis mon doctorat, je citerai mon activité en terme d'encadrement, participation à des projets et animation.

Je finirai par une énumération de mes publications rangées par catégories.

Enseignement

Enseignements devant étudiants

Je ne vais pas énumérer ici tous les enseignements que j'ai effectués depuis ma nomination en 1994, ce serait fastidieux pour le lecteur. Pour les résumer, je peux dire que j'ai effectué mes interventions (cours magistraux, TD ou TP) dans toutes les années du cycle universitaire : Bac+1 à Bac+5. Les formations où je suis intervenu sont également diversifiées, professionnalisées ou non. Il s'agit principalement de : DEUG MIAS, IUP GMI (seconde et troisième année), DESS (surtout IAGL), Licence Informatique, DEA d'Informatique. Actuellement mes enseignements se concentrent sur les semestres S4-S5-S6 de la licence mention Informatique de Lille 1. L'essentiel de mes enseignements peut se ranger dans les rubriques "Programmation/Conception de Logiciel" (de l'initiation à l'approfondissement) et "Intelligence Artificielle". J'ai eu également l'occasion au cours de ces années d'encadrer, aux niveaux Bac+4 et Bac+5, de nombreux projets d'étudiants ainsi que des stages en entreprise.

Je peux également parler rapidement de quelques enseignements dont je suis (ou étais) responsable et que j'ai créés.
D'abord la mise en place au début de ma nomination des enseignements d'initiation à la programmation lors de la création du DEUG MIAS. Jusque là, il n'y avait pas d'enseignement d'Informatique en première année de DEUG à Lille 1. Ce travail a débouché sur la rédaction d'un ouvrage (cf. *Publications*).
Ensuite les cours de "Programmation Orientée objet" et "Conception Orientée Objet" que j'ai créés en 2001-2002 lorsque le paradigme de programmation objet a été choisi comme approche principale de la programmation en licence d'Informatique. Dans la suite de ces cours, je suis également responsable de l'Unité d'Enseignement "Projet Logiciel" de la licence mention Informatique.
J'ai également créé cette année avec Erci Wegrzynowski l'Unité d'Enseignement ELFE de la licence S5 dont l'objectif est de présenter les paradigmes logique et fonctionnel.

Administration de l'enseignement

Au cours de ces années j'ai eu l'occasion de participer à différents niveaux à la gestion de formations professionnalisées ou non.

– Président de jury d'une section de DEUG MIAS, 1995-1996.

– Responsable du DESS Intelligence Artificielle et Génie Logiciel (IAGL) de 1997 à 2002.

  - présidence de jury,
  - sélection/recrutement des étudiants,
  - gestion, coordination des enseignements et de l'équipe pédagogique,
  - gestion des projets et stages,
  - rédaction de la maquette d'habilitation de la formation (1998 et 2001)

– Membre du Groupe de Proposition de Formation "Mathématique-Informatique-Physique-Mécanique" à l'occasion de la mise en place du LMD à Lille 1 (2003-2004).

– Co-directeur des études des semestres S4-S5-S6 la mention Informatique de la licence Sciences et Technologies A de Lille 1 depuis 2004.

- organisation des enseignements, gestion de l'équipe pédagogique,
- relation avec les étudiants,
- rédaction de la maquette d'habilitation de la formation (2004 et 2005),
- gestion/animation des Comissions Pédagogiques Paritaires (CPP).

Diffusion de la connaissance

– rédaction et diffusion de documents de cours en ligne,

– rédaction d'un livre sur l'initiation à la programmation (cf. *Publications*).

Charges Collectives

Il s'agit ici des charges collectives non directement liées à l'enseignement car celles-ci ont déjà été présentées dans la rubrique précédente.

– Membre suppléant de la CSE 27ème section de l'USTL de 2000 à 2003.

– Membre titulaire de la CSE 27ème section de l'USTL depuis 2003

– Membre du conseil de l'UFR d'IEEA de l'USTL depuis 2002.

– Responsable de la gestion des services pour l'Informatique depuis 2003.

  Dans ce cadre j'ai développé un logiciel permettant un gestion plus rationnelle des services via une "interface web". Ce logiciel est également déployé dans la partie EEA de l'UFR cette année.

– Responsable et coordinateur pour l'Informatique des dossiers et opérations "Contenu Numérique en Ligne" de l'Université de Lille 1, en 2004 et 2005.
  Un des résultats de cette action est le portail de présentation de la mention Informatique de la licence `http://www.fil.univ-lille1.fr/Licence`.

Recherche

Les premières parties de ce mémoire constituant une synthèse de mes travaux de recherche depuis l'obtention de mon doctorat, je ne vais pas les résumer à nouveau ici. Je citerai simplement les éléments importants en terme d'animation, d'encadrement et publications.

Contrat

– Collaboration avec la société Cryo Interactive à travers un projet du programme PRIAMM (Programme pour l'Innovation dans l'Audiovisuel et le Multimédia), janvier 2000 à juin 2001.

Organisation de conférences

– Membre du comité d'organisation des Journées Francophones de l'Intelligence Artificielle Distribuée et des Systèmes Multi-Agents (JFIADSMA) 2002. Lille, 27-30 octobre 2002. Chargé de la gestion des inscriptions.

– Membre du comité d'organisation des Journées sur les Modèles Formels de l'Interaction (MFI) 2003. Lille, 20-22 mai 2003. Chargé de la gestion des inscriptions.

Participations à des groupes et projets de recherche

– membre des groupes ASA et MFI du GDR I3.

– membre d'AgentLink à travers l'équipe SMAC,

– participation aux CPER (Contrat de Plan Etat-Région) successifs dans lesquels l'équipe était ou est actuellement impliquée : d'abord le projet *Ganymède* puis ses successeurs *COLORS* ("Composants Logiciels Réutilisables et Sûrs"), *NIPO* ("Nouvelles Interactions Personnes-Organisations") et *Formasciences* sur les nouveaux usages, et actuellement *MIAOU* ("Modèles d'Interaction et Architectures Orientées Usages") et *MOSAIQUES* ("MOdèles et InfraStructures pour Applications ubIQUitairES").

Encadrement

Mémoire de DEA/Master Recherche

– Yann Secq, "*Notion de service et d'échange de services dans les systèmes multi-agents*". DEA d'Informatique de l'Université de Lille 1. Co-dirigé avec le Professeur Philippe Mathieu. 1999.

– Patrick Tessier, "*Planification et exécution dans un environnement non monotone*". DEA d'Informatique de l'Université de Lille 1. Co-dirigé avec Professeur Philippe Mathieu. 2002.

– Damien Devigne, "*Simulation de comportements pour agents rationnels situés et étude du GraphPlan*". DEA d'Informatique de l'Université de Lille 1. Co-dirigé avec le Professeur Philippe Mathieu. 2003.

– Julien Acroute, "*Apport réactif aux comportements cognitifs de la plateforme de simulation CoCoA*". Master Recherche mention Informatique de l'Université de Lille 1. Co-dirigé avec le Professeur Philippe Mathieu. 2005.

Thèse

– Yann Secq, "*RIO : Rôles, Interactions et Organisations, une méthodologie pour les systèmes multi-agents ouverts*". Co-dirigée avec le Professeur Philippe Mathieu. Soutenue le 2 décembre 2003.

– Damien Devigne, "*Modélisation de comportement d'équipes en environnement Multi-Agents situés*". Co-dirigée avec le Professeur Philippe Mathieu. 2003-en cours.

Publications

Livre

– "*Débuter la Programmation avec* SCHEME" Jean-Christophe Routier et Éric Wegrzynowski. International Thomson Publishing France. 349 pages. 1ère édition 1997. seconde édition mai 2003.

Revues internationales

– "*Smallest Horn clause programs*" Philippe Devienne, Patrick Lebègue, Anne Parrain, Jean-Christophe Routier et Jörg Würtz (DFKI, Saarbrücken), in Journal of Logic Programming, Vol. 27(3), pp. 227-267. 1996.

– "*Using agents to build a distributed calculus framework*" Philippe Mathieu, Jean-Christophe Routier et Yann Secq, in The Interdisciplinary Journal of Artificial Intelligence and the Simulation of Behaviour vol. 1 number 2, pp. 197-208. June 2002.

Revues francophones

– "*Une contribution du multi-agent aux applications de travail coopératif*" Philippe Mathieu et Jean-Christophe Routier in TSI Hermès Science Publication. Réseaux et Systèmes Répartis. Calculateurs Parallèles. Volume 13. Numéro Spécial Télé-Applications, pp. 207-226. 2001.

– "*Les agents intelligents*" Philippe Mathieu, Sébastien Picault et Jean-Christophe Routier. "Pour la Science" Numéro 332. Juin 2005, pp. 44-52.

Conférences internationales avec actes

– "*The halting problem of one binary Horn clause is undecidable*" Philippe Devienne, Patrick Lebègue et Jean-Christophe Routier. oin P. Enjalbert, A. Finkel and K.W. Wagner (Eds), Proceedings of STACS'93, Würzburg. LNCS 665, pp. 48–57. 1993.

– "*The emptiness problem of one binary Horn clause is undecidable*" Philippe Devienne, Patrick Lebègue et Jean-Christophe Routier. in Dale Miller (Ed), proceedings of 1993 International Symposium on Logic Porgramming, ILPS'93, Vancouver, pp. 250–265. 1994.

– "*One binary Horn clause is enough*" Philippe Devienne, Patrick Lebègue, Jean-Christophe Routier et Jörg Würtz (DFKI, Saarbrücken). in P. Enjalbert, E.W. Mayr and K.W. Wagner (Eds), proceedings of STACS'94, Caen. LNCS 775, pp. 21–32. 1994.

– "*Dynamic Skill Learning: A Support to Agent Evolution*" Philippe Mathieu, Jean-Christophe Routier et Yann Secq, in Proceedings of the AISB'01 Symposium on Adaptive Agents and Multi-Agent Systems, York, ISBN 1 902956 17 0, pp. 25–32. 2001.

– "*A Multi-Agent Approach to Co-operative Work*" Philippe Mathieu et Jean-Christophe Routier, in C. Kolski and J. Vanderdonckt (Eds), Proceedings of the Computer Aided Design of User Interfaces, CADUI'02, Valenciennes, pp. 367-380. 2002.

– "*RAGE: An agent framework for easy distributed computing*" Philippe Mathieu, Jean-Christophe Routier et Yann Secq, in Proceedings of the AISB'02 Symposium on Artificial Intelligence and Grid Computing, ISBN 1 902956 24 8, London, pp. 20-24. 2002.

– "*Principles for dynamic multi-agent organizations*" Philippe Mathieu, Jean-Christophe Routier et Yann Secq , in K. Kuwabara and J. Lee (Eds), proceedings of the Fifth Pacific Rim International Workshop on Multi-Agents, PRICAI2002-PRIMA2002, Tokyo, Spinger-Verlag LNAI vol. 2413, pp. 109-122. 2002.

– "RIO *: Roles, Interactions and Organizations*" Philippe Mathieu, Jean-Christophe Routier et Yann Secq. in V. Marik and J. Müller and M. Pechoucek (Eds), the Proceedings of the 3rd International/Central and Eastern European Conference on Multi-Agent Systems, CEEMAS 2003, Prague. LNAI 2691, pp. 147-157. June 2003.

– "*Bridging the gap between semantic and pragmatic*" Philippe Mathieu, Jean-Christophe Routier et Yann Secq. in H.R. Arabnia (Ed), the Proceedings of The 2003 International Conference on Information and Knowledge Engineering, IKE'03, vol. I, Las Vegas, pp. 308-314. June 2003.

– "*Runnable specifications of interactions protocols for open multi-agent systems*" Philippe Mathieu, Jean-Christophe Routier et Yann Secq. in Nazli Goharian (Ed), the Proceedings of the 2003 International Conference on Information and Knowledge Engineering, IKE'03, vol. II, Las Vegas, pp. 431-437. June 2003.

– "*Towards a Pragmatic Methodology for Open Multi-agent Systems*" Philippe Mathieu, Jean-Christophe Routier et Yann Secq in N. Zhong, Z. W. Raś, S. Tsumoto and Einoshin Suzuki (eds) , the "Foundations of Intelligent Systems". 14th International Symposium on Methodologies for Intelligent Systems, ISMIS 2003, Maebashi (Japon). Springer-Verlag, LNAI 2871, pp. 206-210. 2003.

– "Planning for Spatially Situated Agents" Damien Devigne, Philippe Mathieu et Jean-Christophe Routier. in the Proceedings of IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'04), Beijing (Chine). IEEE Press, pp. 385-388. 2004.

– "Team of cognitive agents with a leader : how to let them acquire autonomy" Damien Devigne, Philippe Mathieu et Jean-Christophe Routier. in G. Kendall and S. Lucas (Eds), the Proceedings of IEEE Symposium on Computational Intelligence and Games, CIG 2005, York, pp. 256-262. 2005.

– "*Interaction-Based Approach For Game Agents*" Damien Devigne, Philippe Mathieu et Jean-Christophe Routier. in Y. Merkuryev and R. Zobel and E. Kerckhoffs (Eds), the Proceedings of ECMS/SCS/IEEE 19th European Conference on Modelling and Simulation. ECMS 2005, Riga, pp. 705-714. 2005.

Conférences francophones avec actes

– "*RIO: Rôles, Interactions et Organisations*" Philippe Mathieu, Jean-Christophe Routier et Yann Secq, in A. Herzig and B. Chaib-draa and P. Mathieu (Eds), actes des Secondes Journées Francophones sur les Modèles Formels de l'Interaction, MFI'03, Lille, pp. 179-188. Mai 2003.

– "*Simulation de comportements pour agents rationnels situés*" Philippe Mathieu, Sébastien Picault et Jean-Christophe Routier. in A. Herzig and B. Chaib-draa and P. Mathieu (Eds), actes de Modèles Formels des Interactions, MFI'03, Lille, pp. 277-282. mai 2003.

– "*Gestion d'équipes et autonomie des agents*" Damien Devigne, Philippe Mathieu et Jean-Christophe Routier, Calais, actes des JFSMA2005. 2005.

Autres publications

– "*Weighted Sytems of Equations revisited*" Philippe Devienne, Patrick Lebègue et Jean-Christophe Routier. in actes du Workshop on Static Aanalysis WSA'92, Bordeaux, pp. 163–173. 1992.

– "*The halting problem of one binary Horn clause is undecidable*" Philippe Devienne, Patrick Lebègue et Jean-Christophe Routier. "Workshop on termination" à l'occasion de JICSLP'92, Washington. 1992.

– "Tutoriel de l'API MAGIQUE", Jean-Christophe Routier `http://www.lifl.fr/SMAC` rubrique `MAGIQUE`. 2000.

– "*Un modèle de simulation agent basé sur les interactions*" Philippe Mathieu, Jean-Christophe Routier et Pascal Urro (Sté Cryo Interactive), in actes de Modèles Formels des Interactions, Toulouse, MFI'01. (poster) 2001.

– "*Dynamic Organization of Multi-Agent Systems*" Philippe Mathieu, Jean-Christophe Routier et Yann Secq, poster, in proceeedings of AAMAS'02, Bologne, pp. 451-452. (poster) july 2002.

– "*Ubiquitous Computing : vanishing the notion of application*" Philippe Mathieu, Jean-Christophe Routier et Yann Secq, poster, in proceedings of the Workshop UbiAgents'02 on Ubiquitous Agents on embedded, wearable, and mobile devices, Bologne, (poster) july 2002.

– "*Towards a pragmatic use of ontologies in multi-agent platforms*" Philippe Mathieu, Jean-Christophe Routier et Yann Secq in the "Ontology and Multi-Agent Systems Design" session (OMASD'03), in the Seventh International Conference on Knowledge-Based Intelligent Information & Engineering Systems, KES'03, Oxford, 2003.

– "*Gestion simple d'équipes d'agents cognitifs*" Damien Devigne, Philippe Mathieu et Jean-Christophe Routier. in the Proceedings of Majestic'2004, Calais. 2004.