THÈSE

de l'Université de Franche-Comté
et de l' Université de Technologie de Belfort-Montbéliard

numéro attribué par la bibliothèque | 0 | 7 | 0 |

pour obtenir le grade de

DOCTEUR

DISCIPLINE : **INFORMATIQUE**

SYSTÈMES MULTI-AGENTS HOLONIQUES : DE L'ANALYSE À L'IMPLANTATION.

Méta-modèle, méthodologie, et simulation multi-niveaux.

par

Nicolas A. GAUD
Laboratoire Systèmes et Transport
Université de Technologie de Belfort-Montbéliard

Soutenue le 7 décembre 2007 devant le Jury composé de :

Marie-Pierre Gleizes	Rapporteur	Professeur à l'Université de Toulouse III
Juan Pavón Mestras	Rapporteur	Professeur à la Faculté d'Informatique de l'Université de Madrid
Jean-Pierre Müller	Rapporteur	Directeur de Recherche au CIRAD-Montpellier
René Mandiau	Examinateur	Professeur à l'Université de Valenciennes
Yassine Ruichek	Examinateur	Professeur à l'UTBM
Abderrafiâa Koukam	Directeur de Thèse	Professeur à l'UTBM
Stéphane Galland	Co-Directeur de Thèse	Maître de Conférences à l'UTBM

Remerciements

Je tiens tout d'abord à remercier Marie-Pierre Gleizes, Juan Pavón Mestras et Jean-Pierre Müller pour m'avoir fait l'honneur de rapporter sur ce travail. Je sais combien leur temps est précieux, et je leur exprime toute ma gratitude.

Je remercie également René Mandiau et Yassine Ruichek d'avoir accepté de faire partie de mon jury.

Je ne saurais oublier ceux qui ont tant contribuer à ces travaux. Et le premier d'entre eux est Abderrafiâa Koukam, qui fut plus qu'un directeur de thèse pour moi. Sa générosité hors du commun n'a d'égale que son exigence et sa rigueur scientifique, celles-ci même qui l'ont conduit à sacrifier plus d'un week-end ensoleillé et plus d'une soirée d'été pour corriger ce manuscript. La grande liberté dont j'ai bénéficié n'a pas de prix, et j'espère néanmoins que cet ouvrage comblera une partie de mon immense dette à son égard.

Je tiens également à remercier Stéphane Galland. Son amitié et son soutien pendant ces quatres dernières années m'ont apportés plus que je ne l'aurais imaginé en abordant ce travail.

Le *nous* employé tout au long de la thèse englobe dans son anonymat les personnes avec lesquelles ces travaux ont pu aboutir. Il serait sans doute trop long d'en faire la liste exhaustive, mais je tiens à remercier tout particulièrement Sebastian Rodriguez. Sebi, nos interminables discussions sur les holons ont posé les fondements des travaux présentés ici, cette thèse est aussi la tienne.

Un grand merci également à David Meignan (Shalum) et Sana Moujahed qui furent mes compagnons de route durant ces trois années, une route qui fut bien plus agréable à arpenter en leur compagnie.

Merci à toute la bande du monde merveilleux de HolonIA : Mickaël Goncalves (Titi), Renan Zeo (Phil), Vincent Hilaire, Franck Gechter, Olivier Lamotte et Philippe Descamps, avec qui l'informatique et l'intelligence artificielle deviennent philosophie et sciences de l'homme.

Je tiens également à remercier Massimo Cossentino, dont les idées ont imprégné bien des parties de cette thèse. Nous fument plus d'une fois embarqués dans de passionantes discussions et celles-ci ont grandement participé à clarifier ce manuscript.

Merci à celles et ceux qui ont partagé mon quotidien au sein du laboratoire SeT et qui ont fait que ces années se sont écoulées dans la bonne humeur, je pense notamment à Alexandre Gondran, Pablo Gruer, Fabrice Lauri, Davy Monticolo, Sergio Nogueira, Mustapha Oughdi, et Ariane Glatigny.

Sommaire

Sommaire des définitions

Sommaire des Figures

PREMIÈRE PARTIE

Contexte

CHAPITRE 1

INTRODUCTION

1.1 Contexte

La théorie des systèmes complexes a connu récemment un sursaut majeur, et les travaux de recherche sur ces systèmes sont de plus en plus considérés comme une discipline à part entière, transversale à de nombreux domaines scientifiques tels que la physique, la chimie, la biologie, l'informatique, la sociologie ou encore l'économie. La création du Santa Fe Institute[1] en 1984, ou de la Société Européenne des Systèmes Complexes[2] en 2004 ainsi que les différents projets et initiatives soutenus par la commission européenne tels que ONCE-CS[3], FET-CS[4] témoignent de cet intérêt grandissant pour les systèmes complexes et de l'évolution de la recherche dans ce domaine.

Toutes ces recherches inter-disciplinaires s'inscrivent dans la lignée des tentatives d'établissement de *« La théorie générale des systèmes »* ou des visions intégrales, défendues par certains auteurs (ex : [Von Bertalanffy, 1968] ou [Simon, 1996]). L'objectif est de fournir un cadre global pour expliquer la dynamique de systèmes dans un éventail varié de champs de la connaissance.

[Simon, 1996] considère d'ailleurs que *« la complexité prend fréquemment la forme d'une arborescence et que les systèmes arborescents ont quelques propriétés communes qui sont indépendantes de leur contenu spécifique. [...] L'arborescence est un des schémas structurels de base qu'utilise l'architecte de la complexité. Par système hiérarchique (ou arborescent) ou arborescence, il entend un système composé de sous-systèmes inter-reliés, chacun d'entre eux ayant, à son tour, une structure arborescente, cela jusqu'à ce que nous atteignions le plus bas niveau des systèmes élémentaires. Dans la plupart des systèmes que nous rencontrons dans la nature, il y a quelque arbitraire dans la façon dont nous interrompons une partition et dans le choix du sous-système tenu pour élémentaire. »*

Adhérant pleinement au raisonnement de Simon, la thèse défendue dans ce manuscrit

[1] Santa Fe Institute : `http://www.santafe.edu/`

[2] ECSS, European Complex Systems Society : `http://ecss.csregistry.org/tiki-index.php?page=HomePage`

[3] ONCE-CS : `http://complexsystems.lri.fr/Portal/tiki-index.php?page=ONCE-CS`

[4] Complex Systems - Proactive Initiative in the 6th Framework Programme : `http://cordis.europa.eu/ist/fet/co.htm#projects`

s'intéresse à la structure hiérarchique des systèmes complexes et à leur modélisation. Pour ce faire, le paradigme des systèmes multi-agents et celui des systèmes holoniques sont associés pour modéliser les systèmes complexes au moyen d'une hiérarchie d'agents, d'une structure gigogne où les agents sont composés d'agents et s'emboîtent les uns dans les autres.

Cette notion d'agents composés d'agents est modélisée par le concept de *Holon*. Le terme *Holon* résulte de l'association du terme grec *« holos »* qui signifie le *« tout »* et du suffixe *« -on »* qui réfère à l'entité ou la particule comme dans neutron, proton ou électron. Ce terme fut introduit par le philosophe hongrois [Koestler, 1967] pour tenter d'unifier les visions holistique et réductionniste sur le monde. Un holon est un élément qui peut être vu à la fois comme une partie composante d'un élément de niveau supérieur, et comme un tout composé d'autres holons. Par conséquent, la notion de holon est intrinsèquement récursive et permet de décrire naturellement des systèmes de nature hiérarchique. Ce concept a été récemment adopté par la communauté de l'intelligence artificielle distribuée sous le nom de systèmes multi-agents holoniques [Gerber et al., 1999] (désigné par SMAH dans la suite de ce document). Les systèmes multi-agents sont aujourd'hui largement acceptés comme une métaphore efficiente et un paradigme à part entière de modélisation des systèmes complexes. Ils sont en effet très efficaces pour gérer la nature hétérogène des composantes d'un système, pour modéliser les interactions entre ces composantes et tenter de comprendre les phénomènes émergents qui en découlent. Toutefois dans les systèmes multi-agents non holoniques, la notion de hiérarchie est encore relativement peu prise en compte dans les modèles.

Les systèmes multi-agents holoniques peuvent offrir une alternative intéressante à ce problème. L'idée sous-jacente à ce type de systèmes considère que des agents peuvent se regrouper pour créer un agent de niveau supérieur ou qu'un agent peut être décomposé en plusieurs agents de niveau inférieur. Ce processus de composition mutuelle donne naissance à un système hiérarchique. Même si cette idée d'*agent récursif* est présente dans l'esprit de nombreux chercheurs, la plupart des modèles actuels considèrent toujours les agents comme des entités atomiques [Gasser, 1992, Giret and Botti, 2003].

1.2 Première analyse et objectifs de ces travaux

L'objectif principal de cette thèse peut être résumé ainsi :

> Proposer un processus de développement logiciel et les outils associés pour l'analyse, la conception, l'implantation et la simulation de systèmes multi-agents holoniques.

Cet objectif global peut se décomposer en trois sous-objectifs.

Tout d'abord, concevoir un processus de développement passe par l'élaboration d'un métamodèle fournissant l'ensemble des concepts nécessaires à l'analyse et à la conception.

Ensuite, le terme *processus* est généralement entendu comme ayant deux composantes : le processus lui-même et les produits [Rolland et al., 1999]. La description complète du processus de développement et de ses différentes étapes doit être fournie, ainsi que la description des produits pour chaque étape. Décrire les produits implique également de détailler les langages utilisés pour cette description.

Enfin, le troisième volet de la conception d'un processus concerne les outils qui lui sont associés pour assister ses différentes phases. On distingue généralement deux grandes catégories d'outils : les Ateliers de Génie Logiciel (AGL) visant essentiellement à assister les phases d'analyse et de conception, et les plates-formes d'implantation et de déploiement. Ce document se concentre essentiellement sur ce dernier type, mais cette thèse s'inscrit dans un travail de plus grande envergure visant à fournir l'ensemble des outils nécessaires pour assister le processus de développement dans son intégralité.

Les travaux décrits dans cette thèse se placent dans la continuité des travaux de [Hilaire, 2000] qui a proposé une approche de spécification formelle pour les SMA ainsi qu'un premier métamodèle organisationnel. Ce métamodèle a ensuite été étendu par [Rodriguez, 2005] qui proposa un *"framework"* générique pour la conception de systèmes multi-agents holoniques. Cette thèse tente d'unifier ces travaux dans un métamodèle unique qui constitue la base de la conception du processus de développement associé. Le processus en lui-même résulte des travaux menés en collaboration avec Massimo Cossentino, qui est l'un des auteurs des méthodologies PASSI [Cossentino, 2005] et AgilePASSI [Chella et al., 2006] destinées à la modélisation de systèmes multi-agents.

Le cœur des travaux présentés ci-après repose sur le métamodèle CRIO qui fournit les abstractions nécessaires à l'analyse et la conception de systèmes complexes. Ce métamodèle est ensuite intégré dans le processus méthodologique ASPECS afin de faciliter sa mise en œuvre dans le cadre du développement d'applications logicielles considérées comme complexes. Pour mener à bien cette activité de développement, disposer d'une plate-forme d'implantation adaptée aux concepts introduits dans le métamodèle CRIO, est une nécessité. *Janus* est une nouvelle plate-forme spécifiquement conçue par faciliter l'implantation des

modèles fondés sur CRIO et ASPECS. Ces différents sujets constituent les fondements de cette thèse et sont détaillés ci-après.

1.2.1 CRIO : un métamodèle pour l'analyse et la conception de systèmes complexes

Le métamodèle CRIO cherche à exploiter les propriétés hiérarchiques des systèmes complexes pour les analyser et les modéliser. Un métamodèle se doit de fournir un ensemble approprié d'abstractions pour identifier, formuler et décrire un problème ainsi que sa solution éventuelle. Le métamodèle CRIO adopte une approche organisationnelle dont la notion de *comportement* est l'abstraction qui en constitue la base.

Dans un système complexe, ce qui est vu comme un *« tout »* à un niveau d'observation donné peut être décomposé en un ensemble de parties en interaction. En adoptant une approche organisationnelle, le comportement global associé au *« tout »* sera modélisé à l'aide de la notion d'*organisation*, et ses composantes avec le concept de *rôle*. Chaque système peut alors être décomposé niveau par niveau jusqu'à atteindre le niveau d'observation que l'on considère comme élémentaire pour l'étude en cours. De même, chaque niveau du système pourra à son tour être décomposé sous forme d'organisations et de rôles. Le résultat de cette étude est alors constitué d'une hiérarchie organisationnelle à l'image du système étudié. Cette hiérarchie sera ensuite associée à un système multi-agents holonique en charge de lui donner vie et d'exécuter les comportements qui la composent.

Il est nécessaire, pour mener à bien cette activité de décomposition hiérarchique d'un système complexe, de disposer d'un guide méthodologique pour assister le travail de l'ingénieur. Ce guide est fourni par le processus d'ingénierie logicielle ASPECS.

1.2.2 ASPECS : un processus d'ingénierie logicielle pour la modélisation et l'implantation de systèmes complexes

Le métamodèle fournit les abstractions et la philosophie utilisées pour modéliser un problème et sa solution. Le processus décrit les étapes à suivre pour obtenir cette solution. Les résultats intermédiaires, associés à chacune des étapes, doivent également être décrits. ASPECS fournit les moyens nécessaires pour décomposer un système complexe, niveau par niveau, et étudier les relations entre eux. Différents points de vue peuvent être adoptés pour décrire un système, ASPECS tente également de les intégrer dans la conception d'un modèle unique. L'ensemble du processus est illustré par un exemple visant à concevoir un simulateur de robots footballeurs à l'image de ceux utilisés dans la compétition FIRA RoboCup. Il s'agit d'une application relativement célèbre et déjà utilisée pour illustrer certaines méthodologies ou plates-formes telles que Volcano [Ricordel, 2001] par exemple. L'ensemble des diagrammes constituant les produits de chaque étape du processus est illustré sur cet exemple.

Du point de vue de [Dastani and Gomez-Sanz, 2005], les applications multi-agents ne pourront effectivement convaincre les industriels que si on comble le fossé qui sépare d'une part l'analyse et la conception des SMA et d'autre part leur implantation. Combler ce fossé passe d'abord par le développement d'outils qui facilitent une implantation simple, voire directe, des abstractions utilisées dans la phase d'analyse et de conception. Cela implique que les AGLs doivent tendre vers davantage de génération automatique de code et que les plates-formes doivent mettre à disposition du développeur des concepts qui soient plus proches de ceux utilisés durant l'analyse et la conception.

1.2.3 *Janus* : une plate-forme organisationnelle pour l'implantation et la simulation de systèmes complexes

L'approche défendue dans cette thèse vise à réduire l'écart entre le métamodèle de conception et celui de l'implantation. Une telle approche permet de faciliter le processus d'implantation, mais également de réduire le risque d'erreur dû à la transition entre ces modèles tout en conservant pleinement les propriétés avantageuses des concepts utilisés dans la phase de conception. Les concepts utilisés dans l'implantation font partie intégrante du métamodèle et le processus de transition entre le métamodèle de conception et celui d'implantation est directement intégré dans le processus méthodologique. La plate-forme *Janus* a été implantée de manière à fournir directement les concepts à la base de la conception de systèmes multi-agents holoniques. En particulier, les concepts d'*organisation*, de *rôle* et de *holon* sont considérés comme des entités de premier ordre. Un AGL, nommé *Janeiro* et basé sur la spécification formelle proposée dans [Hilaire, 2000, Rodriguez, 2005] est en cours de développement pour tenter d'effectuer l'autre moitié du parcours et ainsi disposer d'une suite complète d'outils. De nombreuses plates-formes multi-agents existent actuellement, mais comme pour les modèles, un tout petit nombre gère la notion d'agent récursif ou de holon. Seules quelques plates-formes organisationnelles intègrent la notion de rôle en tant qu'entité de premier ordre. Cependant, aucune ne gère ces deux aspects en même temps. Cette absence de plates-formes qui allient l'approche holonique et l'approche organisationnelle, constitue notre principale motivation pour le développement d'une nouvelle plate-forme.

En sus de la modélisation des systèmes complexes, cette thèse aborde également les problématiques liées à leur simulation. Les systèmes multi-agents sont appropriés pour concevoir des simulations de systèmes complexes. Cependant, simuler précisément de tels systèmes requiert généralement d'importantes ressources de calcul. Cet aspect peut souvent s'avérer un frein à l'utilisation des SMA dans ce domaine. La simulation multi-agents multi-niveaux permet d'obtenir un compromis entre la précision d'une simulation et les ressources de calcul disponibles. L'approche défendue dans ce manuscrit exploite les propriétés des systèmes multi-agents holoniques pour concevoir des modèles de simulation

multi-agents multi-niveaux.

1.2.4 Exploitation du modèle holonique pour la simulation multi-niveaux

Le problème de la simulation multi-niveaux demeure ouvert dans le domaine de la simulation de systèmes complexes. Son objectif consiste à moduler dynamiquement le niveau de complexité d'une simulation en fonction de certaines contraintes. Pour y parvenir, l'approche proposée repose sur l'adaptation dynamique de la complexité du comportement des entités simulées. Ce type de simulation s'adresse avant tout à des systèmes de grande échelle impliquant un grand nombre d'entités. Son principe vise à faire cohabiter, au sein de la même simulation, des entités simulées à des niveaux d'abstraction différents : niveaux sub-microscopique, microscopique (individu-centré), mésoscopique ou macroscopique. La même entité peut ainsi changer dynamiquement de niveau de comportement en fonction des contraintes spécifiques telles que la disponibilité des ressources de calcul par exemple. Pour permettre ce changement dynamique de comportement, il faut être en mesure d'assurer la transition entre plusieurs niveaux de comportements tout en garantissant la cohérence de la simulation.

La simulation multi-niveaux est utilisée, dans cette thèse, comme une application des systèmes multi-agents holoniques. Le résultat de cette approche a conduit à la conception d'une simulation multi-niveaux de piétons en environnement urbain virtuel. Dans le cadre de cette application, les SMAH sont mis à profit pour faire cohabiter, dans la même simulation, différents niveaux de comportements pour les piétons, et assurer dynamiquement la transition entre ces niveaux en fonction des ressources de calcul disponible.

1.3 Plan de la thèse

Après ce bref tour d'horizon des propositions présentées dans cette thèse, cette section introduit le plan de ce manuscrit. Selon les objectifs énoncés dans la section 1.2, cette thèse est organisée en trois parties principales. Les différents chapitres qui composent ce document ainsi que leur organisation respective sont résumés par la figure 1.1.

Le chapitre 2 et plus généralement la première partie de cette thèse sont consacrés à la présentation des principaux métamodèles et méthodologies orientés-agent. Une attention toute particulière est portée sur ceux qui intègrent une approche organisationnelle ou qui considèrent les agents comme des entités composées d'agents.

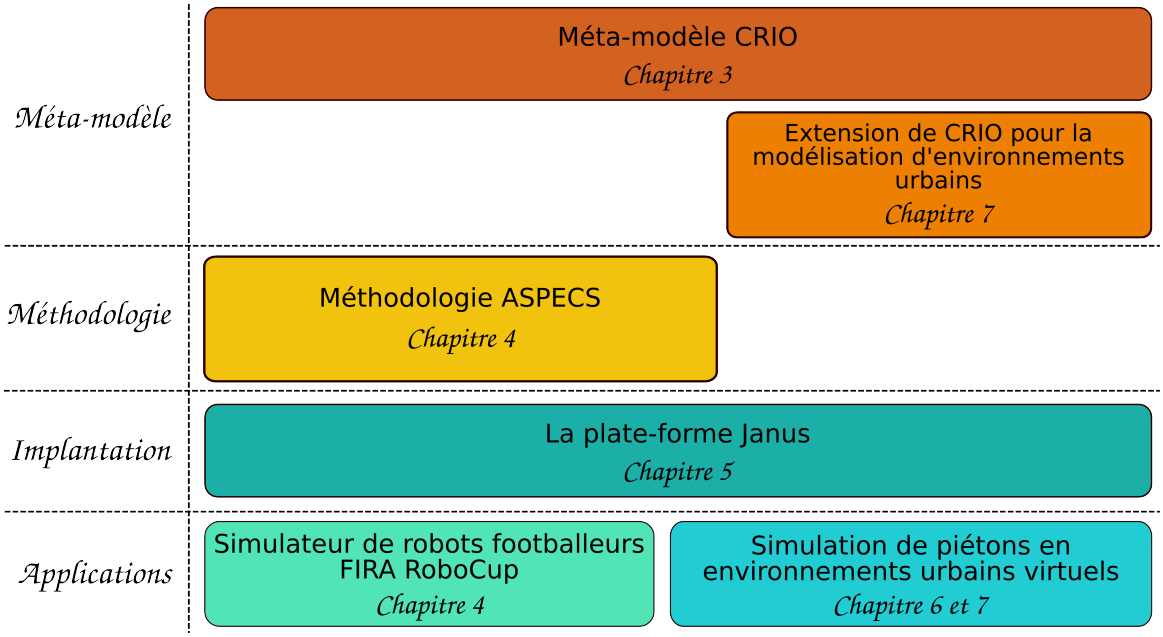


Figure 1.1: Schéma descriptif de l'approche proposée et de l'organisation de cette thèse

La seconde partie (chapitres 3, 4 et 5) constitue le cœur de notre contribution et présente le métamodèle CRIO, le processus de développement ASPECS ainsi que la plate-forme *Janus* dédiée à l'implantation et à la simulation des SMAH.

Le chapitre 3 introduit le métamodèle CRIO et décrit les différents concepts qui le composent. Les principales propriétés de ce modèle sont également décrites.

Le chapitre 4 présente le processus de développement ASPECS et montre comment exploiter les concepts introduits au chapitre précédent pour développer une solution multi-agents holonique à un problème complexe donné. Ce chapitre est illustré par un exemple visant à développer un simulateur de robots footballeurs.

Le chapitre 5 décrit la plate-forme *Janus* et ses principales caractéristiques. Un bref état de l'art sur les nombreuses plates-formes multi-agents est inclus dans cette thèse à l'annexe B (page 223).

Enfin la troisième et dernière partie (chapitres 6 et 7) est consacrée à la simulation multi-niveaux et à son illustration sur l'exemple de la simulation de piétons en environnement urbain virtuel.

Le chapitre 6 montre comment exploiter les propriétés des SMAH et du métamodèle CRIO pour le développement d'outils destinés à la gestion de simulations multi-niveaux.

Les outils et modèles, présentés dans ce chapitre, sont le fruit d'une expérience menée dans le cadre du développement de la simulation multi-niveaux de piétons en environnement virtuel qui est présentée au chapitre 7.

Le chapitre 7 illustre le principe de la simulation multi-niveaux dans le cadre d'une simulation de piétons en environnement urbain virtuel. Ce chapitre présente également une extension du métamodèle CRIO pour la modélisation multi-niveaux des environnements urbains dans le cadre d'une simulation multi-agents.

Le chapitre 8 présente un bilan des travaux de recherche décrits dans cette thèse et dresse un panorama de quelque unes des perspectives possibles afin de guider les travaux futurs et les éventuelles extensions des modèles proposés.

CHAPITRE 2

INGÉNIERIE LOGICIELLE ORIENTÉE-AGENT

LES AGENTS ont été utilisés dans un grand nombre d'applications et ont donné naissance à de nombreux modèles et méthodologies. Ce chapitre ne couvre pas l'intégralité des modèles existants, mais tente de cibler les plus importants et ceux qui ont constitué l'inspiration principale de l'approche proposée. Les modèles adoptant une approche organisationnelle seront donc plus particulièrement étudiés.

Sommaire

2.1 Introduction

Les systèmes multi-agents revêtent de plus en plus d'importance pour leur capacité à aborder les systèmes complexes. Le domaine des SMA peut être vu comme le confluent de plusieurs disciplines de recherche : l'intelligence artificielle pour les aspects décisionnels de l'agent, l'intelligence artificielle distribuée et plus généralement les systèmes distribués pour les interactions entre agents et la distribution de la résolution et de l'exécution, et enfin le génie logiciel pour l'approche de modélisation orientée-agent et la création de composants logiciels autonomes.

Les préambules de nombreux articles de recherche traitant de l'ingénierie logicielle orientée-agent ou des SMA dressent fréquemment le constat de leur intérêt grandissant, leurs potentialités, ou notent que les SMA sont devenus un nouveau paradigme de modélisation utilisé dans de plus en plus d'applications. Il est dès lors naturel de s'interroger sur les raisons d'un tel succès. [Jennings et al., 1998, p. 6-10] et [Ferber, 1995, p. 8-13] apportent quelques éléments de réponse.

Tout d'abord, les problèmes complexes sont fonctionnellement distribués. Concevoir un lanceur spatial ou un avion de ligne réclame l'intervention d'un grand nombre de personnes, chacune disposant d'une vue partielle du produit final. La complexité de la conception de produits industriels aussi élaborés est trop importante pour un seul individu. Dès lors, l'approche agent apparaît comme une méthode naturelle de conception. Par exemple, un scénario suscitant bon nombre d'intérêt dans les SMA est celui des agents logiciels qui achètent et vendent des marchandises par l'intermédiaire d'Internet au nom des utilisateurs. Il apparaît naturel de considérer les participants à de telles transactions comme des agents (semi-)autonomes. En cela l'approche agent vient compléter le panel des approches de modélisation possibles en offrant des abstractions plus adaptées à la modélisation de certains domaines.

Le second point concerne la distribution des données, de la commande ou du contrôle. Les problèmes complexes sont physiquement distribués à travers différents types de réseaux. Pour beaucoup de systèmes logiciels, il n'est pas possible d'identifier un point unique de commandement global, la commande est généralement distribuée à travers un réseau de différents nœuds de traitement, géographiquement distincts. Afin de faire fonctionner ces différents nœuds de manière cohérente et d'en assurer la robustesse, ceux-ci doivent être capables d'interagir de manière autonome. *« Ils doivent donc être des agents »* [Jennings et al., 1998].

Le troisième point concerne l'histoire des systèmes logiciels et des entreprises qui les utilisent. De nombreux systèmes d'ancienne génération continuent à être utilisés (*"legacy systems"*) pour diverses raisons : ils fonctionnent bien, il serait trop coûteux d'en changer, les compétences requises pour les mettre à jour ne sont plus dans l'entreprise, mauvaise documentation, etc. Un moyen naturel d'intégrer ces anciens systèmes dans les systèmes distribués modernes consiste à les *« agentifier »*, ou du moins à ajouter une interface agent

leur permettant d'interagir avec les autres agents. Cela rejoint le fait que concevoir un logiciel efficace et fiable ne suffit plus. Les logiciels actuels doivent être en mesure de s'adapter et d'évoluer rapidement pour répondre à de nouveaux besoins. La nature distribuée des SMA permet d'ajouter ou de retirer des fonctionnalités par l'addition ou la suppression de nouveaux agents.

Le quatrième et dernier point concerne les systèmes ouverts dans lesquels il est impossible de déterminer à la conception quelles seront leurs composantes, ou comment ces composantes seront susceptibles d'interagir. La capacité à disposer de mécanismes de prise de décision autonomes et flexibles s'avère donc indispensable pour concevoir des logiciels évoluant dans des environnements ouverts.

Bien que les SMA offrent de nombreux avantages les rendant très attractifs, [Jennings et al., 1998] affirment également qu'ils sont souvent dangereusement surestimés et risquent de subir un sérieux retour de flamme, comme celui vécu par l'intelligence artificielle *« classique »*[1] dans les années 80 [Guilfoyle and Warner, 1994, Wooldridge, 1997]. S'il ne fait aujourd'hui plus guère de doute concernant l'efficacité des SMA dans la modélisation des systèmes complexes, le constat précédent conduit à adopter une approche pragmatique face à leur utilisation. Éviter ce revers passe par le développement de logiciels multi-agents efficaces, fiables, modulaires et aisément adaptables, mais surtout par le transfert de cette technologie auprès des industriels. Le développement des méthodologies orientées-agent s'intègre dans cette perspective. En effet, la conception de méthodologies impose par nature pragmatisme et cohérence pour à la fois intégrer les principes et les composants des méthodes existantes qui ont fait leurs preuves, et garantir une transition aisée vers la nouvelle technologie incarnée par les SMA. Considérer le développement logiciel comme un processus a considérablement contribué à la qualité des processus de développement, et par conséquent à celle des logiciels. En effet, il y a une corrélation directe entre la qualité du processus de développement logiciel et la qualité du logiciel produit [Fuggetta, 2000]. Pour prendre en compte les nouveaux besoins requis pour le développement de systèmes multi-agents, de nouveaux processus de développement se doivent d'être élaborés. Ce chapitre tente de dresser un tour d'horizon de l'ingénierie logicielle orientée-agent et des standards qui la régissent. Après un rappel succinct des fondements des systèmes multi-agents, les processus de développement et les approches de modélisation de logiciels multi-agents existants seront détaillés.

2.2 Définitions et fondamentaux

Cette section s'attache à fournir un bref récapitulatif des définitions fondamentales au cœur de cette thèse. Les concepts d'agent et de systèmes multi-agents y seront notamment introduits ainsi que la notion de holon.

[1] Par opposition à l'intelligence artificielle distribuée (IAD) dont les SMA font partie.

2.2.1 Agent ou objet ?

La différence entre les concepts d'Agent et d'Objet a été étudiée par de nombreux travaux, parmi lesquels on peut notamment citer : [Bergenti and Huhns, 2004, Odell, 2002a, Wooldridge and Ciancarini, 2001]. Cette question reste d'ailleurs toujours posée, même si de nombreux auteurs s'accordent pour dire que les agents diffèrent des objets essentiellement par leur capacité à interagir et par leur autonomie.

L'autonomie d'un agent réfère d'une part à sa possibilité de déterminer seul le comportement qu'il doit adopter et d'autre part à son indépendance en termes de ressources d'exécution. Les agents sont par nature des entités actives, alors que les objets sont considérés comme des entités passives devenant actives uniquement lorsqu'une de leurs méthodes est invoquée. En effet dans la perspective traditionnelle de l'orienté-objet [Booch, 1993], l'objet est intrinsèquement non autonome, il ne devient actif qu'à la suite d'une demande de service provenant d'un processus de contrôle externe. Un objet, contrairement à un agent, n'est pas en mesure de choisir laquelle de ces méthodes il devra exécuter (pas de sélection d'action). Il ne dispose donc que d'un contrôle limité sur son propre comportement, assujetti à la décision d'un acteur externe. L'agent en revanche détermine le comportement qu'il doit adopter en accord avec ses objectifs propres, ses connaissances, son état interne et ses interactions.

En revanche, les objets, tels qu'ils sont considérés dans les systèmes distribués actuels, ont tout de même largement évolué depuis leur définition originelle et se rapprochent donc sensiblement du concept d'agent. Aujourd'hui les objets peuvent être actifs et disposer de leurs propres ressources d'exécution. Ils sont ainsi capables de répondre à plusieurs demandes de service de manière concurrente, ou de refuser une demande non autorisée ou considérée comme peu sûre. En somme ils sont en mesure de décider de ne pas effectuer une action [Zambonelli et al., 2003]. On peut considérer en quelque sorte qu'un objet actif deviendrait agent s'il était capable de faire preuve d'initiative [Bernon et al., 2005].

Mais la différence majeure entre agent et objet tient davantage dans la dimension sociale de l'agent. Un agent est rarement considéré de manière individuelle, il est généralement intégré dans une société évoluant dans un environnement commun où chaque agent communique directement ou indirectement avec ses partenaires pour satisfaire des objectifs communs ou une tâche partagée. Ces différences entre agent et objet ne signifient pas que les techniques d'ingénierie logicielle orientée-objet ne puissent pas être transférées à la modélisation orientée-agent, mais cela implique qu'elles doivent au moins être étendues pour notamment prendre en compte cette dimension sociale intrinsèque à la notion d'agent.

2.2.2 Définition de la notion d'agent

L'une des définitions les plus célèbres de la notion d'agent a été formulée par [Russell and Norvig, 1995], ils considèrent un agent comme *« Tout ce qui peut être vu comme per-*

cevant son environnement à l'aide de capteurs et agissant sur cet environnement à l'aide d'effecteurs, de façon autonome ». Cette définition très générale et volontairement minimaliste dans sa formulation (uniquement) a été étendue notamment par Ferber pour, en outre, accentuer l'importance de l'environnement (qui demeurait rarement spécifié). Pour [Ferber, 1995, p. 13], *« l'agent est une entité physique ou virtuelle :*

1. *qui est capable d'agir dans un environnement ;*
2. *qui peut communiquer directement avec d'autres agents ;*
3. *qui est mue par un ensemble de tendances (sous la forme d'objectifs individuels ou de fonctions de satisfaction, voire de survie, qu'elle cherche à optimiser) ;*
4. *qui possède des ressources propres ;*
5. *qui est capable de percevoir son environnement (mais de manière limitée) ;*
6. *qui ne dispose que d'une représentation partielle de cet environnement (et éventuellement aucune) ;*
7. *qui possède des compétences et offre des services ;*
8. *qui peut éventuellement se reproduire ;*
9. *et dont le comportement tend à satisfaire ses objectifs, en tenant compte des ressources et des compétences dont elle dispose, et en fonction de sa perception, de ses représentations et des communications qu'elle reçoit. »*

2.2.3 Définition d'un système multi-agents

La plupart des auteurs s'accordent généralement pour définir un système multi-agents (SMA) comme un système composé d'agents qui communiquent et collaborent pour achever des objectifs spécifiques personnels ou collectifs. La communication implique l'existence d'un espace partagé support de cette communication. Cet espace est généralement qualifié d'Environnement.

Pour [Ferber, 1995, p. 15], *« un Système Multi-Agents est un système composé des éléments suivants :*

- *Un environnement E, c'est-à-dire un espace disposant généralement d'une métrique.*
- *Un ensemble d'objets O. Ces objets sont situés, c'est-à-dire que pour tout objet, il est possible, à un moment donné, d'associer une position dans E. Ces objets sont passifs, c'est-à-dire qu'ils peuvent être perçus, créés, détruits et modifiés par les agents.*
- *Un ensemble A d'agents qui sont des objets particuliers ($A \subseteq O$), lesquels représentent les entités actives du système.*
- *Un ensemble de relations R qui unissent des objets (et donc des agents) entre eux.*
- *Un ensemble d'opérations Op permettant aux agents de A de percevoir, produire, consommer, transformer, et manipuler des objets de O.*[2]
- *Des opérateurs chargés de représenter l'application de ces opérations et la réaction du monde à cette tentative de modification, que l'on appellera les lois de l'univers. »*

[2]Cela correspond entre autres à la faculté des agents de percevoir leur environnement, de manger, etc.

Un SMA est une société organisée d'agents dans laquelle un certain nombre de phénomènes peuvent émerger comme la résultante des interactions entre les agents[3]. Cette notion d'émergence est essentielle dans les SMA, car c'est l'une des propriétés qui les rendent si aptes à modéliser les systèmes complexes.

2.2.4 Systèmes multi-agents holoniques

Comme le précise l'introduction de ce document, la plupart des modèles actuels considèrent les agents comme des entités atomiques [Gasser, 1992, Giret and Botti, 2003]. La question de la représentation du fait qu'un groupe, une organisation ou une société d'agents peuvent, à un certain niveau d'abstraction, se comporter comme une entité individuelle demeure une question relativement ouverte dans le domaine des SMA. De nombreuses approches ont tenté de modéliser cette idée d'agents composés d'agents. Chaque chercheur donne un nom spécifique à ce type d'agent : agents *collectifs* et *individuels* [Ferber, 1995], *méta-agents* [Holland, 1995], *agents intermédiaires* [Marcenac and Calderoni, 1997], *agents récursifs* ou *intermédiaires* [Correa e Silva Fernandes, 2001] ou encore *groupes agentifiés* [Odell et al., 2005]. Dans ce manuscrit, ce type d'agent sera qualifié par les termes *holon* ou *agent holonique*, en référence à l'approche holonique initialement introduite par [Koestler, 1967]. Ce concept de holon n'a cessé d'évoluer particulièrement dans le domaine de la philosophie où il est considéré comme un outil puissant pour modéliser les phénomènes naturels et sociaux [Edwards, 2003, Kauffman, 1996, Kofman, 2001, Simon, 1996, Wilber, 1995].

Le modèle holonique a également été utilisé, dans les années 90, dans le domaine des systèmes manufacturiers (HMS[4] et IMS[5]) comme un nouveau paradigme de modélisation. Il a depuis fait l'objet de beaucoup d'attentions et de nombreux modèles sont apparus tel que PROSA [Brussel et al., 1998, Wyns, 1999][6] ou MetaMorph [Maturana, 1997, Maturana et al., 1999].

Toutefois, la vision holonique dans les systèmes manufacturiers diffère de celle utilisée dans les SMA [Giret and Botti, 2004, Schillo, 2004]. [Fischer et al., 2003] identifient d'ailleurs des différences très claires entre ces deux visions :

– La modélisation de la composition hiérarchique entre les groupes d'agents et de la structure gigogne qui en découle fait partie intégrante des systèmes multi-agents holoniques. En fonction de la complexité des tâches, les agents holoniques peuvent créer des hiérarchies de composition de profondeur arbitraire, ce qui n'est pas le cas dans les systèmes holoniques manufacturiers où un holon est généralement composé d'une entité physique simple (unité de production) et d'un contrôleur logiciel associé à cette

[3] Cette résultante n'est d'ailleurs pas forcément prédictible.

[4] HMS, Holonic Manufacturing Systems : `http://hms.ifw.uni-hannover.de`

[5] IMS, Intelligent Manufacturing System : `http://www.ims.org`

[6] PROSA - Product, Resource, Order, Staff Architectures : `http://www.mech.kuleuven.be/goa/prosa.htm`

entité.

- Les systèmes holoniques manufacturiers ne font aucune pré-supposition sur les caractéristiques et propriétés internes d'un holon. La seule exigence est que chaque holon agisse en tant qu'unité de contrôle alors que dans les SMAH, un holon se doit de posséder les propriétés classiquement assignées aux agents telles que l'autonomie, l'habilité sociale, la réactivité et la pro-activité [Wooldridge and Jennings, 1995].
- Les systèmes holoniques manufacturiers sont basés sur la métaphore du marché (*"market metaphor"*) pour concevoir la coordination entre holons. Les recherches dans les SMAH s'attachent également à déterminer des partenaires pour des collaborations à long terme (et ainsi permettre la création de coalition), mais favorisent surtout la diversité des structures organisationnelles possibles [Knabe et al., 2003].

Malgré ces différences entre la vision des holons adoptée dans le SMA et celle des systèmes manufacturiers, de nombreux modèles proposés dans ce dernier domaine peuvent être adaptés aux systèmes multi-agents holoniques. La référence actuelle de la notion de holon dans les systèmes multi-agents est incarnée par l'introduction de [Gerber et al., 1999] où il est proposé une vision des holons correspondant davantage à la notion d'agent récursif ou composé. [Rodriguez, 2005] a ensuite adapté cette vision des holons, avant tout centrée sur l'agent, vers une vision organisationnelle et plus modulaire en proposant un *"framework"* organisationnel générique pour décrire la notion de holon qui constitue l'une des bases du métamodèle CRIO décrit au chapitre suivant.

La figure 2.1 résume la définition de la notion de holon telle qu'elle est considérée dans les SMA. Les holons de niveau n sont regroupés en organisation qui au niveau $n+1$ peuvent être considérés comme des entités individuelles. Inversement des holons considérés comme des entités individuelles de niveau $n+1$ peuvent être considérés comme des organisations au niveau n. Ce processus de composition hiérarchique peut être réitéré sur un nombre arbitraire de niveaux.

Depuis leur utilisation dans les systèmes manufacturiers, les systèmes holoniques ont été appliqués dans de nombreux autres domaines tels que la santé [Ulieru and Geras, 2002], les systèmes de transport [Bürckert et al., 1998], les problèmes de couverture dans les réseaux radio-mobiles [Rodriguez et al., 2003], les robots footballeurs [Barbat et al., 2001, Candea et al., 2000] ou encore le travail collaboratif [Adam, 2000]. Mais il n'existe actuellement aucune méthodologie générique qui exploite les propriétés des systèmes multi-agents holoniques dans leur processus de développement de logiciel.

2.3 Langages de modélisation orientée-agent

Une méthodologie est généralement composée d'un processus et de ses produits [Rolland et al., 1999]. Or ces produits sont documentés, représentés ou implantés par des langages. Dans les langages, on distingue généralement les langages de spécification (formels

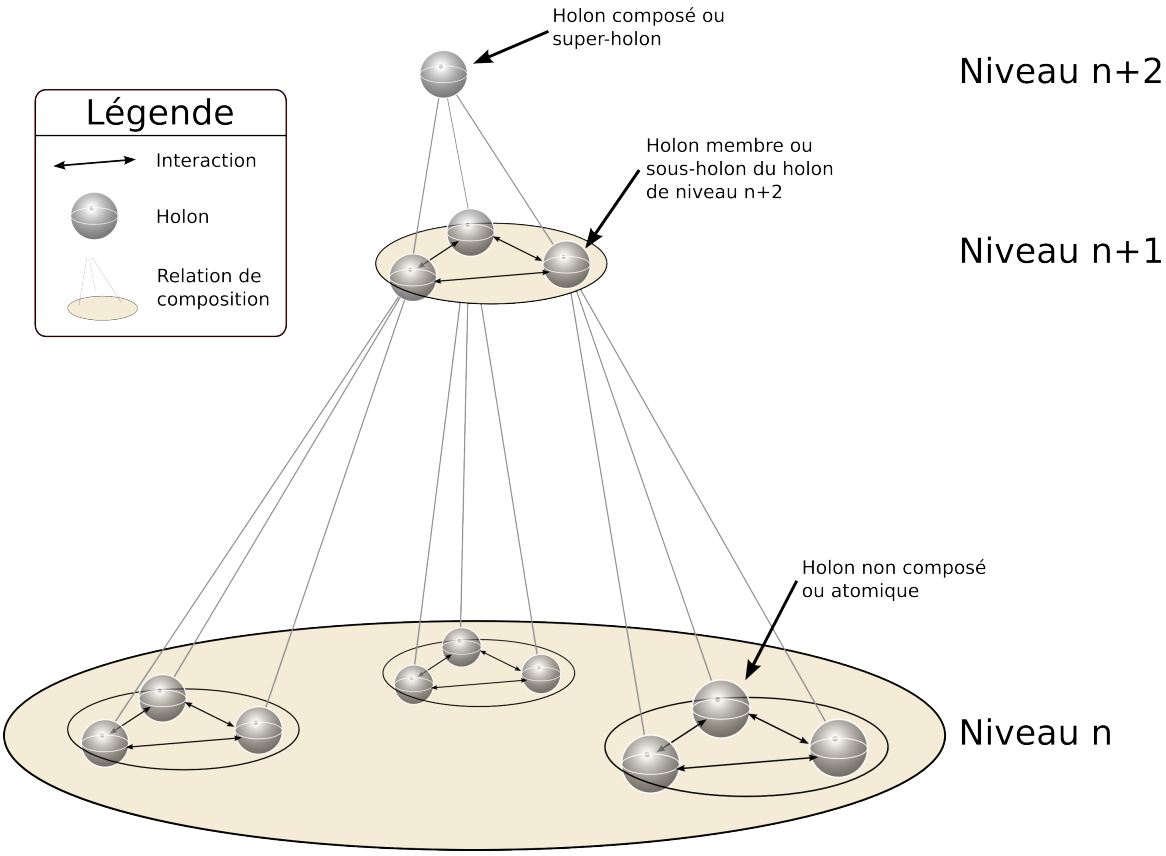


Figure 2.1: La notion de Holon et la terminologie associée. Les holons de niveau n sont regroupés en organisations qui au niveau $n+1$ peuvent être considérées comme des entités individuelles (fig. inspirée de [Ferber, 1995, p. 100])

ou non, ex : Z, Object-Z, B), les langages de modélisation (ex : OMT[7], UML[8], OML[9]), les langages d'implantation (ex : Java, C++, SQL, IDL, Lisp, etc), les langages de contrainte (ex : OCL[10]) et le langage naturel.

Dans cette section, seuls les langages de modélisation, supports indispensables à toute méthodologie, seront étudiés. Comme cela a été précisé au début de ce chapitre, les agents ne sont pas si différents des objets, et la plupart des techniques de modélisation sont issues de l'ingénierie logicielle orientée-objet. Le plus connu et le plus utilisé des langages de modélisation orientée-objet est UML. UML est également un métamodèle, lui même issu d'un métamétamodèle : la MOF[11]. Il est aujourd'hui considéré comme un standard. Mais étant originellement destiné aux méthodes orientées-objet, UML ne fournit pas toutes les fonctionnalités nécessaires pour la modélisation orientée-agent. En revanche UML définit toute une gamme d'outils pour faciliter son extension (les profils qui intègrent des stéréotypes particuliers, des contraintes, etc.). Ces outils peuvent ensuite être exploités pour intégrer les

[7]Object Modeling Technique de James Rumbaugh.
[8]Unified Modeling Language.
[9]Open Modeling Language de Brian Henderson-Sellers, Donald G. Firesmith et Ian Graham.
[10]Object Constraint Language [OCL, 2006]
[11]*"Meta-Object Facility"* : `http://www.omg.org/mof/`

spécificités de la modélisation orientée-agent au sein du langage.

Cette section présente trois tentatives d'extensions du modèle UML pour intégrer les concepts nécessaires à la modélisation orientée-agent. Elle détaille notamment les langages AUML, AML et AORML.

AUML [Bauer et al., 2001, Bergenti and Poggi, 2000, Odell et al., 2000, 2001][12] — Agent Unified Modeling Language — est issu d'un travail collaboratif entre de nombreux chercheurs. Divers diagrammes UML ont été étendus. [Odell et al., 2001] et [Bauer et al., 2001] redéfinissent notamment les diagrammes de séquence pour permettre la spécification des protocoles d'interaction entre agents. Ils définissent la notion de rôle d'agent, de ligne de vie, ainsi que les différents types de protocoles. La sémantique des messages UML est également étendue pour intégrer notamment les actes de langage. [Bauer, 2002] étend les diagrammes de classes et plus spécifiquement le concept de classe (représentant un agent) pour y intégrer les notions d'actions, de capacité et d'état de l'agent notamment utilisé pour représenter les croyances, les désirs et les intentions des agents (architecture BDI[13]). Une partie est également consacrée à la représentation du comportement de l'agent sous forme d'automate à états. AUML est désormais en partie intégré dans certains outils tels que INGENIAS Development Kit [Pavón et al., 2005] ou Opentool [Bernon et al., 2003] associé à la méthodologie ADELFE.

AML [Trencansky and Cervenka, 2005] — Agent Modeling Language — fournit tout un ensemble d'extensions à UML décrites dans la figure 2.2. AML définit trois types d'éléments semi-abstraits nommés semi-entités :

- Les semi-entités comportementales représentent des éléments possédant des capacités propres et capables d'agir, d'observer ou de percevoir leur environnement.
- Les semi-entités sociales qui peuvent former des sociétés, avoir des relations sociales et posséder des propriétés sociales propres.
- Enfin, les semi-entités mentales qui représentent des éléments pouvant être caractérisés en termes d'états mentaux : croyances, objectifs, besoins, désirs, buts, etc.

Ces semi-entités peuvent ensuite être surchargées par héritage pour intégrer les besoins d'approches plus spécifiques. Les entités fondamentales composantes d'un SMA telles les agents, les ressources et l'environnement disposent également de leurs définitions dans ce langage. AML peut lui-même être aisément étendu puisqu'il définit un métamodèle complet héritant de la super-structure UML.

AORML [Wagner, 2003] — Agent-Object-Relationship Modeling Language — Dans cette approche, deux points de vue différents sont adoptés sur la notion d'agent.

[12] AUML : `http://www.auml.org`
[13] BDI : Beliefs, Desires and Intentions

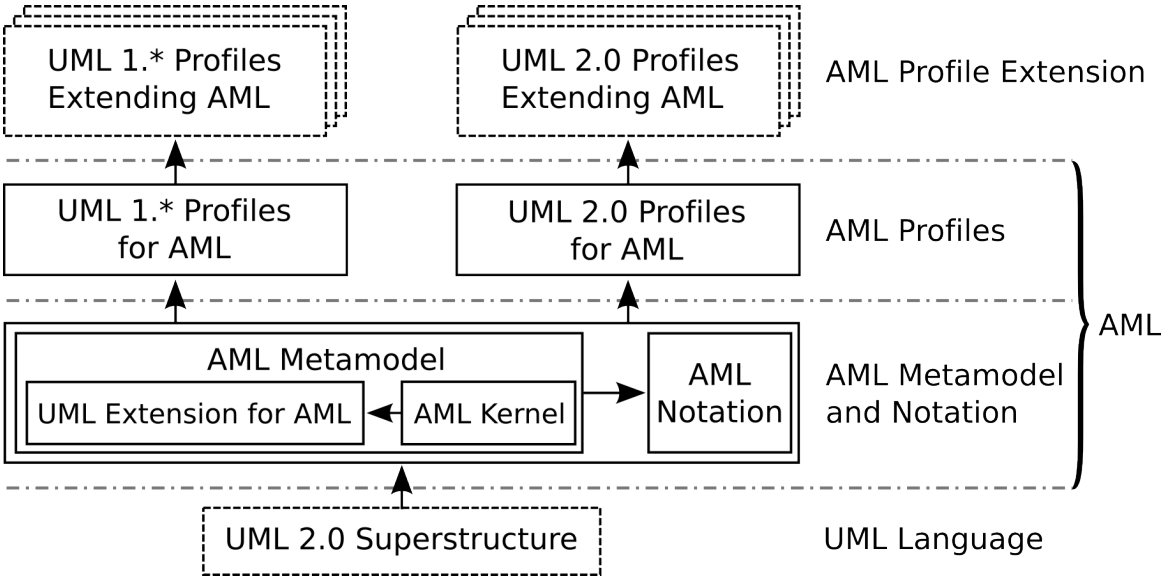


Figure 2.2: Définition des différents niveaux de AML [Trencansky and Cervenka, 2005]

Le point de vue externe décrit les agents, leurs interactions dans le domaine de l'application, leurs croyances sur les objets qu'ils manipulent ainsi que les relations qui relient ces différents éléments. Ces différents points sont représentés par les diagrammes d'Agent (*"Agent Diagram"*), de fenêtre d'interaction (*"Interaction Frame Diagram"*), de séquence d'interaction (*"Interaction Sequence Diagram"*) et de schéma d'interaction (*"Interaction Pattern Diagram"*). Le diagramme d'Agent décrit les différents types d'agents et les objets du domaine. Le diagramme de fenêtre d'interaction décrit les interactions possibles entre deux types d'agents, les types d'événements possibles ainsi que les types d'engagements (*"commitment"*). Le diagramme de séquence d'interaction décrit une instance d'un processus d'interaction. Enfin, le diagramme de schéma d'interaction décrit les schémas généraux d'interaction par un ensemble de règles de réaction définies dans le type de processus d'interaction.

Le point de vue interne décrit quant à lui les différentes composantes internes d'un agent. Ce point de vue est associé aux diagrammes de fenêtre de réaction (*"Reaction Frame Diagram"*), de séquence de réaction (*"Reaction Sequence Diagram"*) et de schéma de réaction (*"Reaction Pattern Diagram"*). Le diagramme de fenêtre de réaction décrit les autres agents ou types d'agents, les types d'actions et d'événements ainsi que les engagements qui déterminent les diverses interactions possibles que l'agent en cours d'étude peut avoir avec eux. Le diagramme de séquence de réaction décrit les instances du processus d'interaction donné du point de vue interne de l'agent étudié. Enfin, le diagramme de schéma de réaction se concentre sur les modèles de réaction de l'agent en cours d'étude, ces réactions sont également exprimées sous la forme de règles de réaction.

2.4 Métamodèle et méthodologie orientés-agent

Comme le précise [Pavón, 2006, p. 19], malgré le nombre important d'applications basées sur le paradigme multi-agents, l'expérience de la conception et de l'utilisation de SMA à un niveau industriel et le retour de cette expérience manquent encore cruellement. La grande majorité des applications multi-agents est construite sans utiliser de composants agentifiés réutilisables et n'est pas généralisable. Les méthodologies multi-agents sont essentielles pour parvenir à étendre l'utilisation du paradigme agent hors du domaine académique vers celui de l'industrie. La plupart des méthodologies existantes s'inspirent des résultats et des contributions issus de l'ingénierie logicielle orientée-objet en y intégrant les spécificités liées à l'approche agent telles que l'autonomie ou la nature sociale des agents (PASSI [Chella et al., 2006, Cossentino, 2005], MESSAGE [Caire et al., 2002], AAII [Kinny et al., 1996], MaSE [DeLoach et al., 2001], ADELFE [Bernon et al., 2002]). Ce constat n'a d'ailleurs rien d'étonnant vu les nombreux points communs que partagent les concepts d'agent et d'objet.

2.4.1 Définition et description générale

Avant de procéder plus avant dans cette section consacrée aux méthodologies orientées-agent et de sorte à dissiper toute confusion possible, un glossaire de quelques définitions de base en ingénierie logicielle est fourni, notamment concernant les termes de méthodologie et de processus (de développement) qui sont parfois considérés comme des synonymes, et parfois employés avec des significations contradictoires.

Définition 2.1 Métamodèle, [Bézivin, 2005]

Un métamodèle est une spécification formelle d'une abstraction, généralement consensuelle et normative. Depuis un système donné, on peut extraire un modèle particulier à l'aide d'un métamodèle spécifique. Un métamodèle agit comme un filtre précisément défini exprimé dans un formalisme donné.

Définition 2.2 Processus de développement logiciel, [Fuggetta, 2000]

Un processus de développement de logiciel est l'ensemble cohérent des politiques, structures organisationnelles, technologies, procédures, et diagrammes nécessaires à la conception, au développement, au déploiement et à la maintenance (évolution) d'un produit logiciel.

Définition 2.3 Méthode, [Cernuzzi et al., 2005]

Une méthode prescrit une manière d'exécuter un type d'activité dans un processus donné afin de produire correctement un résultat spécifique à partir d'une entrée spécifique. Toutes les phases d'un processus, pour être applicables avec succès, doivent être complétées par toutes directives méthodologiques (identification des techniques et outils à employer, et définition de la façon dont les résultats doivent être produits) susceptibles d'aider les parties prenantes du projet à accomplir leur travail selon les meilleures pratiques définies.

Définition 2.4 Méthodologie, [Ghezzi et al., 2002]

Une méthodologie est une collection de méthodes couvrant et reliant les différentes étapes d'un processus. L'objectif d'une méthodologie est de prescrire une approche cohérente à adopter pour résoudre un problème dans le contexte d'un processus de développement de logiciel en pré-sélectionnant et reliant un certain nombre de méthodes.

Dans les dix dernières années, de nombreux efforts de l'ingénierie logicielle orientée-agent se sont portés sur la définition de méthodologies pour guider le processus de développement des systèmes multi-agents. La conception de la plupart des méthodologies (orientées agent ou non) débute généralement par la définition d'un métamodèle qui identifie les abstractions de base nécessaires à la description du problème et de sa solution. Ces différentes abstractions sont ensuite organisées afin de définir les étapes à suivre pour l'analyse, la conception, et l'implantation ainsi que les résultats que chacune d'elles doit produire. Malheureusement, ceci est loin d'être suffisant du point de vue de la mise en pratique du développement logiciel [Cernuzzi et al., 2005]. En effet dans le domaine du développement de systèmes logiciels et donc des systèmes multi-agents, l'identification d'une méthodologie appropriée passe d'abord par l'identification d'un modèle spécifique pour son processus de développement logiciel [Boehm, 1988]. Un tel modèle doit définir : (i) l'organisation et la coordination des différentes phases du développement, (ii) les différentes personnes qui doivent intervenir dans chaque activité et à quel moment, (iii) les technologies et les outils qui doivent être utilisés dans chaque activité, (iv) les produits fournis par chaque activité et enfin (v) les ressources impliquées dans chaque phase du processus de production. En d'autres termes, le modèle de processus et le processus qui en découle doivent guider l'ensemble de l'effort de production et être complétés par un ensemble de lignes méthodologiques spécifiques à la méthodologie développée. Le terme *« méthodologie »* est donc souvent employé dans un sens qui le réduit à une sous-partie de son processus et à ses produits. Ce document n'abordera pas les différents types de modèles de processus de développement (Cascade, Spirale, Fontaine, etc.). Cependant d'après [Sommerville, 2004], les

différents types de processus de développement logiciel partagent quelques activités fondamentales. Celles-ci incluent la spécification, la conception, l'implantation, la validation et enfin l'évolution.

Spécification La spécification est généralement effectuée en adoptant deux perspectives différentes sur le logiciel : Utilisateur et Système. La première adopte le point de vue externe d'un utilisateur et vise à définir ses besoins. La seconde consiste à déterminer d'une part les fonctionnalités du logiciel en accord avec les besoins des utilisateurs, et d'autre part les contraintes que le système doit satisfaire tout en intégrant les contraintes issues du domaine de l'application.

Au sein de la spécification, les approches formelles sont habituellement clairement distinguées des approches semi-formelles. Les premières adoptent généralement un formalisme mathématique pour élaborer une spécification du système qui peut ensuite être utilisée pour l'implanter et vérifier si cette implantation valide la spécification initiale. Parmi ces approches on peut notamment citer les travaux de [Luck and d'Inverno, 2001, 1995] basés sur le langage Z, ou la méthodologie DESIRE [Brazier et al., 1995, 1997, 1998]. Même si le métamodèle CRIO dispose d'une spécification formelle partielle [Hilaire et al., 2000, Rodriguez et al., 2005a] basée sur le formalisme OZS[14] [Gruer et al., 2004], ces aspects ne sont pas abordés dans cette thèse et les méthodes de spécification formelle ne seront donc pas traitées dans cet état de l'art.

Les approches semi-formelles sont quant à elles généralement basées sur une utilisation conjointe de notations graphiques et du langage naturel en suivant un modèle structuré. Trois grandes classes de spécifications semi-formelles peuvent être discernées :

- les approches fonctionnelles [Jacobson et al., 1992] sont le plus souvent basées sur les diagrammes de cas d'utilisation (*"use cases"*). Elles sont notamment utilisées dans les méthodologies PASSI [Cossentino, 2005], ROADMAP [Juan et al., 2002], Prometheus [Padgham and Winikoff, 2002a,b], ADELFE [Bernon et al., 2002, Picard, 2004].
- les approches orientées-but [Van Lamsweerde, 2001] visent à identifier et à modéliser les objectifs du système et sont en outre utilisées dans les méthodologies MESSAGE [Caire et al., 2002], INGENIAS [Pavón et al., 2005] et TROPOS [Bresciani et al., 2004, Kolp et al., 2006].
- et enfin les approches orientées-rôle [Kendall, 2000]. Si les deux premières sont considérées comme conventionnelles, car déjà largement utilisées dans l'approche orientée-objet, celle-ci est en revanche plus spécifique au domaine agent. Les méthodologies intégrant cette approche sont notamment GAIA, SODA [Omicini, 2000] et Rica [Serrano and Ossowski, 2004]. Toutefois MESSAGE, INGENIAS et PASSI utilisent également le concept de rôle dans leurs spécifications.

D'autres méthodologies combinent ces approches ; MaSe [DeLoach et al., 2001] par

[14] Object Z et Statecharts

exemple débute par la capture des objectifs puis étudie les cas d'utilisation. Enfin, la dernière étape de la phase d'analyse est consacrée au raffinement des rôles.

Conception et Implantation La conception et l'implantation consistent à produire le logiciel et à convertir les spécifications du système en un système exécutable. Le logiciel est alors généralement structuré sous forme de modules. Les interfaces entre les modules dépendants et celles avec l'extérieur du logiciel ainsi que les données manipulées doivent être définies.

Validation et Vérification On différencie généralement, dans cette phase du processus de développement, la vérification et la validation.

L'approche de vérification est essentiellement formelle, basée sur la preuve, et consiste à déterminer si le système vérifie certaines propriétés. Deux grandes classes de propriétés sont en général discernées : la sûreté qui tend à montrer que le système n'atteindra jamais un mauvais état (ex : absence de blocage), et la vivacité qui vise à montrer que, sous certaines conditions, un événement attendu finira par arriver (ex : un agent finira par accéder à une ressource donnée). La vérification nécessite une spécification précise et non ambiguë, et notamment formelle, lui permettant de traquer les erreurs de développement. Deux grands types d'approches de vérification sont à distinguer : les approches axiomatiques (*"proof checking"*) et les approches sémantiques (*"model checking"*).

La validation, quant à elle, tente de répondre à la question : construisons-nous le bon produit, répond-t-il aux attentes des utilisateurs ? Cela revient à déterminer si le logiciel satisfait aux besoins utilisateurs. La validation traque les défauts de conception.

La phase de test est transversale à ses deux aspects de vérification et de validation. [Myers, 2004] définit le test comme le *« processus qui consiste à exécuter un programme avec l'intention de trouver des erreurs »*[15]. La norme [IEEE-729, 1983] précise cette définition en considérant le test comme un processus, manuel ou automatique, qui vise à établir qu'un système vérifie les propriétés exigées par sa spécification, ou à détecter des différences entre les résultats engendrés par le système et ceux qui sont attendus par la spécification. Les tests sont généralement classifiés en fonction du cycle de vie logiciel, selon quatre niveaux :

Test unitaire : Test d'un programme ou d'un module isolé dans le but de s'assurer qu'il ne comporte pas d'erreur d'analyse ou de programmation.

Test d'intégration : Une progression ordonnée de tests dans laquelle des éléments logiciels et matériels sont assemblés et testés jusqu'à ce que l'ensemble du système soit testé [IEEE-729, 1983].

Test de réception : Tests généralement effectués par l'acquéreur dans ses locaux après installation d'un système ou d'une unité fonctionnelle, avec la participation du fournis-

[15] Version originale : *"Testing is the process of executing a program with the intent of finding errors."*

seur, pour vérifier que les dispositions contractuelles sont bien respectées [ISO/IEC 2382-20, 1990].

Test de régression : À la suite de la modification d'un logiciel (ou d'un de ses constituants), un test de régression a pour but de montrer que les autres parties du logiciel n'ont pas été affectées par cette modification.

Il existe également deux grandes catégories d'approches pour mener des tests logiciels :

Méthodes de tests statiques : consistent en l'analyse textuelle du code du logiciel afin d'y détecter des erreurs, sans exécution du programme.

Méthodes de tests dynamiques : consistent en l'exécution du programme à valider à l'aide d'un jeu de tests. Elles visent à détecter des erreurs en confrontant les résultats obtenus par l'exécution du programme à ceux attendus par la spécification de l'application.

La majorité des méthodologies orientées-agent utilisent les approches de vérification et de validation héritées de l'orienté-objet. Certaines méthodologies utilisent des approches spécifiques. En ce qui concerne l'approche axiomatique pour la vérification, les travaux de [Wooldridge, 1992] peuvent être cités, où deux langages de programmation orientés-agent *Agent0* [Shoham, 1993] et *Concurrent* METATEM [Fisher and Wooldridge, 1997] ont été axiomatisés avec un type particulier de logique temporelle (*"Temporal Belief Logic"*). TROPOS utilise également une approche de vérification basée sur la logique temporelle [Fuxman et al., 2001]. DESIRE exploite son approche de spécification formelle qui décrit un SMA comme un réseau de tâches hiérarchiquement organisées. Les propriétés à vérifier sont également spécifiées en logique temporelle. Toutes ces approches sont pour l'instant limitées et ne permettent pas de tester des logiciels considérés comme complexes.

Évolution et maintenance L'évolution doit permettre au logiciel d'être modifié afin d'intégrer de nouveaux besoins clients. L'évolution et la maintenance ne seront pas détaillées dans cet état de l'art, car elles sont rarement explicitées dans les approches actuelles. Seules les méthodologies s'inspirant directement des processus issus de l'orienté-objet (ex : Rational Unified Process) traitent explicitement de ces deux phases, mais elles n'intègrent généralement aucune extension spécifique au domaine agent pour cette partie du processus. Ceci peut d'ailleurs être aisément expliqué, car pour contribuer efficacement à ces phases du processus, un retour d'expérience sur plusieurs projets de dimension industrielle est indispensable. La majorité des métamodèles et méthodologies orientés-agents existants se focalisent essentiellement sur les phases de spécification et de conception.

2.4.2 Approches centrées-agent ou centrées-organisation

Dans la spécification et la conception de SMA, les méthodologies ont évolué, depuis une vision initiale où le système était essentiellement centré sur l'agent et sur ses aspects indivi-

duels, vers une vision où il est désormais considéré comme une organisation dans laquelle les agents forment des groupes et des hiérarchies, et suivent des règles et des comportements spécifiques [Argente et al., 2006]. L'évolution des méthodologies GAIA [Wooldridge et al., 2000, Zambonelli et al., 2003] et TROPOS [Bresciani et al., 2004, Garzetti et al., 2002, Giunchiglia et al., 2002, Kolp et al., 2006] en sont d'ailleurs les exemples les plus frappants. Les approches centrées-agent (ACMAS[16]) qui s'intéressent essentiellement aux actions individuelles des agents et où le SMA est conçu sur la base des états mentaux des agents (but, désir, croyance, intention, engagement, etc), seront distinguées des approches centrées-organisation ou organisationnelles (OCMAS[17]). Les premières sont généralement dépendantes d'architectures spécifiques d'agent et de ce fait s'avèrent peu compatibles avec la nature ouverte et hétérogène des systèmes complexes. En revanche comme le décrivent [Ferber et al., 2004, Ferber and Gutknecht, 1998], les approches organisationnelles offrent de nombreux avantages par rapport aux approches centrées sur l'agent :

Hétérogénéité des langages : Si chaque groupe (instance concrète d'organisation) est considéré comme un espace d'interaction à part entière, des moyens de communication spécifiques peuvent être mis en place dans chacun d'entre eux tels que KQML[18] ou ACL[19], et ce, sans avoir à modifier l'architecture du système.

Modularité : Chaque organisation peut être considérée comme un module qui décrit un comportement particulier pour ses membres. Chacune peut donc être utilisée pour définir des règles de visibilité claire au sein du processus de conception de SMA.

De multiples architectures et applications : L'approche organisationnelle ne faisant aucune pré-supposition sur l'architecture interne des agents, elle laisse ouverte la spécification à un grand nombre de modèles et de techniques d'implantation.

Sécurité des applications : Si tous les agents communiquent sans aucun contrôle extérieur, cela peut aisément entraîner des problèmes de sécurité. En revanche, si au sein de chaque groupe l'accès aux rôles peut être réglementé lorsque cela est nécessaire, on augmente alors le niveau de sécurité sans avoir à faire appel à un système de contrôle centralisé ou *« global »*.

Cette section se focalise essentiellement sur les approches organisationnelles. C'est dans ce cadre que s'inscrivent en effet le métamodèle CRIO et la méthodologie ASPECS développés dans cette thèse. Un tour d'horizon des différentes méthodologies et métamodèles existants peut être trouvé dans la littérature suivante : [Bernon et al., 2005], [Dam and Winikoff, 2003], [Silva et al., 2003], [Wooldridge and Ciancarini, 2001], [Iglesias et al., 1999]. Au sein des approches organisationnelles, deux grandes tendances peuvent être distinguées en fonction de la vision adoptée sur la notion d'organisation [Argente et al., 2006, Coutinho

[16] ACMAS pour *"Agent Centered Multi-Agent Systems"*
[17] OCMAS pour *"Organisation Centered Multi-Agent Systems"*
[18] Knowledge Query and Manipulation Language, `http://www.csee.umbc.edu/kqml/`
[19] Agent Communication Language

et al., 2005]. La première est basée sur les notions de rôle, de groupe et sur leurs relations, mais ne considère pas explicitement la notion de norme sociale[20]. Elle est typiquement incarnée par les métamodèles AGR/AALAADIN [Ferber et al., 2004, Ferber and Gutknecht, 1998, Gutknecht, 2001], et MOISE [Hannoun et al., 2000], et les méthodologies MESSAGE et INGENIAS, ou encore TROPOS. La seconde quant à elle se focalise davantage sur la notion de norme et définit explicitement les politiques de contrôle ainsi que les règles à établir et à suivre. Elle est plutôt associée aux méthodologies GAIA, SODA, OMNI [Dignum et al., 2005] et plus globalement à la notion d'institution électronique [Esteva et al., 2001].

2.4.3 Comparatif entre les approches existantes

Selon [McFarlane and Bussmann, 2002], les systèmes holoniques ont besoin d'une méthodologie qui soit basée sur les principes de l'ingénierie logicielle pour assister le concepteur à chaque stade du processus de développement. Cette méthodologie doit fournir des lignes méthodologiques claires et non ambiguës pour l'analyse et la conception. Comme il fut mentionné précédemment, la notion d'agents composés n'est pas nouvelle dans les systèmes multi-agents. Il n'est donc pas surprenant que de nombreuses études aient été proposées pour modéliser et implanter ce type d'agent. Toutes ces approches n'utilisent pas le terme de *« holon »*, mais des similarités peuvent être trouvées dans les concepts sous-jacents. Dans la lignée du raisonnement de [McFarlane and Bussmann, 2002], cette thèse fournit une approche de modélisation exploitant à la fois les principes holoniques et ceux de l'ingénierie logicielle orientée-agent (approche organisationnelle). Cette section se propose d'étudier et de comparer quelqu'un des métamodèles et méthodologies existants dans ces deux domaines selon cinq critères de comparaison.

2.4.3.1 Critères de comparaison

Processus de développement : Ce premier axe d'étude se concentre sur l'analyse du processus de développement. Il vise à déterminer si le modèle de processus utilisé est clairement défini, et si la méthodologie étudiée couvre l'intégralité du processus de développement. Il détermine également si un ensemble de lignes méthodologiques est fourni pour guider l'ensemble de l'effort de production.

Concepts : Ce second critère se focalise sur l'étude du métamodèle et des abstractions sous-jacentes. Il vise notamment à déterminer si ce métamodèle supporte la notion de holon ou de composition d'agent et si la méthodologie adopte une approche organisationnelle pour l'étude du système.

Structure du système : Ce troisième point s'intéresse à la modélisation de la structure du système et doit déterminer si l'approche utilisée autorise la modélisation d'un

[20] Le lecteur peut se référer à [Stratulat, 2002, chap. 4] pour davantage de détails sur la notion de norme dans les SMA.

système à un nombre arbitraire de niveaux d'abstraction et si elle offre un large panel de structures possibles pour le système.

Relation Système-Environnement : L'environnement est un des concepts clefs dans les systèmes multi-agents [Michel, 2004, Odell et al., 2002, Weyns et al., 2006]. Il est donc important de considérer sa relation avec le système. Cet axe ne traite pas de la modélisation de l'environnement proprement dit, mais il cherche en revanche à déterminer comment l'approche étudiée permet de caractériser la limite entre le système et son environnement, et comment sont modélisées leurs interactions.

Respect des standards : Cet axe s'attache à déterminer si l'approche étudiée s'inscrit dans les standards actuels, notamment concernant la description de son processus et les langages de modélisation utilisés pour décrire ses produits.

Outils et Implantation : Ce dernier point vise à déterminer si dans l'approche étudiée, le processus de développement est assisté par un ensemble d'outils : AGL et plateforme d'implantation. En outre, selon [Dastani and Gomez-Sanz, 2005], les applications multi-agents ne pourront effectivement convaincre les industriels que si le fossé qui sépare, d'une part l'analyse et la conception des SMA, et d'autre part leur implantation est comblé. Cet axe tentera donc de déterminer si la transition entre les concepts utilisés pendant la conception et ceux disponibles dans l'implantation est relativement aisée ou non.

Enfin, les aspects relatifs à la réutilisation des modèles et des connaissances au sein de ces différentes approches sont également pris en compte. La suite de cette section est consacrée à la comparaison entre cinq approches considérées comme significatives pour les travaux défendus dans cette thèse.

2.4.3.2 Comparatif

AGR/AALAADIN [Ferber et al., 2004, Ferber and Gutknecht, 1998, Gutknecht, 2001]

i) Processus de développement : AGR est davantage un métamodèle qu'une véritable méthodologie. Par conséquent, il se concentre essentiellement sur les aspects de conception. Aucun processus de développement ne lui est véritablement associé ; les aspects liés à l'analyse et à l'implantation ne bénéficient d'aucunes lignes méthodologiques.

ii) Concepts : AGR est l'un des premiers métamodèles organisationnels pour les SMA. Les concepts qu'il introduit sont décrits dans le diagramme UML présenté sur la figure 2.3. Les concepts d'agent, de groupe et de rôle en constituent la base. Deux niveaux d'abstraction organisationnelle sont pris en compte : l'organisation et le groupe. Le groupe correspond à une instance concrète d'organisation et représente un ensemble d'agents partageant certaines caractéristiques communes. Un rôle est la représentation abstraite de la position fonctionnelle d'un agent dans un groupe et un

agent est une entité active, communicante, capable de jouer plusieurs rôles dans différents groupes. Aucune contrainte n'est imposée sur l'architecture des agents et sur leurs éventuels états mentaux. Deux agents ne peuvent communiquer que s'ils appartiennent à un groupe commun. Le principe de l'approche méthodologique associée, consiste à identifier les principaux groupes de l'application de sorte à déterminer la structure organisationnelle générale du système. Les aspects liés à la dynamique de cette structure, tels que la création dynamique des groupes ou encore la gestion de l'accès et de l'exclusion des agents au sein des groupes, sont ensuite spécifiés.

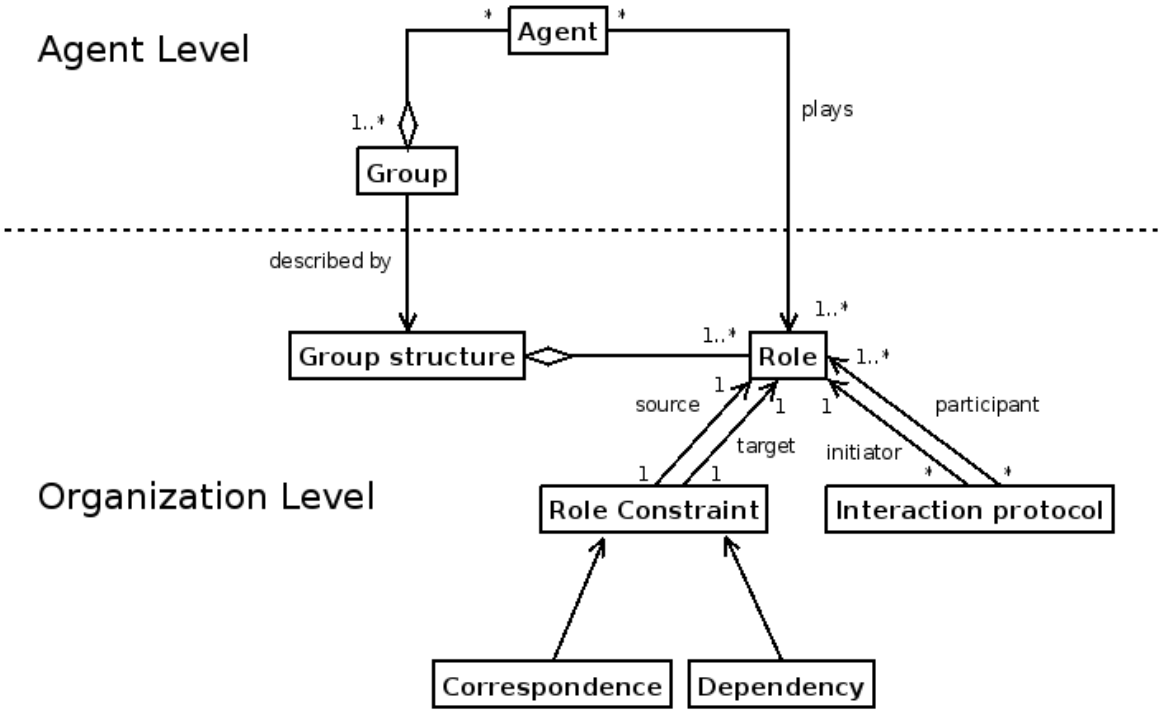


Figure 2.3: Le métamodèle AGR [Ferber et al., 2004]

Suivant une approche relativement similaire à celle d'AGR, MOISE[21] [Hannoun et al., 2000, Hübner et al., 2002] distingue les aspects structurel, fonctionnel et déontique dans la modélisation d'organisations. L'aspect structurel de MOISE correspond globalement à l'approche décrite dans AGR. Sur ce point, MOISE étend AGR notamment par la définition de la notion d'héritage et de composition entre rôles.

iii) Structure du système : AGR ne supporte pas la composition d'agents ou la notion de holon. En revanche l'approche organisationnelle qu'il adopte, permet de modéliser un système à différents niveaux d'abstraction, même si le métamodèle AGR ne fournit pas explicitement les outils nécessaires pour la définition des contributions entre des organisations situées à des niveaux d'abstraction différents.

iv) Relation Système-Environnement : AGR ne fournit pas de moyens pour modéliser l'environnement et identifier la limite entre le système et l'environnement.

[21] MOISE : Model of Organization for multI-agent SystEms

v) Respect des standards : [Ferber et al., 2004] n'étendent pas la sémantique d'UML, mais proposent leur propre ensemble de diagrammes spécifiquement dédiés à l'approche organisationnelle. Ils proposent notamment le diagramme *"cheese-board"* pour représenter les instances concrètes d'organisations, une représentation pour la structure des organisations ainsi qu'une version de diagramme de séquence organisationnelle. En termes de notations, AGR ne s'intègre donc pas dans les standards actuels.

vi) Outils et Implantation : AGR est associé à la plate-forme d'implantation Madkit[22] qui fournit les concepts de base nécessaires à l'implantation du modèle organisationnel. En revanche, la définition des rôles qu'elle propose ne correspond pas pleinement à celle utilisée durant la phase de conception. Dans Madkit, les rôles ne sont pas véritablement des comportements que les agents peuvent acquérir dynamiquement, mais sont réduits à de simples *étiquettes*. Cette approche des rôles nuit à la modularité et à la généricité des organisations. Madkit adopte une vision qui est davantage centrée sur la notion d'agent que ne l'est le métamodèle AGR. Sur cet aspect, la plate-forme MOCA [Amiguet, 2003, Amiguet et al., 2002] vient étendre Madkit et tente de corriger ce défaut d'implantation des notions de rôle et d'organisation et renforce également le contrôle d'accès à un rôle. MOCA considère l'organisation comme une véritable structure réutilisable.

GAIA [Wooldridge et al., 2000, Zambonelli et al., 2003]

i) Processus de développement : GAIA est probablement l'une des méthodologies multi-agents les plus connues, mais elle ne couvre que les phases d'analyse et de conception. Les différents modèles associés à cette méthodologie sont décrits dans la figure 2.4. GAIA considère que le système peut être vu comme une société ou une organisation d'agents. L'analyse est consacrée à l'identification des rôles et à la description de leurs interactions. La phase de conception s'attache ensuite à raffiner ces deux modèles préliminaires (rôle et interaction). Les rôles sont ensuite associés à des types d'agents pour créer des modèles d'agents. Pour chaque modèle le concepteur précise le nombre d'instances. L'ensemble des services fourni par chaque type d'agent est déduit des attributs des rôles et constitue le modèle de service.

ii) Concepts : Les rôles dans GAIA disposent de quatre attributs : responsabilités, droits, activités et protocoles. Les droits définissent les ressources accessibles aux rôles ainsi que les informations qu'ils peuvent lire, écrire ou créer. Les activités déterminent les tâches ou les actions qu'un rôle peut réaliser seul. Les protocoles détaillent comment satisfaire des tâches collectives nécessitant l'interaction entre plusieurs rôles. Les responsabilités d'un rôle définissent ses attributs et déterminent son comportement. GAIA distingue deux types de responsabilités : les propriétés de vivacité et de

[22] Madkit : `http://www.madkit.org`

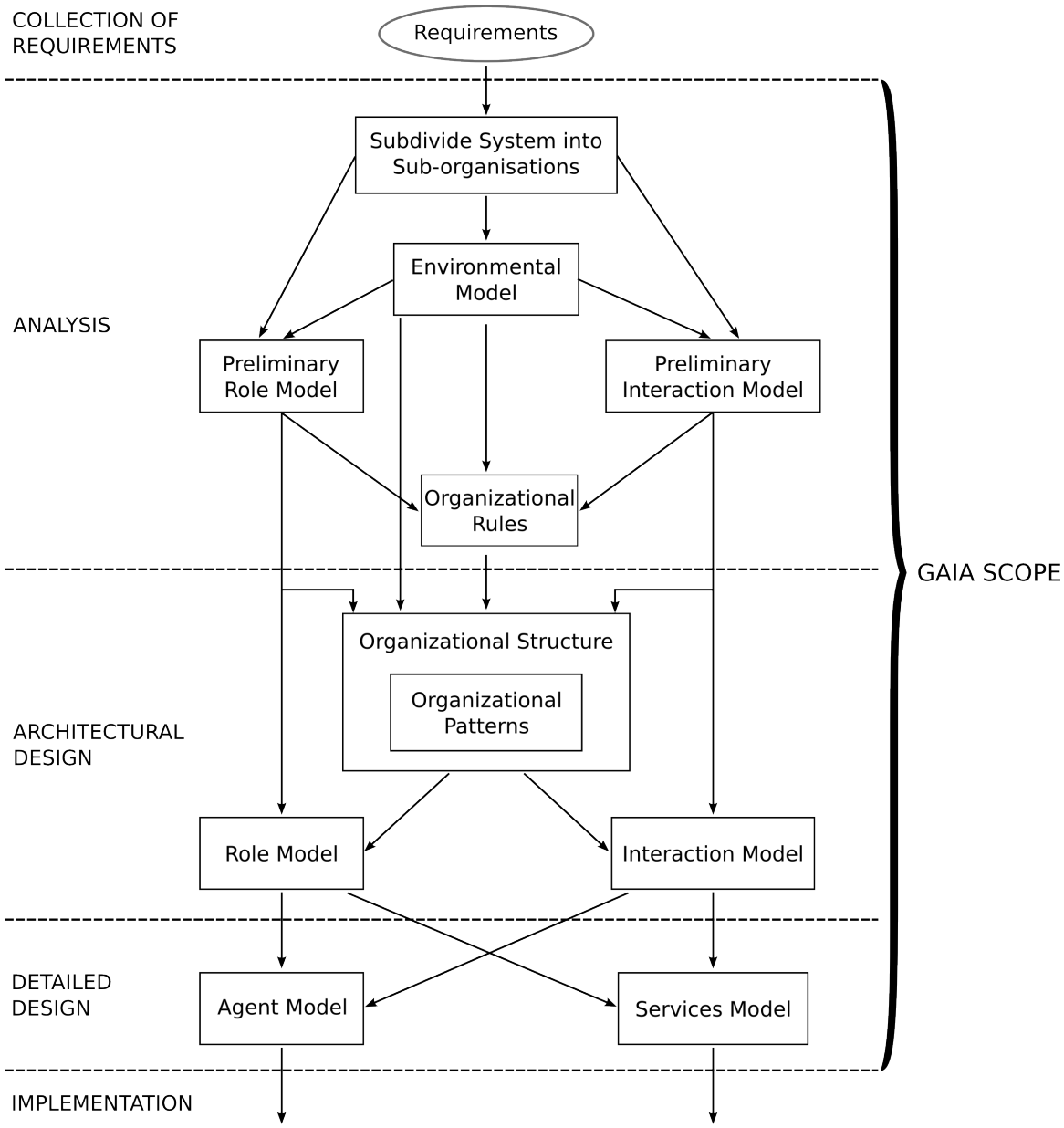


Figure 2.4: Les modèles associés à la méthodologie GAIA [Zambonelli et al., 2003]

sûreté (cf. section 2.4.1 page 38). À partir d'un modèle initial des rôles du système, un premier modèle d'interaction est développé. Il contient la définition du protocole de chaque interaction du système. Un protocole représente un schéma abstrait d'interaction et définit les éléments suivants : nom, initiateur, partenaires, entrées, sorties, description. Chaque service est décrit par quatre attributs : ses entrées, ses sorties, ses pré- et post-conditions.

Les rôles, qui représentent les responsabilités et les compétences des agents, ne sont pas concrétisés à la conception et à l'exécution. Ceci rend particulièrement difficile

toute vérification dynamique sur le comportement des agents durant l'exécution. Au niveau des agents, GAIA n'offre aucun mécanisme pour modéliser les notions de raisonnement dynamique, d'extension et de modification des responsabilités et des compétences des agents durant l'exécution.

iii) Structure du système : GAIA ne facilite pas la modélisation d'un système à plusieurs niveaux d'abstraction et le raffinement de ses composantes de manière itérative. Ceci est en partie dû au fait qu'elle n'autorise pas une décomposition récursive du système. Les rôles sont en effet des entités atomiques situées à un niveau d'abstraction donné et aucun mécanisme de composition des rôles n'est explicité.

La méthodologie ROADMAP [Juan et al., 2002] étend GAIA pour tenter de combler certaines de ses faiblesses quant à la représentation de l'environnement et des connaissances du domaine de l'application ainsi que la modélisation à plusieurs niveaux d'abstraction. Le cœur de la contribution de ROADMAP est incarné par la hiérarchie de rôles (version modifiée du modèle de rôle de GAIA). La construction de cette hiérarchie est basée sur le mécanisme d'agrégation de rôles. Un rôle de niveau n correspond à l'agrégation des rôles de niveau $n-1$. Un rôle composé représente une organisation ou une société de rôles. Cette vision de l'organisation en tant que rôle composé est également exploitée dans CRIO même si les définitions associées à ces deux concepts diffèrent largement.

iv) Relation Système-Environnement : Les informations relatives à l'environnement sont implicitement traduites dans les droits et les protocoles des rôles. GAIA ne fournit pas de modèle complet de l'environnement ce qui la rend inappropriée pour modéliser des applications en environnement hétérogène et dynamique [Juan et al., 2002].

v) Respect des standards : La modélisation dans GAIA est associée aux langages AUML et UML. Une description de son processus de développement a été effectuée à l'aide du formalisme SPEM par [A. Garro, 2004].

vi) Outils et Implantation : GAIA n'adresse pas la phase d'implantation, et ne fournit par conséquent aucune ligne méthodologique sur cet aspect. Elle n'est pas associée à une plate-forme d'implantation donnée ou à un AGL particulier. Les travaux de [Moraitis and Spanoudakis, 2006] présentent comment combiner l'approche méthodologique de GAIA pour l'analyse et la conception des SMA à l'utilisation de la plate-forme JADE pour leur implantation.

ROADMAP, quant à elle, est associée à l'AGL Rebel[23] [Kuan et al., 2005] pour assister les différentes phases de son processus de développement.

En terme de réutilisation des connaissances, GAIA ne présente pas de modèle complet de la structure des connaissances du domaine ainsi que des interactions et dépendances entre

[23] Rebel - Roadmap Editor Built for Easy deveLopment : `http://www.cs.mu.oz.au/agentlab/rebel.html`

ses composantes. Dans GAIA, les connaissances liées au domaine de l'application sont implicitement traduites dans les attributs des rôles. Ceci empêche que les connaissances du domaine et de l'application puissent aisément être partagées, réutilisées, étendues ou maintenues de manière modulaire. ROADMAP étend également GAIA sur ce point en ajoutant la notion de modèle de connaissances censé présenter une description holistique de connaissances du domaine du système. Ce modèle peut s'apparenter à la notion de modèle du domaine (*"Domain Model"*) utilisé dans la méthodologie MESSAGE.

ADELFE[24] [Bernon et al., 2002, Picard, 2004] — Atelier de Développement de Logiciels à Fonctionnalité Émergente — est une méthodologie spécialisée pour étudier les aspects adaptatifs et auto-organisateurs des SMA.

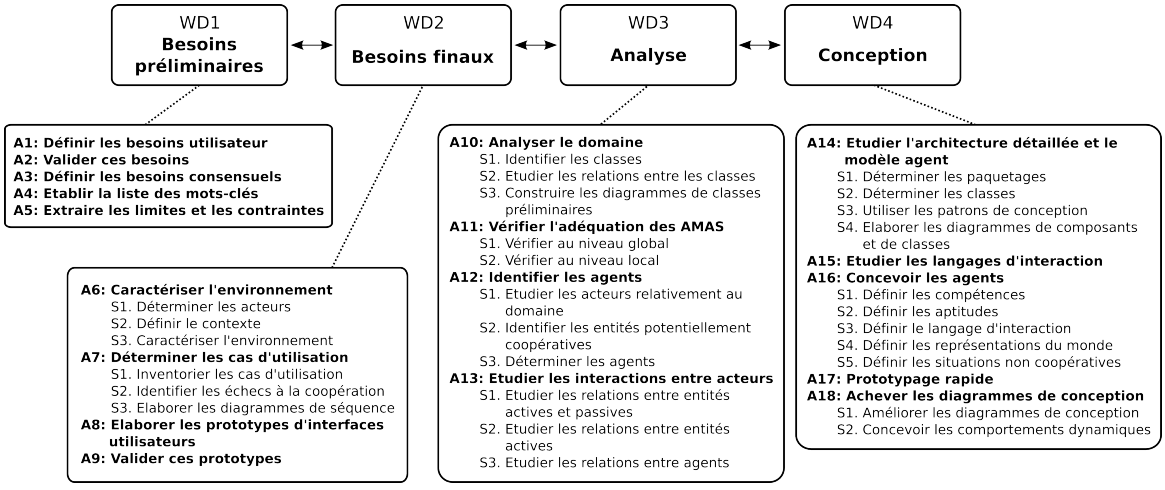


Figure 2.5: Le processus de la méthodologie ADELFE [Bernon et al., 2002, Picard, 2004]

i) Processus de développement : Le processus de développement d'ADELFE est également basé sur *"Unified Process"*. ADELFE couvre les phases d'analyse et de conception, les différentes étapes de son processus sont décrites dans la figure 2.5.

ii) Concepts : ADELFE n'est pas une méthodologie organisationnelle, l'organisation du SMA n'est pas explicitement spécifiée. Elle demeure essentiellement centrée sur le concept d'agent.

iii) Structure du système : ADELFE ne permet pas la modélisation d'un système à différents niveaux d'abstraction. Son processus de structuration du système est directement inspiré de l'orienté-objet. Il est basé sur l'exploitation des diagrammes de cas d'utilisation et de séquence.

iv) Relation Système-Environnement : Dès l'analyse des besoins, ADELFE consacre une activité de son processus à la caractérisation de l'environnement du système et

[24] ADELFE : `http://www.irit.fr/ADELFE/`

à l'identification des principales entités qui le composent ainsi qu'à la description de leur interaction avec le système.

v) Respect des standards : La modélisation dans ADELFE repose sur les langages de modélisation UML et AUML. La description de son processus est basée sur le formalisme SPEM. Cette méthodologie s'intègre donc dans les standards actuels aussi bien en termes de notations que de processus.

vi) Outils et Implantation : : Le travail de conception au sein d'ADELFE est assisté par l'outil *"OpenTool"* [Gleizes and Picard, 2003].

PASSI [25] [Chella et al., 2004, 2006, Cossentino and Potts, 2002, Cossentino et al., 2004, Cossentino, 2005]

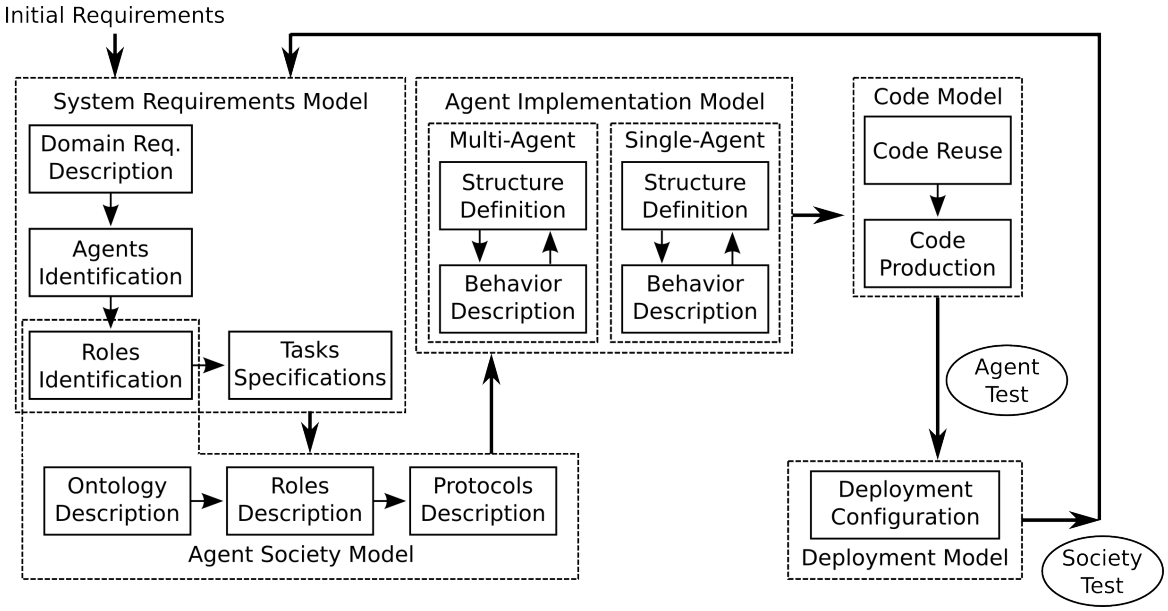


Figure 2.6: Les phases d'analyse et de conception de la méthodologie INGENIAS [Cossentino, 2005]

i) Processus de développement : PASSI fut l'une des premières méthodologies orientées-agent à couvrir l'intégralité du processus de développement depuis l'analyse des besoins jusqu'au déploiement.

ii) Concepts : PASSI n'est pas une méthodologie organisationnelle même si son métamodèle introduit le concept de rôle [Cossentino et al., 2005]. Le rôle est considéré comme une interface de communication pour l'agent. Le métamodèle de PASSI n'introduit pas le concept d'organisation.

iii) Structure du système : PASSI ne permet pas la modélisation d'un système à différents niveaux d'abstraction.

[25] PASSI : *"Process for Agent Societies Specification and Implementation"*.

iv) Relation Système-Environnement : L'environnement du système n'est pas explicitement modélisé dans PASSI. L'activité d'identification des rôles permet cependant de modéliser les interactions entre les agents du système et leur environnement. Les ontologies du problème et de sa solution multi-agents permettent quant à elles de rassembler les connaissances disponibles sur l'environnement du système.

v) Respect des standards : La modélisation dans PASSI repose sur les langages de modélisation UML et AUML. La description de son processus est basée sur le formalisme SPEM. Cette méthodologie s'intègre donc dans les standards actuels aussi bien en termes de notations que de processus.

vi) Outils et Implantation : Le travail de conception dans PASSI est assisté par l'outil PTK[26] [Chella et al., 2003]. L'outil Metameth est également proposé pour la création de nouveaux processus méthodologiques à partir de fragments de méthodes existantes [Cossentino et al., 2006a].

MESSAGE[27] et INGENIAS[28] [Caire et al., 2002, Pavón et al., 2005]

INGENIAS est une extension du projet MESSAGE, et vise à fournir un ensemble de méthodes et d'outils pour le développement de SMA.

		PHASES		
		Inception	Elaboration	Construction
WORKFLOWS	Analysis	- Generate use cases and identify actions of these ues cases with interactions models. - Sktech a system architecture with an orgfanization model. - Generate environment models to represent results from requirement gathering stage.	- Refined use cases. - Agent models that detail elements of the system architecture. - Workflows and tasks in organization models. - Models of tasks and goals to highlight control constraints (main goals, goal decomposition). - Refinements of environment model to include new environment elements.	- Refinements on existing models to cover uses cases.
	Design	- Generate prototypes perhaps with rapid application development tool such as Zeus or Agent Tools.	- Refinements in workflows. - Interactions models that show how tasks are executed. - Models of tasks and goals that reflect dependencies and needs identified in workflows and how system goals are achieved. - Agent models to show required mental state patterns.	- Generate new models. - Social Relationships that perfects organization behaviour.

Figure 2.7: Les phases d'analyse et de conception de la méthodologie INGENIAS [Pavón and Gómez-Sanz, 2003]

i) Processus de développement : INGENIAS couvre l'intégralité du processus de développement et la définition de son processus est basée sur *"Unified Process"* [Jacobson

[26] PTK : *"PASSI ToolKit"*, un plug-in pour l'outil IBM Rational Rose.

[27] MESSAGE- Methodology for Engineering Systems of Software AGEnts : `http://www.eurescom.de/~public-webspace/P900-series/P907/index.htm`

[28] INGENIAS : `http://grasia.fdi.ucm.es/ingenias`

et al., 1999]. La figure 2.7 détaille les phases d'analyse et de conception de cette méthodologie. INGENIAS considère cinq perspectives sur le système à étudier et associe à chacune d'elle un modèle particulier :

- Le modèle d'organisation qui décrit la structure générale du système et précise le nombre et les différents types d'agents du système. Les *« relations de pouvoir »* entre les agents sont également décrites. Ce modèle correspond à une vision externe sur le système.
- Le modèle des objectifs et tâches qui définit les objectifs que le système et les agents devront satisfaire.
- Le modèle d'agent qui détaille chaque agent et chaque rôle du système, mais également les objectifs de chaque agent et les services qu'il fournit. Ce modèle correspond à la vision interne du système.
- Le modèle du domaine collecte l'ensemble des connaissances acquises sur le domaine de l'application sous la forme d'une ontologie.
- Le modèle d'interaction qui décrit la manière dont les agents communiquent et les protocoles d'interaction associés.

L'approche adoptée par INGENIAS pourrait être qualifiée de multi-vues ou multi-perspectives. De façon à réduire la complexité du système, celui-ci est tour à tour étudié selon différentes perspectives qui agissent tels des filtres et permettant au concepteur de se concentrer sur un aspect particulier du système à la fois. En cela INGENIAS adopte une approche similaire à d'autres méthodologies telles que MASSIVE [Lind, 2001] (7 vues : environnement, tâche, rôle, interaction, société, architecture et système) ou MAS-CommonKADS[29] [Iglesias et al., 1996, 1997] (7 vues : organisation, tâches, expérience, agents, communications, coordination, et conception).

ii) Concepts : INGENIAS suit une approche de développement dirigée par modèle[30] où un métamodèle est associé à chacune des perspectives précédentes. Dans INGENIAS, un agent est considéré comme une entité intentionnelle suivant le principe de rationalité de [Newel, 1982]. La définition de la notion d'organisation au sein d'INGENIAS est à mi-chemin entre les deux grandes tendances qui gouvernent les approches organisationnelles. L'organisation est considérée à la fois comme une structure permettant la décomposition du système, mais également comme un support pour la définition des normes et des relations entre agents, groupes et organisations. Elle définit ainsi une structure, une fonctionnalité et des relations sociales qui permettent la décomposition d'un SMA en termes de groupes et de *"workflow"*. En cela INGENIAS enrichit le concept d'organisation tel qu'il est défini dans AGR et lui apporte davantage de dynamisme.

iii) Structure du système : INGENIAS n'intègre pas la notion de holon ou la composi-

[29] CommonKADS : `http://www.commonkads.uva.nl`
[30] MDD : *"Model Driven Development"* [Schmidt, 2006]

tion entre agent. En revanche tout comme AGR, l'approche organisationnelle qu'il adopte permet de modéliser un système à différents niveaux d'abstraction. Là encore, la modélisation des contributions entre des organisations situées à des niveaux d'abstraction différents ou les notions de composition de rôles ou de décomposition hiérarchique de rôles ne sont pas prises en compte.

iv) Relation Système-Environnement : INGENIAS dédie une perspective complète à la modélisation de la relation entre l'environnement et le système. Elle vise à déterminer les entités qui interagissent avec le SMA. L'environnement est décomposé en deux éléments, d'une part la catégorisation des entités considérées comme pertinentes et d'autre part les restrictions des interactions avec ces dernières. L'environnement se compose d'applications, d'agents et de ressources.

v) Respect des standards : L'intégration des standards de l'ingénierie logicielle est l'un des principes de base de INGENIAS. Le processus de modélisation est notamment associé aux langages AUML et UML. Le processus de développement a été décrit à l'aide du formalisme SPEM par M. Cossentino.

vi) Outils et Implantation : Pour l'implantation des SMA, INGENIAS ne fait pas de pré-supposition sur l'utilisation d'une plate-forme d'implantation particulière. En revanche INGENIAS est associée à un AGL : IDK[31], pour assister les différentes phases du processus de développement. Suivant une approche MDD, INGENIAS sépare le modèle SMA du modèle d'implantation, ce dernier permettant de décrire comment réaliser les concepts agent dans une plate-forme spécifique. Une transformation entre ces deux modèles est ensuite définie pour faciliter la génération de code. IDK permet notamment d'obtenir un modèle computationnel pour la plate-forme JADE[32] [Bellifemine et al., 2001].

ANEMONA [Giret, 2005, Giret and Botti, 2003, 2005, 2006] est la première méthodologie multi-agents, au sens entendu dans l'ingénierie logicielle, qui fut proposée pour les systèmes holoniques. Elle est spécialisée dans le domaine des systèmes holoniques manufacturiers.

i) Processus de développement : ANEMONA est inspirée de la méthodologie INGENIAS et adopte les mêmes perspectives sur le système. Le processus de développement associé à cette méthodologie est décrit dans la figure 2.8. La figure 2.9 détaille plus spécifiquement les phases d'analyse et de conception. ANEMONA contribue en effet essentiellement à ces deux phases et vient notamment étendre INGENIAS pour intégrer les différentes activités liées à l'identification, la conception et l'implantation de la notion de holon.

[31] IDK - Ingenias Development Kit : `http://ingenias.sourceforge.net/`
[32] JADE — Java Agent DEvelopment framework :`http://jade.cselt.it`

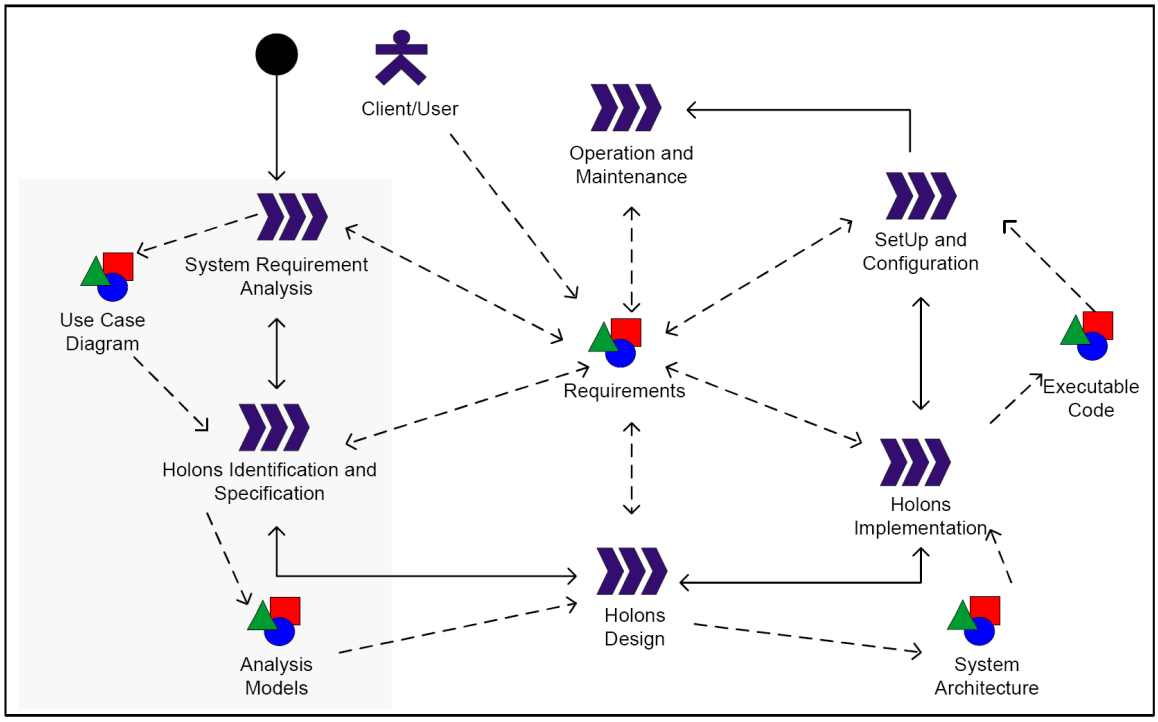


Figure 2.8: Le processus de la méthodologie ANEMONA [Giret et al., 2005]

ii) Concepts : Le métamodèle d'ANEMONA définit le concept d'*"Abstract Agent"* pour la modélisation des holons et la composition entre agents (cf. figure 2.10). Concernant l'implantation des holons, cette méthodologie réutilise les différentes architectures de holons proposées dans le modèle PROSA [Brussel et al., 1998, Wyns, 1999]. Bien qu'organisationnelle, car basée sur INGENIAS, ANEMONA adopte tout de même une approche centrée sur l'agent, et son architecture, à la fin du processus de conception.

iii) Structure du système : ANEMONA permet la modélisation d'un système à un nombre arbitraire de niveaux d'abstraction sur la base du concept d'*"Abstract Agent"*.

iv) Relation Système-Environnement : Même approche que INGENIAS.

v) Respect des standards : Le modèle de processus de cette méthodologie est *"Unified Process"*, et sa description est basée sur le formalisme [SPEM, 2002]. Sur cet aspect, elle s'intègre dans les standards actuels. En revanche, elle introduit un ensemble de notations spécifiques pour assister le processus de modélisation et représenter les concepts de rôle, d'agent, de but, de groupe, etc.

vi) Outils et Implantation : ANEMONA fournit un ensemble de lignes méthodologiques pour faciliter l'implantation de ses modèles sur la plate-forme multi-agents JADE.

Après ce bref tour d'horizon de quelques-unes des méthodologies orientées-agent existantes, la suite de ce chapitre se concentre sur l'étude des plates-formes disponibles pour l'implantation des systèmes multi-agents.

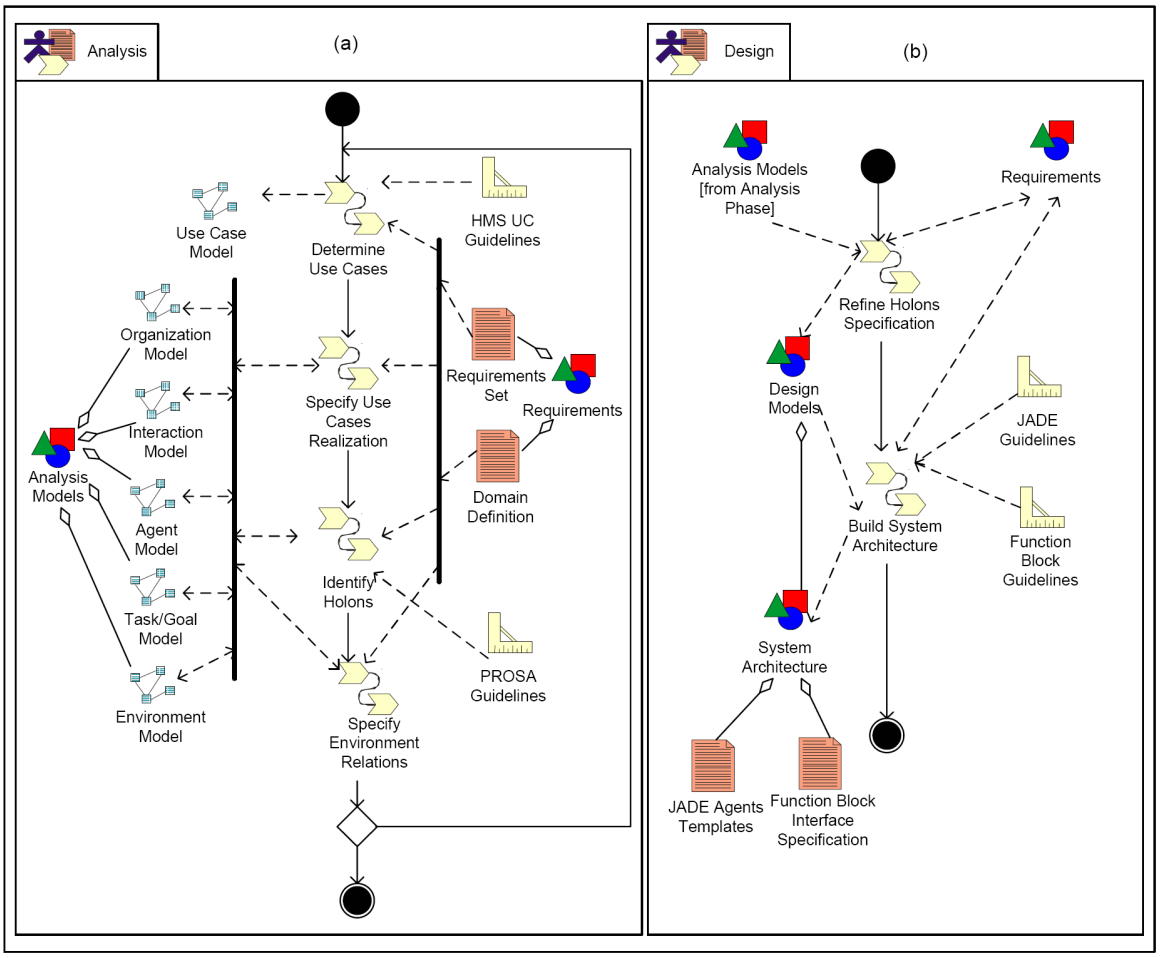


Figure 2.9: Les phases d'analyse et de conception de la méthodologie ANEMONA [Giret et al., 2005]

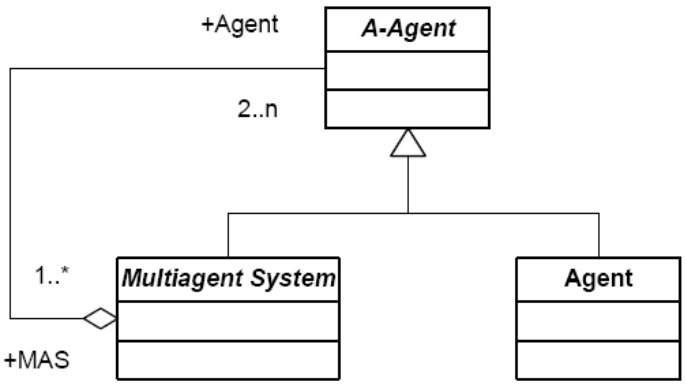


Figure 2.10: La notion d'agent abstrait récursif dans ANEMONA [Giret and Botti, 2003]

2.5 Implantation des systèmes multi-agents

Cette section se focalise sur l'implantation des SMA et récapitule les deux standards majeurs qui gouvernent ce domaine. Elle propose ensuite une classification des principaux types de plates-formes existantes.

2.5.1 Standardisation

Les systèmes multi-agents peuvent être implantés et déployés sur une grande variété de plates-formes. Seuls deux véritables standards existent dans le domaine des plates-formes multi-agents. Ils tentent de fournir une description générale des fonctionnalités de base à fournir pour faciliter l'interopérabilité des plates-formes.

Le standard FIPA[33] fixe notamment les composantes et services essentiels qu'une plate-forme doit fournir. Mais il définit avant tout l'interface minimale que doit exhiber une plate-forme pour faciliter son interopérabilité, la structure des messages et de leur enveloppe ainsi que les protocoles de transport de messages. La FIPA standardise également des aspects de plus haut niveau relatifs à la communication entre agents dont notamment les protocoles et langages d'interaction, les actes de langages et leurs représentations.

L'autre standard [MAF, 2000][34] est fourni par l'OMG et place CORBA au cœur de l'interopérabilité entre les plates-formes d'agents mobiles. Les éléments de la standardisation MAF traitent de la gestion des agents, des protocoles de transfert, de l'identification des agents et de leur architecture générale, ainsi que des interfaces des plates-formes (*MAFFinder* : localisation des agents et *MAFAgentSystem* : gestion du cycle de vie). Ces différents éléments sont regroupés sous forme de services CORBA (nommage, cycle de vie, externalisation et securité) interrogeables par les agents.

Le standard FIPA est implanté par bon nombre de plates-formes telles que JADE [Bellifemine et al., 2001] par exemple. En revanche, rares sont les plates-formes qui satisfont ces deux standards, l'une des plus connues est la plate-forme commerciale GrassHopper [Bäumer and Magedanz, 1999, Bäumer et al., 2000].

2.5.2 Plates-formes d'implantation des systèmes multi-agents

Un tour d'horizon de quelques-unes des plates-formes multi-agents existantes est présenté en annexe de ce document (cf. section B page 223). Trente plates-formes y sont rapidement exposées, mais cette présentation ne détaille pourtant qu'une toute petite partie des plates-formes qui existent actuellement, près de deux cents ont en effet été répertoriées. Il est dès lors relativement difficile de les classifier. Quelques catégories majeures peuvent toutefois être discernées :

[33] FIPA, Foundation for Intelligent Physical Agents : `http://www.fipa.org`

[34] MAF, Mobile Agent Facility : `http://www.omg.org/technology/documents/formal/mobile_agent_facility.htm`

- Les approches orientées-langage qui proposent de nouveaux langages orientés-agent ou plus généralement étendent des langages existants (Java, SmallTalk) pour y intégrer le concept d'agent. Les langages déclaratifs tels que CLAIM[35] [Seghrouchni and Suna., 2004], DALI [Costantini and Tocchio, 2002] ou ReSpecT [Omicini and Denti, 2001], sont généralement distingués des langages impératifs tels que Jack [Hodgson et al., 1999, Howden et al., 2001]. Des approches hybrides combinant les deux aspects existent également dans la littérature telles que Jason [Bordini et al., 2005] ou IMPACT [Subrahmanian et al., 2000].
- Les plates-formes pour agents mobiles ou agents *"webs"*, telles que JADE [Bellifemine et al., 2001] ou AgentBuilder [Reticular, 1999]. Plusieurs d'entre elles étendent des approches existantes telles que CORBA, RMI ou IIOP destinées initialement à l'orienté-objet, l'orienté-service ou aux composants : GrassHopper [Bäumer and Magedanz, 1999], Aglet [Clements et al., 1997, Lange and Mitsuru, 1998] ou Able [Bigus et al., 2002].
- Les plates-formes spécifiquement dédiées à la simulation telles que Swarm [Minar et al., 1996], Cormas [Bousquet et al., 1998] ou Mobidyc [Ginot and Le Page, 1998, Ginot et al., 2002].
- Enfin, les plates-formes généralistes qui s'intègrent le plus souvent dans un projet global visant à fournir un métamodèle, une méthodologie et l'ensemble des outils nécessaires pour leur mise en œuvre (AGL et plate-forme) telles que MAST [Boissier et al., 1998, Vercouter, 2004], Geamas [Marcenac, 1997, Marcenac and Giroux, 1998, Soulie et al., 1998] ou encore Madkit [Gutknecht and Ferber, 2000, Gutknecht et al., 2000, Gutknecht, 2001].

Parmi toutes ces plates-formes, on peut tout de même constater l'émergence de la plate-forme JADE. En effet, JADE est aujourd'hui considérée comme la référence européenne en terme de satisfaction du standard FIPA et de nombreux projets ont été développés sur sa base ou ont contribué à ses extensions [Bernon et al., 2005].

2.6 Discussions

Dans ce chapitre, un tour d'horizon des différentes méthodologies et métamodèles dans le domaine de l'ingénierie multi-agents a été effectué. Devant cette pléthore de modèles, il est intéressant de noter qu'à l'heure actuelle aucune méthodologie ne se dégage véritablement comme un standard pour le développement des SMA. La plupart des méthodologies existantes manquent encore de maturité et bénéficient de peu d'expériences industrielles. Une tentative de standardisation des méthodologies orientées-agent est actuellement menée par le comité technique sur les méthodologies de la FIPA[36].

[35] Computational Language for Autonomous, Intelligent and Mobile Agents

[36] FIPA Methodology Technical Committee : `http://www.pa.icar.cnr.it/~cossentino/FIPAmeth/`

Les systèmes multi-agents holoniques ne disposent actuellement pas d'une méthodologie complète et indépendante de tout type d'architecture d'agent, seule ANEMONA [Giret, 2005] adopte une approche véritablement inspirée de l'ingénierie logicielle dans ce domaine. Mais ANEMONA se focalise uniquement sur les systèmes holoniques manufacturiers, et la vision holonique dans ce domaine diffère de celle utilisée dans les SMA. De plus, ANEMONA reste fortement dépendante des architectures de holon définies dans PROSA [Brussel et al., 1998, Wyns, 1999]. Les approches à base d'agents classiques peuvent bien sûr être adaptées à ce domaine, mais au prix d'efforts coûteux en terme de développement ou de concessions sur la modularité et la réutilisation des modèles.

Du point de vue de l'ingénierie logicielle, la plupart des méthodologies actuelles se focalisent essentiellement sur les phases d'analyse et de conception. Peu d'entre elles correspondent véritablement à la définition de méthodologies en tant que telles, fournissant une description complète de son modèle de processus, de son processus en lui-même et de ses différents produits [Cernuzzi et al., 2005]. Chacune des étapes composant le processus doit également bénéficier de lignes méthodologiques claires pour guider le travail des différentes personnes impliquées dans le développement du logiciel.

Dans l'analyse et la conception, les approches organisationnelles offrent de nombreux avantages : hétérogénéité des langages, modularité, multiples architectures, un panel large d'applications, et une plus grande sécurité des applications (cf. section 2.4.2, page 39). Cependant, bon nombre des méthodologies existantes adoptent l'approche organisationnelle durant l'analyse et reviennent à une approche centrée-agent au fil de la conception, pour obtenir une implantation complètement centrée-agent, perdant ainsi une grande partie des avantages de l'approche organisationnelle. Ceci peut aisément être expliqué par l'absence de plate-forme multi-agents véritablement organisationnelle permettant d'implanter la notion de rôle en tant qu'entité de premier ordre et autorisant ainsi une dynamique importante pour les rôles d'un agent.

Dans l'implantation, JADE semble se dégager comme le standard, au moins en Europe. Concernant les langages de modélisation, UML est clairement considéré comme le standard. Diverses extensions existent, mais elles ne couvrent pas encore l'intégralité des besoins de la modélisation Agent et encore moins ceux de l'approche organisationnelle.

Quelques-unes des méthodologies existantes ont été étudiées pour tenter de déterminer les principes clefs qui régissent leur processus et la manière dont elles modélisent un système. Les résultats de ce comparatif sont résumés sur la figure 2.6. Au sortir de cet état de l'art et en réunissant les avantages de ces différentes approches, quelques principes peuvent être dégagés pour une méthodologie orientée-agent. Ces différents principes sont résumés ci-dessous :

Principes en terme de processus : Une méthodologie doit couvrir l'intégralité du processus de développement logiciel. En outre, le modèle de son processus doit être spécifié :

1. l'organisation et la coordination des différentes phases et sous-phases du développement ;
2. les produits fournis et le (ou les) langage(s) utilisé(s) pour les décrire ;
3. les différentes personnes qui doivent intervenir dans chacune de ces phases et à quel moment ;
4. les technologies et outils qui doivent être utilisés dans chaque activité ;
5. les ressources impliquées dans chaque phase du processus.

Chaque étape du processus doit être associée à un ensemble de lignes méthodologiques décrivant le travail à effectuer.

Principes en terme de métamodèle Chaque méthodologie doit disposer d'un métamodèle définissant les abstractions nécessaires à la modélisation du problème et de sa solution. En outre, pour modéliser des systèmes complexes, la modélisation du système à plusieurs niveaux d'abstraction apparaît comme un pré-requis essentiel. De plus, la modélisation à différents niveaux d'abstraction facilite d'autant le processus itératif de raffinement des différents modèles du système [Juan et al., 2002]. Le métamodèle correspondant doit permettre une telle modélisation. Dans les approches organisationnelles, le principe de composition de rôles semble essentiel pour permettre la décomposition du système et sa modélisation à différents niveaux d'abstraction.

Principes en terme d'implantation : Une méthodologie doit être associée à une plate-forme logicielle permettant l'implantation des concepts utilisés dans les phases d'analyse et de conception. Sur ce point, une approche MDD, comme celle proposée dans INGENIAS, permet de disposer d'un ensemble de transformations pour obtenir un modèle opérationnel et faciliter ainsi la transition depuis la conception.

De manière plus générale, une méthodologie se doit de favoriser la modularité et la réutilisation de ses modèles et éventuellement encourager l'utilisation des schémas de conception (*"design pattern"*).

Dans cette perspective, les connaissances liées au domaine de l'application doivent être explicitement modélisées et regroupées afin d'être aisément étendues, réutilisées et maintenues de manière modulaire. De même, l'environnement du système, ou plus généralement les informations relatives à celui-ci, doivent également être explicitement modélisées pour favoriser le déploiement des applications *agent* sur des environnements dynamiques et hétérogènes.

C'est cet ensemble de principes, qui est à la base de la conception du métamodèle CRIO, de la méthodologie ASPECS et de la plate-forme *Janus*, et qui ont été combinés avec l'héritage et l'histoire de l'approche inspirée du métamodèle RIO, de son extension holonique et de la méthodologie PASSI [Chella et al., 2004, 2006, Cossentino et al., 2004, Cossentino, 2005].

	AGR	INGENIAS	ANEMONA	GAIA	ROADMAP	ADELFE	PASSI
Modèle et Standard de processus	-	UP et MDD, SPEM	UP, SPEM	-, SPEM	-	UP, SPEM	SPEM
Cycle de vie	Conception	Complet	Complet	Analyse et Conception	Analyse et Conception	Analyse et Conception	Complet
Holonique ?	Non	Non	Oui	Non	Non	Non	Non
Organisationnelle ? Norme et/ou structure	Structure	Normes et Structure	Normes et Structure	Normes	Normes	Non	Non
Nombre de niveau d'abstraction dans la modélisation	n (Partiellement)	n (Partiellement)	n	1	n	1	1
Nombre de niveau d'abstraction dans l'implantation	1	1	n	1	1	1	1
Environnement	Non	Oui, 1 perspective dédiée	Oui	Oui, mais noyé dans les modèles de rôles	Oui	Oui	Oui mais noyé dans l'ontologie et les modèles de rôles
Modèle de connaissances	Non	Oui	Oui	Non	Oui	Oui	Oui
Standard Notations	-	UML et AUML	-	UML et AUML	UML et AUML	UML et AUML	UML et AUML
AGL	-	IDK	-	-	Rebel	OpenTool	Metameth et PTK
Plate-forme	Madkit	Outils pour JADE	Guide pour JADE	Guide pour JADE	-	-	Outils pour JADE

Table 2.1: Comparaison de quelques métamodèles et méthodologies orientés-agent et holoniques

DEUXIÈME PARTIE

Les systèmes multi-agents holoniques : De l'analyse à l'implantation de systèmes complexes

« Nous ne raisonnons-que sur des modèles »

Paul Valéry

Oeuvres (2t.) et Cahiers (2t.),

Gallimard, Bibliothèque de la Pléiade

« Mais comment élaborons-nous les modèles sur lesquels nous raisonnons ?

Les modèles, c'est-à-dire les représentations intelligibles artificielles, symboliques, des situations dans lesquelles nous intervenons : modéliser c'est à la fois identifier et formuler quelques problèmes en construisant des énoncés, et chercher à résoudre ces problèmes en raisonnant par des simulations. En faisant fonctionner le modèle énoncé, on tente de produire des modèles solutions. Modélisation et simulation, réflexion et raisonnements, sont les deux faces inséparables de toute délibération. »

Jean-Louis Le Moigne

La modélisation des systèmes complexes,

Dunod, 4ième édition, 1999, p. 15

CHAPITRE 3

CRIO : UN MÉTAMODÈLE HOLONIQUE POUR L'ANALYSE ET LA CONCEPTION DE SYSTÈMES COMPLEXES

CE CHAPITRE EST CONSACRÉ À LA PRÉSENTATION D'UN MODÈLE ÉNONCÉ ou métamodèle pour la conception de systèmes complexes. Il se base sur l'approche holonique et la décomposition organisationnelle hiérarchique du problème traité. Le chapitre 4 tentera de fournir un guide pour produire des modèles solutions à partir du modèle énoncé proposé dans ce chapitre.

Sommaire

3.1 Introduction

Construire une solution multi-agents, qui satisfasse les objectifs d'une problématique passe d'abord par une phase d'analyse et de modélisation. De nombreuses approches d'analyse existent déjà telles que les approches fonctionnelles, orientées composant, orientées objet, orientées service, etc. À chacune d'entre elles est associée au moins une méthode ou une méthodologie d'analyse et de conception. Par exemple, l'approche fonctionnelle est fréquemment associée aux méthodes SA, SART et SADT. L'approche objet, quant à elle, est généralement associée aux méthodologies Booch, HOOD, OMT, OOSD, OOSA, OOSE, etc. L'approche multi-agents, comme le décrit le chapitre précédent, dispose elle aussi d'une grande variété de méthodologies pour faciliter sa mise en œuvre.

Les méthodologies d'analyse et de conception ont pour objectif de guider les étapes préliminaires du développement d'un système afin de rendre le produit de ce développement plus fidèle aux besoins des clients. Le principe général des méthodes et des méthodologies précédentes repose sur la décomposition (fonctionnelle ou orientée-objet) du système modélisé, et sur la description de cette décomposition. Nombre d'entre elles sont basées sur un métamodèle qui fournit un ensemble d'abstractions et de concepts permettant de décrire le système. Elles sont généralement alliées à un langage de modélisation dont le standard est UML pour les méthodologies orientées-objet.

Concevoir une méthodologie exige l'élaboration d'un métamodèle, le choix ou la création d'un langage de modélisation et la détermination d'un ensemble codifié et reproductible d'étapes à suivre pour concevoir un logiciel. Pour supporter et faciliter le processus de conception, des outils tels que les Ateliers de Génie Logiciel (AGL) s'avèrent de plus en plus indispensables. De même, disposer de plates-formes de développement et de déploiement, basées sur des métamodèles proches de ceux utilisés durant la conception, facilite la phase d'implantation et limite le risque d'erreurs dues à la transition entre des métamodèles différents.

Un métamodèle constitue la base de la conception d'une méthodologie et doit fournir les abstractions nécessaires pour décrire un système. Pour ce faire, les concepts sociologiques ont toujours été une source d'inspiration pour les recherches dans le domaine des systèmes multi-agents. Plus récemment, des travaux sur les SMA ont inversé cette tendance en explorant le potentiel de leurs modèles pour étudier des phénomènes sociologiques [Carley and Prietula, 1994, Epstein and Axtell, 1996, Gilbert and Troitzsch, 2005, Prietula et al., 1998]. Le résultat de ces interactions a été la formalisation d'un certain nombre de concepts sociologiques, psychologiques et philosophiques avec des applications importantes dans l'analyse et la conception des SMA. Le concept de *Holon* et les concepts organisationnels de *Rôle*, *Organisation*, ou *Groupe* en sont probablement les exemples les plus frappants. De nombreux métamodèles et méthodologies ont déjà été proposés pour les SMA [Bernon et al., 2005, Iglesias et al., 1999] et certains d'entre eux adoptent clairement une vision organisationnelle : AGR [Ferber and Gutknecht, 1998], RIO [Hilaire et al., 2000], MOCA

[Amiguet, 2003] pour les métamodèles, et les méthodologies telles que GAIA [Zambonelli et al., 2003], INGENIAS [Pavón et al., 2005], ISLANDER [Sierra et al., 2004], MESSAGE [Caire et al., 2002], ou encore SODA [Omicini, 2000].

L'approche décrite dans ce document se propose de combiner les points de vue organisationnel et holonique pour observer, analyser et décomposer un système. L'objectif recherché vise à associer les aspects modulaires de l'approche organisationnelle avec les prédispositions intrinsèques de l'approche holonique pour la modélisation des systèmes complexes et hiérarchiques. L'approche proposée pour la modélisation est basée sur la décomposition hiérarchique et organisationnelle d'un système. Le système est décomposé niveau par niveau et les organisations qui composent chaque niveau sont ensuite identifiées et spécifiées. L'organisation est considérée comme un module de conception réutilisable et adaptable. Pour supporter cette approche, ce chapitre introduit un métamodèle organisationnel et holonique (nommé CRIO[1]) pour l'analyse et la conception de systèmes complexes de grande échelle[2].

Ce chapitre est organisé de la manière suivante : après une description générale du métamodèle CRIO (section 3.2), les différents concepts qu'il définit, sont décrits dans les sections 3.3 et 3.4. Les principales contributions du métamodèle sont finalement récapitulées à la section 3.5.

3.2 Présentation générale

3.2.1 Motivations

L'approche organisationnelle est désormais considérée comme une approche adaptée à l'analyse et à la conception des systèmes complexes. En effet, les méthodologies orientées-agent ont évolué depuis une vision initiale où le système était essentiellement centré sur l'agent et ses aspects individuels, vers une vision où il est désormais considéré comme une organisation dans laquelle les agents forment des groupes et des hiérarchies, et suivent des règles et des comportements spécifiques [Argente et al., 2006]. L'évolution des méthodologies GAIA [Wooldridge et al., 2000, Zambonelli et al., 2003] et TROPOS [Giunchiglia et al., 2002, Kolp et al., 2006] en sont d'ailleurs les exemples les plus évidents. Dans leur majorité, les méthodologies orientées-agent admettent qu'un SMA puisse être conçu comme une société organisée d'individus dans laquelle chaque agent joue des rôles spécifiques et interagissent avec d'autres agents [Jennings, 2000, Zambonelli et al., 2003].

[1] CRIO signifie : Capacity Role Interaction Organization

[2] *"Large complex systems"*, Un système est parfois considéré comme de grande échelle s'il peut être divisé ou partitionné en un certain nombre de sous-systèmes d'échelle plus petite. Un autre point de vue considère qu'un système est de grande échelle, si sa complexité est trop importante pour que les techniques conventionnelles de modélisation et de commande fournissent des résultats satisfaisants en un temps raisonnable [Jamshidi, 2003].

Cependant, dans ce contexte, modéliser le fait qu'un groupe d'agents en interaction exhibe, à un certain niveau d'abstraction, un comportement global spécifique et qu'en même temps, ses membres puissent se comporter comme des entités individuelles partiellement indépendantes, demeure un problème récurrent. De nombreux travaux ont déjà étudié cette question, et plusieurs modèles ont été proposés dans des domaines très variés. Les systèmes holoniques offrent une approche permettant de faire cohabiter ces différents niveaux d'abstraction au sein d'un même système. En effet, l'une des propriétés les plus intéressantes des systèmes holoniques, mais qui constitue également l'essence même de leur complexité, est qu'un holon peut être à la fois une entité et un ensemble d'organisations. Même si l'idée d'agent holonique est déjà largement répandue dans la communauté multi-agents, de nombreux métamodèles considèrent encore les agents comme des entités atomiques, les rendant de fait inappropriées aux systèmes multi-agents holoniques. Les métamodèles intégrant la notion de holon sont quant à eux généralement associés à un domaine d'application particulier tel que les systèmes holoniques manufacturiers par exemple.

Ce besoin d'un métamodèle, pour la modélisation holonique de systèmes considérés comme complexes, constitue la principale motivation pour l'élaboration d'une nouvelle méthodologie et de la plate-forme qui lui est associée. L'approche organisationnelle est utilisée pour faciliter la modélisation holonique et encourager la modularité et la réutilisation des modèles. CRIO diffère essentiellement des autres métamodèles organisationnels par la manière dont il définit le concept de rôle. En effet, le *rôle* est considéré comme une entité fondamentale, présente depuis l'analyse jusqu'à l'implantation. Or peu de métamodèles [Amiguet, 2003, Durand, 1996] considèrent le concept de rôle comme l'abstraction d'un comportement qui peut être défini indépendamment de l'entité qui le joue. De plus, certaines des plates-formes d'implantation les plus connues ne supportent tout simplement pas le concept de rôle : Jade [Bellifemine et al., 2001], FIPA-OS [Poslad et al., 2000] pour n'en citer que quelques-unes. Cette vision spécifique du concept du rôle, et son impact sur la modélisation organisationnelle d'un système, constituent la seconde motivation qui justifie la création du processus méthodologique ASPECS, mais également de la plate-forme *Janus*.

3.2.2 Présentation du métamodèle CRIO

CRIO est un métamodèle organisationnel destiné à la modélisation de systèmes complexes ouverts et de grande échelle. Un métamodèle se doit de définir de manière exhaustive l'ensemble des concepts manipulés dans le processus de développement. Le métamodèle CRIO est issu de l'intégration et l'extension de deux métamodèles existants. Le premier RIO a été proposé dans [Hilaire, 2000] et fut conçu pour la modélisation organisationnelle de systèmes multi-agents non hiérarchiques. Le second est le *"framework"* générique pour la modélisation de systèmes multi-agents holoniques proposé dans [Rodriguez, 2005]. RIO tire son nom des trois principaux concepts sur lesquels il repose : Rôle, Interaction, et

Organisation. CRIO, quant à lui précise et redéfinit certains des concepts précédemment introduits dans RIO et leur adjoint celui de Capacité. À cela il intègre ensuite les concepts holoniques et s'intègre dans le processus de développement logiciel ASPECS qui sera décrit au chapitre 4.

CRIO s'inspire de l'approche de développement dirigée par modèles[3] (ou MDD). Ce type de méthode est de plus en plus utilisé dans l'industrie du logiciel. En témoignent d'ailleurs l'adoption par l'*"Object Management Group"* (OMG) en 2003 du standard *"Model Driven Architecture"* [MDA, 2003] et le nombre grandissant d'AGLs basés sur les principes de cette approche, ex : Eclipse, Borland Together Architect, Codagen Architect, etc. L'approche MDD place la notion de modèle au cœur du processus de conception d'un logiciel et ses principes sont les suivants : (i) spécifier le système cible indépendamment d'une plate-forme d'implantation donnée, (ii) spécifier les plates-formes d'implantation, et déterminer une plate-forme particulière pour le système considéré, (iii) et finalement transformer la spécification du système en une spécification compatible avec la plate-forme choisie.

Sur la base de ces principes, MDA considère trois niveaux de modèles :

- Le modèle indépendant de la solution ou CIM[4] se focalise sur l'environnement du système et la spécification des besoins que le système devra satisfaire. Les détails de la structure et du fonctionnement du système sont cachés ou indéterminés. Le CIM est parfois nommé modèle du domaine. Dans ce type de modèle, un vocabulaire spécifique aux acteurs du domaine de l'application est employé dans la spécification. Le CIM a pour objectif de combler le fossé entre les experts du domaine de l'application et leurs besoins d'une part, et les experts de la conception du système en charge de satisfaire ces besoins d'autre part.
- Le modèle indépendant d'une plate-forme d'implantation ou PIM[5] se concentre sur le fonctionnement d'un système tout en dissimulant les détails nécessaires à son implantation sur une plate-forme particulière. Ce type de modèle détaille la partie de la spécification complète qui ne change pas d'une plate-forme d'implantation à l'autre. Un tel modèle peut employer un langage de modélisation générique ou un langage spécifique au domaine dans lequel le système sera utilisé.
- Le modèle spécifique à une plate-forme ou PSM[6] combine le PIM avec un modèle décrivant les détails de l'utilisation d'une plate-forme d'implantation spécifique.

Pour assurer la transition entre ces différents modèles, l'approche MDD définit des transformations entre modèles. Une transformation de modèle consiste à combiner un modèle avec un ensemble d'informations additionnelles, et à appliquer des règles de transformation pour dériver un autre modèle. Le guide MDA de l'OMG [MDA, 2003] décrit ainsi un patron de transformation pour transformer un modèle PIM vers un modèle spécifique à

[3]MDD : *"Model Driven Development"*
[4]CIM :*"Computation Independent Model"*
[5]PIM :*"Platform Independent Model"*
[6]PSM :*"Platform Specific Model"*

une plate-forme donnée (PSM) (voir figure 3.1).

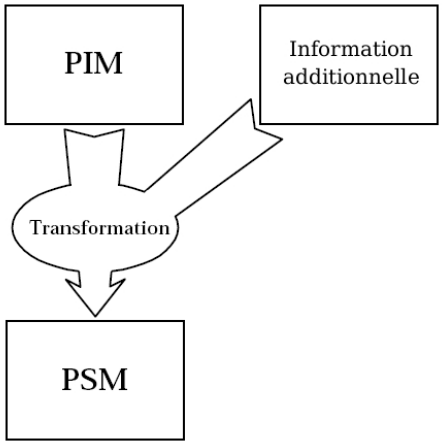


Figure 3.1: Le patron de transformation de PIM vers PSM dans l'approche MDA

Dans la logique de l'approche MDD, CRIO offre trois niveaux de modèles. Chacun de ces trois modèles sera qualifié de domaine[7] :

Le domaine du problème (CIM) qui fournit la description organisationnelle du problème indépendamment d'une solution spécifique. Les concepts introduits dans ce domaine seront principalement utilisés durant la phase d'analyse et au début de la phase de conception du processus de développement.

Le domaine agent (PIM) qui introduit les concepts multi-agents et fournit une description d'une solution multi-agents, éventuellement holonique, basée sur les éléments du *domaine du problème*. Le *domaine agent* est davantage associé à la fin de la phase de conception.

Le domaine de la solution (PSM) est relatif à l'implantation de la solution sur une plate-forme spécifique. Cet aspect est donc dépendant d'une plate-forme de déploiement particulière. Dans le cas présent, cette phase repose sur la plate-forme *Janus* qui fut spécifiquement développée pour faciliter l'implantation de modèles organisationnels et holoniques. Les concepts liés à ce domaine ainsi que la plate-forme *Janus* seront décrits au chapitre 5 de ce document. Ils ne seront donc pas détaillés dans ce chapitre.

Les concepts définis dans le *domaine du problème* et dans le *domaine agent* seront décrits dans la suite de ce chapitre (sections 3.3–3.4). Tout au long de ce chapitre, la notation UML est utilisée pour décrire chacun de ces métamodèles et les exemples qui leur sont associés. Différents *"profiles"* UML sont également introduits afin d'adapter les diverses notations UML aux besoins spécifiques de l'approche proposée.

[7] en référence au métamodèle de la méthodologie PASSI qui constitue également l'une des inspirations pour ce document, davantage de détails sur cet aspect au chapitre 4

3.3 Domaine du problème

La figure 3.2 présente le diagramme UML de la partie du métamodèle CRIO consacrée à la modélisation d'un problème[8]. Le *domaine du problème* introduit les concepts au cœur de l'approche proposée : organisation, rôle, interaction, capacité et ontologie. L'ensemble de ces concepts est détaillé dans la suite de cette section.

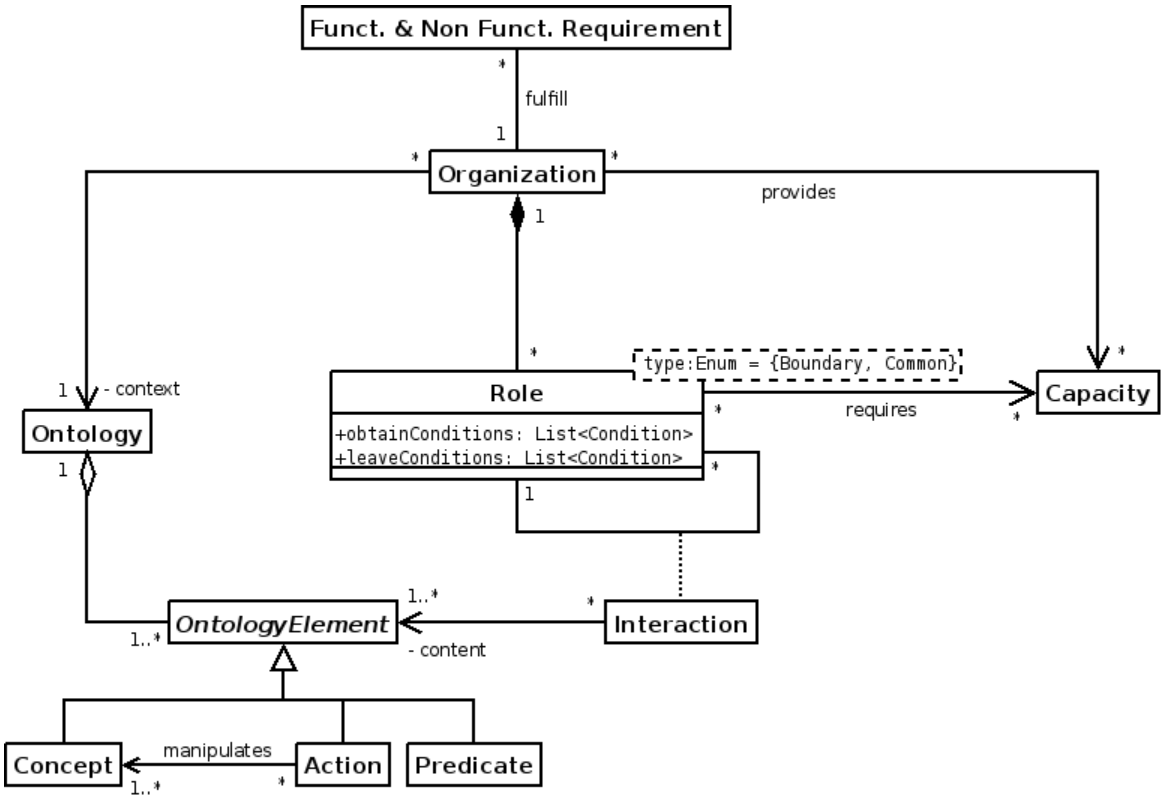


Figure 3.2: Diagramme UML du domaine du problème du métamodèle CRIO

3.3.1 Notion d'ontologie et capitalisation des connaissances du domaine du problème

L'ontologie (*"Ontology"*) et les éléments d'ontologie (*"Ontology Element"*) sont utilisés pour décrire les connaissances du domaine de l'application et définir le contexte des organisations du système. Une ontologie est un ensemble structuré de concepts visant à décrire un ou plusieurs domaines d'étude. Les relations entre concepts peuvent être sémantiques, de composition ou d'héritage.

[Gruber, 1995] fournit une définition générale de la notion d'ontologie : *« Une ontologie est la spécification d'une conceptualisation d'un domaine de connaissance »*. Dans

[8]Nous rappelons qu'une classe liée à une association par des pointillés correspond à une classe d'association (ex : Interaction). Il s'agit d'une classe qui réalise la navigation entre les instances d'autres classes.

cette thèse, une définition plus opérationnelle, proposée par l'OMG dans le document de spécification *"Ontology Definition Metamodel"*[9] (ODM), est adoptée : *« Une ontologie définit un ensemble de termes et de concepts communs utilisés pour décrire et représenter un domaine de connaissance. Une ontologie peut prendre diverses formes telles qu'une taxonomie (ensemble de connaissances avec une hiérarchie minimale), un thésaurus (mots et synonymes), un modèle conceptuel (avec des connaissances plus complexes) ou une théorie logique (avec des connaissances très riches, complexes, consistantes et significatives). Une ontologie bien formée est celle qui est exprimée dans une syntaxe bien définie et qui dispose d'un interpréteur bien précis et conforme avec la définition ci-dessus. »*

Dans le métamodèle CRIO, l'ontologie possède une double fonction. Elle permet tout d'abord de rassembler et d'organiser l'ensemble des connaissances disponibles sur le problème et sur son domaine, et de définir le contexte des organisations utilisées pour les modéliser. La structuration des concepts de l'ontologie du problème permet notamment de renseigner le concepteur sur la structure éventuelle du système. Concernant l'aspect de capitalisation des connaissances du domaine, l'ontologie constitue également une base de connaissances commune à tous les acteurs intervenant dans le développement du système. Elle permet ainsi de regrouper les concepts issus du vocabulaire spécifique des experts du domaine et de les organiser de sorte à faciliter leur compréhension par les équipes en charge de la conception du système.

L'ontologie regroupe également l'ensemble des connaissances qui seront échangées au cours des interactions entre les rôles qui composent les organisations du système. L'ontologie constituera ainsi dans le *domaine agent* la base de connaissances nécessaire à la définition des communications entre les agents.

Une ontologie est composée d'*éléments d'ontologie* (abstraits) disposant de trois types concrets possibles (cf. figure 3.2) :

Concept : une catégorie, une abstraction qui abrège et résume une multiplicité d'objets par généralisation de traits communs identifiables.

Action : mécanisme réalisé par un acteur qui modifie une ou plusieurs propriétés (et par conséquent leurs états) d'un ou plusieurs concepts récepteurs.

Prédicat : assertions sur les propriétés des concepts.

La figure 3.3 présente un exemple d'ontologie issu du domaine de la gestion de projet. Cette dernière détaille le cas d'une société gérant un ensemble de projets. Chaque projet est géré par un *"Manager"* qui est en charge de la décomposition du travail en tâches et de l'assignation de ces tâches aux différents membres de son équipe.

[9] `http://www.omg.org/ontology/`

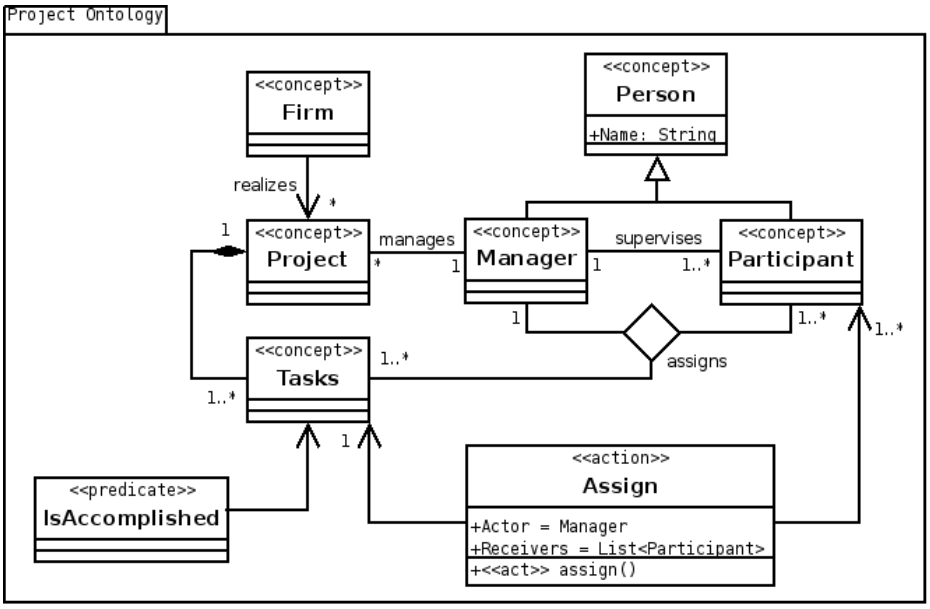


Figure 3.3: Un fragment de l'ontologie du domaine de la gestion de projet

3.3.2 Décomposition comportementale d'un système à l'aide des concepts d'organisation et de rôle

L'organisation est, avec l'interaction, l'un des concepts clefs des systèmes multi-agents. D'ailleurs ces deux concepts sont intrinsèquement liés. L'interaction a lieu au sein d'une organisation, mais réciproquement l'organisation exige l'existence des interactions puisqu'elle-même est définie sur leur base. Selon [Ferber, 1995], les organisations constituent à la fois le support et la manière dont se déroulent les interactions, c'est-à-dire la façon dont sont réparties les tâches, les informations, les ressources, et la coordination des actions. La vision proposée de l'organisation, présentée dans la définition 3.1, est très proche de ce que [Ferber, 1995, chap. 3] qualifie de *structure organisationnelle*.

Définition 3.1 Organisation, inspirée de [Hilaire, 2000, Rodriguez, 2005]

Une organisation est définie par un ensemble de rôles, leurs interactions et un contexte commun définissant ainsi un schéma d'interaction spécifique. Le contexte d'une organisation est défini par tout ou partie de l'ontologie du domaine associée aux besoins qu'elle est en charge de satisfaire.

Chaque organisation est associée à au moins un besoin fonctionnel qui correspond à l'objectif qu'elle doit satisfaire, à la tâche qu'elle doit effectuer, ou au comportement global qu'elle doit exhiber. L'objectif de chaque organisation est donc de satisfaire les différents besoins auxquels elle est associée. Chacun d'entre eux devra être satisfait soit par le comportement individuel de l'un des rôles de l'organisation, soit par le comportement global émergent des interactions de tout ou partie des rôles.

La figure 3.4 décrit l'exemple d'une organisation destinée à la *Gestion de projet*. Elle définit deux rôles *Manager* et *Participant*, et une interaction. Le contexte de cette organisation peut être défini par un sous-ensemble des concepts introduits dans l'ontologie décrite précédemment sur la figure 3.3.

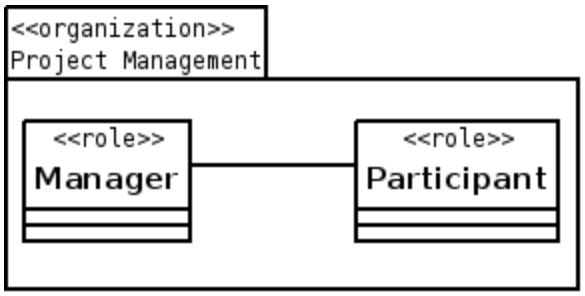


Figure 3.4: Diagramme UML de l'organisation *Gestion de projet*

L'organisation étant à la fois le support et la manière de satisfaire un besoin, elle peut être considérée comme la description d'un comportement global auquel devront se plier une ou plusieurs entités, afin de satisfaire les objectifs qui leurs sont attribués. Si l'on cherche à étudier ce comportement et à le décomposer d'un point de vue fonctionnel, on obtient un ensemble de comportements de complexité inférieure interagissant pour satisfaire les objectifs associés à l'organisation. Selon le niveau d'abstraction considéré, une organisation peut être vue soit comme un comportement unitaire soit comme un ensemble de comportements en interaction. L'organisation est donc un concept intrinsèquement récursif.

Cette dualité est également présente dans le concept de *holon* comme cela sera démontré dans la suite de ce chapitre. Tous deux sont d'ailleurs souvent illustrés par la même analogie : la composition du corps humain. Le corps est considéré d'un certain point de vue comme une entité à part entière disposant d'une identité et d'un comportement propre. Il peut être également considéré comme un agrégat d'organes, eux-mêmes composés de cellules, etc. À chaque niveau de cette hiérarchie de composition, des comportements spécifiques apparaissent [Chauvet, 1998]. Le corps dispose d'une identité et d'un comportement unique pour chaque individu. Les organes disposent chacun d'une mission qui leur est propre : filtration pour les reins, extraction de l'oxygène pour les poumons ou encore circulation du sang pour le cœur.

L'organisation est à la fois l'agrégation d'un ensemble de comportements, et un comportement composant une organisation de niveau supérieur ; le tout constituant une hiérarchie de comportements spécifiques avec à chaque niveau des objectifs précis à satisfaire. Cette définition récursive de l'organisation constituera la base du processus d'analyse associé à ASPECS. Les différents comportements qui composent un système seront récursivement décomposés en un ensemble de sous-comportements en interaction. Cette décomposition se poursuit jusqu'à atteindre un niveau où les comportements obtenus sont considérés comme suffisamment simples pour être gérés par des entités atomiques faciles à implanter. Les

comportements considérés comme élémentaires à un niveau d'abstraction donné sont appelés *rôles*. La définition 3.2 a été retenue pour définir le concept de *rôle*. Cette vision de l'organisation qui est perçue tantôt comme comportement à part entière, tantôt comme un rôle dans une organisation de niveau d'abstraction supérieur est partagée par d'autres auteurs : [Anderson and Reenskaug, 1992] et [Singh, 1992].

La figure 3.5 illustre ce processus de décomposition hiérarchique et organisationnelle d'un système. Après avoir identifié les besoins que l'application devra satisfaire, le système est décomposé hiérarchiquement en accord avec la décomposition effectuée sur les besoins. À un niveau donné, chaque besoin est associé à un comportement global en charge de le satisfaire. Ce comportement est représenté par une organisation et chaque organisation est ensuite décomposée en un ensemble de rôles en interaction. Une fois que l'ensemble des niveaux est clairement spécifié, les contributions entre niveaux adjacents sont ensuite étudiées. Cette étude vise à préciser comment une organisation de niveau n est en mesure de contribuer au comportement d'un rôle de niveau $n+1$. Le processus de décomposition s'arrête lorsque les rôles identifiés à un niveau d'abstraction donné sont considérés comme simples et aisément implantables. Par exemple, dans le cas de la figure 3.5, le rôle 2 de niveau $n+2$ ainsi que les rôles 7 et 8 de niveau $n+1$ sont considérés comme suffisamment simples. Ils ne sont pas décomposés davantage et aucune organisation de niveau inférieur ne leur est donc associée.

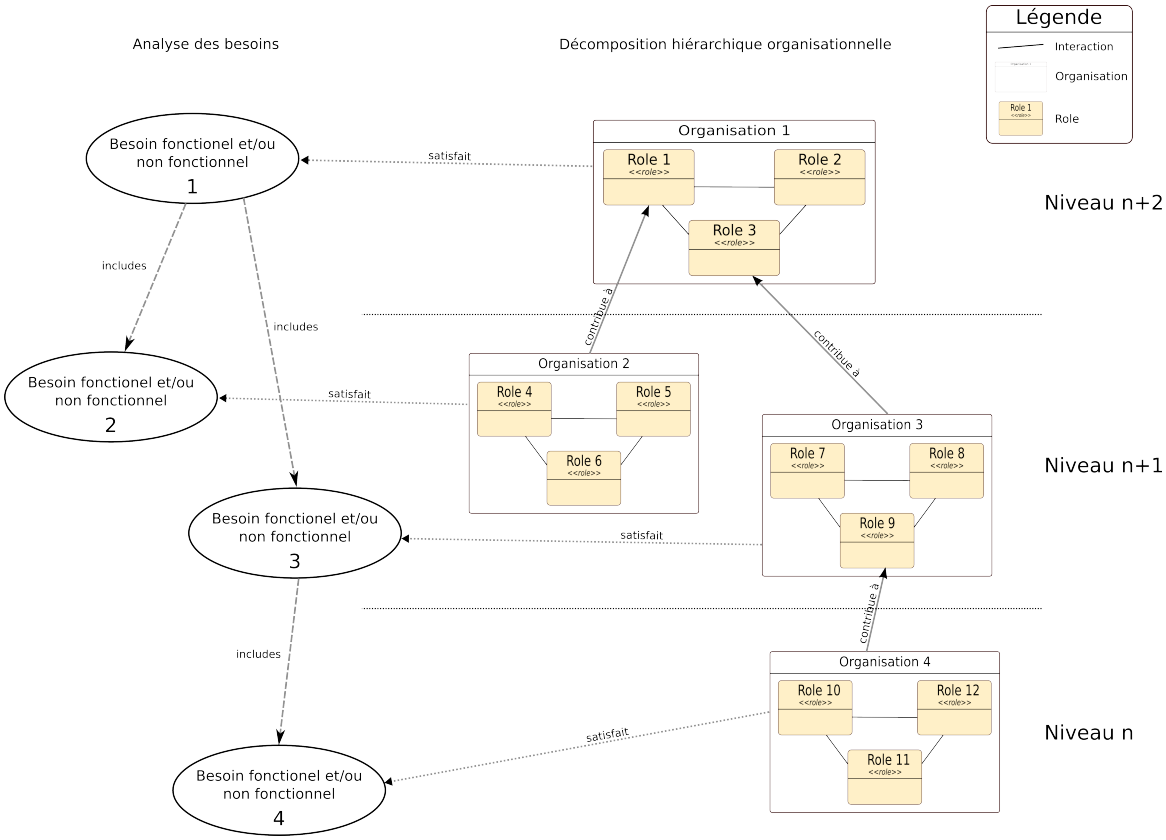


Figure 3.5: Décomposition organisationnelle d'un système en fonction des besoins

Définition 3.2 Rôle, inspirée de [Hilaire, 2000, Rodriguez, 2005]

Un rôle est l'abstraction d'un comportement dans le contexte (espace d'interaction) d'une organisation. Il confère à l'entité qui le joue un statut dans cette organisation. Ce statut définit un ensemble de droits et de devoirs. Un rôle peut interagir avec les autres rôles définis par la même organisation.

Chaque rôle est défini dans une et une seule organisation. Dans le contexte de son organisation, un rôle définit un comportement et un statut, ces deux composantes du rôle sont détaillées ci-après.

Le statut d'un rôle définit la position de ce dernier au sein de son organisation ainsi qu'un ensemble de droits et d'obligations pour l'agent qui le joue. Le statut définit également l'interface au travers de laquelle l'agent jouant le rôle est perçu par les autres entités de la même organisation. Il fournit à cet agent le droit d'exercer ses capacités (compétences) dans le contexte de l'organisation et l'obligation de respecter le comportement décrit par le rôle. Le statut d'un rôle est caractérisé par au moins un concept de l'ontologie définissant le contexte de son organisation.

Le comportement d'un rôle fixe les responsabilités qui sont associées au rôle et la méthode pour les satisfaire. Pour définir ce comportement, chaque rôle dispose d'attributs qui lui sont propres. L'objectif d'un rôle est de contribuer pour toute ou partie aux besoins associés à l'organisation dans laquelle il est défini. Le comportement d'un rôle est spécifié par un plan comportemental (ou plan de comportement) qui peut être représenté par un diagramme d'activité UML ou un diagramme état-transition (*"state-chart"*). Un tel plan décrit comment un ou plusieurs objectifs d'un rôle peuvent être satisfaits par son comportement. Il détaille comment combiner et ordonner les interactions avec les autres rôles, les événements extérieurs au rôle, et avec les tâches qui composent le comportement du rôle.

L'aspect dual du concept de rôle dans CRIO, combinant statut et comportement, est une différence majeure avec bien des définitions existantes de ce concept. En effet, le rôle est souvent considéré comme une interface au travers de laquelle les autres agents perçoivent l'agent qui le joue. Le rôle est alors assimilé à une sorte de filtre pour l'agent, ce qui correspond à la notion de statut dans l'approche proposée. Dans MESSAGE [Caire et al., 2002] par exemple, la relation entre le Rôle et l'Agent est considérée comme analogue à celle entre l'*Interface* et la *Classe* dans les modèles orientés-objet. [Kristensen and Osterbye, 1996], étudiant la notion de rôle pour les objets, nomment cette approche la métaphore du filtre (ou *"Filter Metaphor"*) et examinent les problèmes liés à une telle vision : *"This is mistaken because the filter metaphor implies that the persons*[10] *has all the properties from the outset, and we choose to see only some of them. This neglects the important meaning behind roles, that the properties are extrinsic, — the person only has them because of the role"*. Une

[10]Le terme *"Personne"* est à comprendre ici dans le sens de l'entité qui joue le rôle

étude du concept de rôle pour les objets, qui partage d'ailleurs de nombreux points communs avec l'approche présentée ici consacrée aux rôles d'agents, peut être trouvée dans la thèse de [Graversen, 2006].

Afin de clairement comprendre ces deux aspects de la notion de rôle : statut et comportement, revenons à l'exemple précédent de l'organisation *Gestion de projet* (cf. figure 3.4). Le statut du *Manager* l'autorise à utiliser son autorité pour assigner des tâches aux *Participants* d'un projet. Aucun d'entre eux ne sera surpris que le *Manager* face usage de son autorité, car chacun d'eux perçoit le *Manager* comme son responsable (statut) qui par conséquent dispose de ce droit. Le comportement du rôle *Manager* décrit en revanche la manière d'effectuer cette répartition des tâches parmi les différents membres de l'équipe projet.

Après cette description des aspects internes à un rôle, la suite immédiate est logiquement consacrée à l'étude des relations entre rôles. La base de la coordination entre rôles est incarnée par l'interaction.

De nombreux auteurs s'accordent sur le fait que la complexité des SMA est une conséquence directe de l'interaction entre les agents [Jennings, 2001, Odell, 2002b]. Un système est d'ailleurs généralement considéré comme complexe lorsqu'il ne peut pas être intégralement compris par la seule analyse de ses composants, et que leurs interactions doivent également être prises en compte. La notion d'interaction est fondamentale puisqu'elle permet à un groupe d'agents d'accomplir davantage ensemble que la somme de leurs actions individuelles. Mais l'interaction va également de pair avec la nécessité de coordination et l'émergence de conflits. Une interaction entre rôles est définie de la manière suivante :

Définition 3.3 Interaction, inspirée de [Hilaire, 2000, Rodriguez, 2005]

Une interaction entre rôles est composée de l'événement généré par un premier rôle et perçu par les autres, ainsi que les réactions induites dans ces derniers. Les rôles qui interagissent partagent automatiquement un contexte d'interaction commun.

Les interactions survenant entre les rôles au sein d'une organisation sont décrites dans un scénario d'interaction qui est généralement représenté par un diagramme de séquence UML. Un scénario d'interaction décrit comment un ensemble de rôles interagissent et se coordonnent pour satisfaire un objectif commun, lequel peut lui même être associé à un rôle de niveau d'abstraction supérieur. Cette association implique de pouvoir faire transiter de l'information entre deux niveaux d'abstraction adjacents. Ce mécanisme est géré dans CRIO par l'utilisation combinée des concepts de capacité, d'organisation et service, il sera présenté à la section 3.4.5 de ce chapitre.

Les concepts jusqu'alors introduits dans le *domaine du problème* permettent d'assurer l'identification des besoins, et de dresser une première décomposition du système sous la forme d'une hiérarchie d'organisations ; chacune de ces organisations ayant pour objectif de

satisfaire un ou plusieurs besoins fonctionnels. Cependant, la phase d'analyse d'un système passe, certes, par l'identification des besoins auxquels il devra répondre, mais également par la délimitation du périmètre du système. Ce périmètre marque la frontière entre le système et son environnement, celle-ci étant modélisée par un type spécifique de rôle : le rôle frontière (*"Boundary Role"*), qui est responsable des interactions survenant à la limite entre le système et l'extérieur : interface graphique, base de données, périphérique, etc. Dans CRIO et ASPECS, et contrairement à ce qui est parfois recommandé dans la littérature, aucun modèle global de l'environnement n'est explicitement fourni. En effet, le point de vue adopté sur l'environnement dépend du contexte et de l'organisation considérés. Cette vision est donc distribuée, et le modèle d'environnement l'est également au travers des différentes organisations situées à la frontière du système. Distribuer le point de vue adopté sur l'environnement facilite la distribution des applications dans des environnements hétérogènes et distribués. De plus, l'environnement est par nature à l'extérieur du système, par conséquent seules les parties manipulées par le système ou en interaction avec lui sont effectivement modélisées.

3.3.3 Notion de capacité : comment décrire les compétences d'une organisation ou d'un agent

Les systèmes de grande échelle doivent pouvoir coopérer et fonctionner en environnement ouvert. Les agents impliqués dans de tels systèmes doivent par conséquent collaborer avec d'autres agents, éventuellement intégrés à des systèmes différents, pour satisfaire leurs objectifs. Il en découle que tout agent doit être en mesure d'évaluer les compétences de ses partenaires potentiels et ainsi identifier les collaborateurs les plus appropriés. La notion de capacité fut initialement introduite pour permettre aux agents de raisonner sur leurs propres compétences et celles de leurs collaborateurs de sorte à pouvoir s'adapter et satisfaire des objectifs nouveaux [Rodriguez et al., 2007].

L'objectif de l'introduction du concept de capacité est de disposer d'un outil permettant de décrire les compétences d'un agent ou d'un groupe d'agents de manière générique, tout en faisant abstraction de l'architecture interne de l'agent (voir définition 3.4).

Définition 3.4 Capacité, [Rodriguez et al., 2007]

Une capacité est la description abstraite d'un savoir-faire. Elle regroupe la description de moyens qui permettent l'accomplissement d'une tâche. Ceci incarne une fonctionnalité logique aux yeux des entités qui la fournissent (les propriétaires) et de celles qui l'utilisent ou la requièrent (les utilisateurs).

Les propriétaires de capacités peuvent être des agents, des holons, des rôles ou des organisations. Les utilisateurs sont généralement des rôles ou des organisations.

La capacité possède une double fonction dans le métamodèle CRIO. (i) Elle constitue tout d'abord une interface entre l'agent et le rôle, permettant de définir le comportement du rôle en faisant abstraction de l'architecture de l'agent. Cette notion permet d'obtenir des modèles génériques d'organisation. Une capacité représente en effet une compétence d'un agent ou d'un groupe d'agent. La capacité permet d'effectuer l'interface entre le rôle et l'entité qui le joue. Le rôle requiert certaines compétences pour définir son comportement, lesquelles sont modélisées par des capacités. Les capacités peuvent ensuite être invoquées dans l'une des tâches qui compose le comportement du rôle. En contre-partie, une entité qui souhaite accéder à un rôle, devra fournir une réalisation concrète (ou implantation) à chacune des capacités que le rôle requiert. Ces relations entre capacité, rôle et agent sont décrites dans la figure 3.6. La relation entre capacité et agent sera approfondie dans le *domaine agent*.

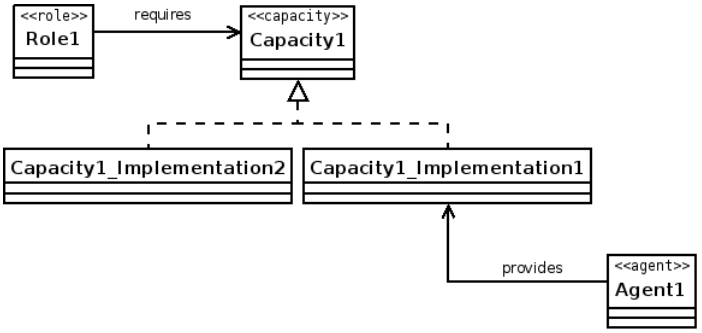


Figure 3.6: Relations entre les concepts de Capacité, de Rôle et d'Agent

(ii) La capacité permet également, dans le processus de modélisation, d'effectuer l'interface entre deux niveaux d'abstraction adjacents dans la hiérarchie organisationnelle du système. La figure 3.8 décrit cette seconde fonction de la notion de capacité.

Afin d'illustrer ce double aspect du concept de capacité, considérons la capacité à trouver le plus court chemin dans un graphe pondéré et connexe $\mathcal{G}(N, E)$ depuis un nœud source s vers un nœud destination d (capacité que l'on nommera *TrouverPlusCourtChemin*). La description associée à une telle capacité est présentée dans la figure 3.7. Cette capacité est paramétrée par le graphe $\mathcal{G}$, défini par son ensemble de nœuds N et d'arêtes E pondérées par la fonction w, ainsi que les nœuds source s et destination d du plus court chemin à identifier. Le résultat de cette capacité est matérialisé par le plus court chemin P liant s à d. Ce chemin représente une séquence de nœuds adjacents du graphe. La clause *requiert* précise que les ensembles de nœuds et d'arêtes ne doivent pas être vides. Elle impose également que les nœuds source et destination appartiennent au graphe et que les poids associés aux arêtes soient positifs ou nuls. La clause *garantit*, précise qu'il n'existe aucun chemin Q du graphe, liant s à d, plus court que P.

La définition d'une *Capacité* ne contient aucune référence aux entités ou groupes d'entités susceptibles de posséder ou d'exhiber ce savoir-faire. Ainsi, la notion de capacité est clairement distincte de la manière dont elle est réalisée. En effet, une capacité décrit ce

Nom : *TrouverPlusCourtChemin - FindShortestPath*

Paramètres :

- $\mathcal{G} = (N, E)$, graphe orienté. $E = N \times N$
- $w : E \rightarrow \mathbb{R}$, fonction de poids.
- $s \in N$, nœud source.
- $d \in N$, nœud destination.

Résultats : $P = \langle s = i_0, i_1, \cdots, i_{n-1}, d = i_n \rangle$, with $\forall\, k \in \{0..n\}, i_k \in N$
Le plus court chemin P entre s et d.

Requiert : $N \neq \varnothing$ et $E \neq \varnothing$ et $\forall (u, v) \in E / w(u, v) \geq 0$

Garanti : $\forall\, j_t \in N,\, t \in \{0..m\}$

$$\nexists\ Q = \langle s = j_0, j_1 \cdots, j_m = d \rangle / P \neq Q \wedge \sum_{t=0}^{m-1} w(j_t, j_{t+1}) < \sum_{k=0}^{n-1} w(i_k, i_{k+1})$$

Il n'existe aucun chemin Q du graphe, liant s à d, plus court que P.

Description Textuelle : Fournit une solution au problème de plus court chemin à un seul nœud source dans un graphe connexe et pondéré dont le poids lié aux arcs est positif ou nul.

Figure 3.7: La capacité *TrouverPlusCourtChemin*

qu'une entité ou un groupe d'entités est capable de faire, indépendamment de sa réalisation effective. Par exemple, la capacité *TrouverPlusCourtChemin* peut être réalisée de diverses manières. Il est possible d'utiliser l'algorithme de *Dijkstra*, ou celui de *Bellman-Ford* si l'on considère le savoir-faire d'une unique entité. D'autres exemples de réalisations peuvent être apportés, notamment si l'on considère la capacité d'un groupe d'entités, plutôt que celle d'une entité individuelle. En effet, la *Colonie de fourmis* est une organisation connue pour être susceptible d'offrir une solution au problème de plus court chemin dans un graphe. La solution (le plus court chemin) émerge des interactions entre fourmis dans leur environnement. En accord avec la description fournie dans la figure 3.7, l'environnement est représenté par le graphe $\mathcal{G}$, le nœud source s est assimilé à la fourmilière, et le nœud destination d à une source de nourriture.

La figure 3.8 présente le diagramme UML des organisations *Colonie de fourmis* et *Détermination de route dans un graphe*, et décrit leurs relations. Ces deux organisations se situent à deux niveaux d'abstraction différents : le niveau des fourmis, et le niveau où le résultat (le plus court chemin) émergeant de leurs interactions est effectivement exploité. Au niveau $n + 1$, le rôle *Route Provider* requiert la capacité à trouver le plus court chemin dans un graphe. Cette capacité est ensuite fournit par une organisation située au niveau inférieur dans la hiérarchie. Au niveau n, le comportement global de l'organisation *Colonie de fourmis* fournit la capacité permettant de déterminer le plus court chemin.

Le concept de capacité permet ainsi de définir comment une organisation de niveau n peut contribuer au comportement d'un rôle de niveau $n + 1$. La manière précise permettant

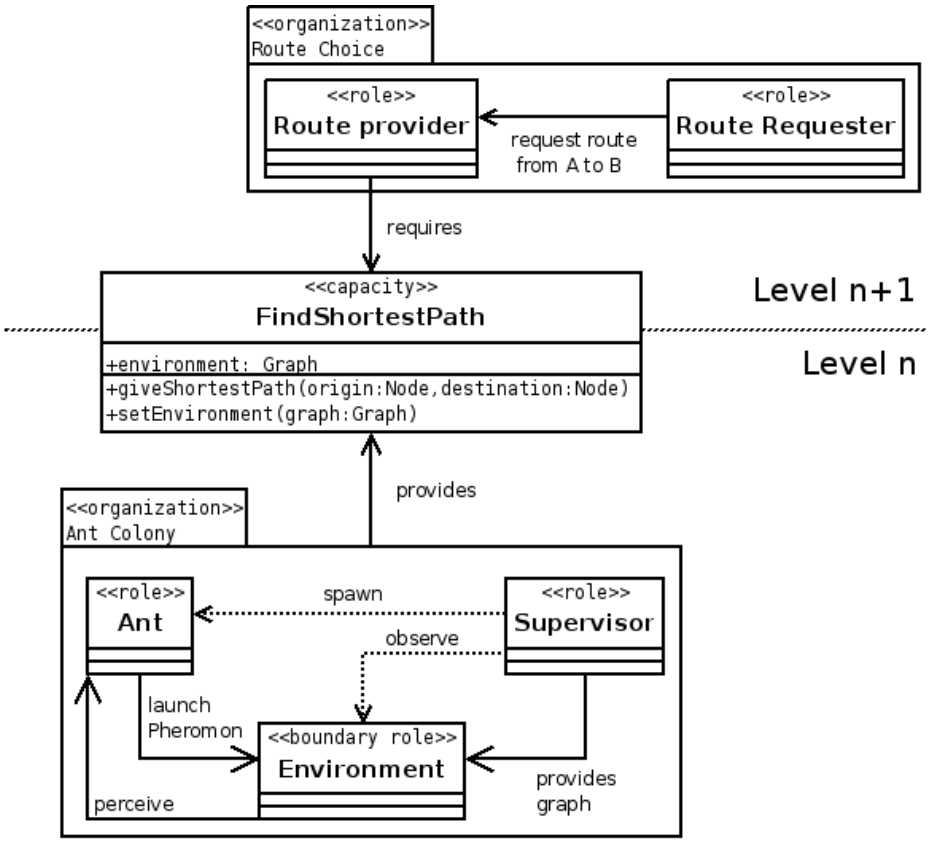


Figure 3.8: Le concept de capacité en tant que charnière entre deux niveaux d'abstraction adjacent

à une organisation de fournir une capacité n'est pas l'objet du *domaine du problème*. Cet aspect sera raffiné et spécialisé dans le *domaine agent* (cf. section 3.4.5) notamment grâce à l'introduction du concept de service.

En revanche, les conditions nécessaires pour qu'une organisation donnée soit effectivement en mesure de fournir une capacité peuvent d'ores et déjà être précisées, et spécifiées directement sur le diagramme UML représentant la hiérarchie d'organisation du système sous la forme de contraintes OCL[11].

3.4 Domaine Agent

Le *domaine du problème* de CRIO permet la modélisation du problème en termes d'organisations, de rôles, de capacités et d'interactions. Le résultat de cette modélisation doit aboutir à la définition d'une hiérarchie d'organisations combinant leurs comportements respectifs pour satisfaire les besoins identifiés. Disposant désormais d'un modèle du problème, l'objectif est alors de concevoir le *modèle d'une solution multi-agents*. Il s'agit d'élaborer le modèle d'une société d'agents (et/ou de holons) capable d'offrir une solution au problème

[11] OCL : *"Object Constraint Language"* [OCL, 2006]

étudié, en décrivant les interactions entre agents ainsi que leurs éventuelles dépendances. La figure 3.9 présente le diagramme UML du *domaine agent* du métamodèle CRIO.

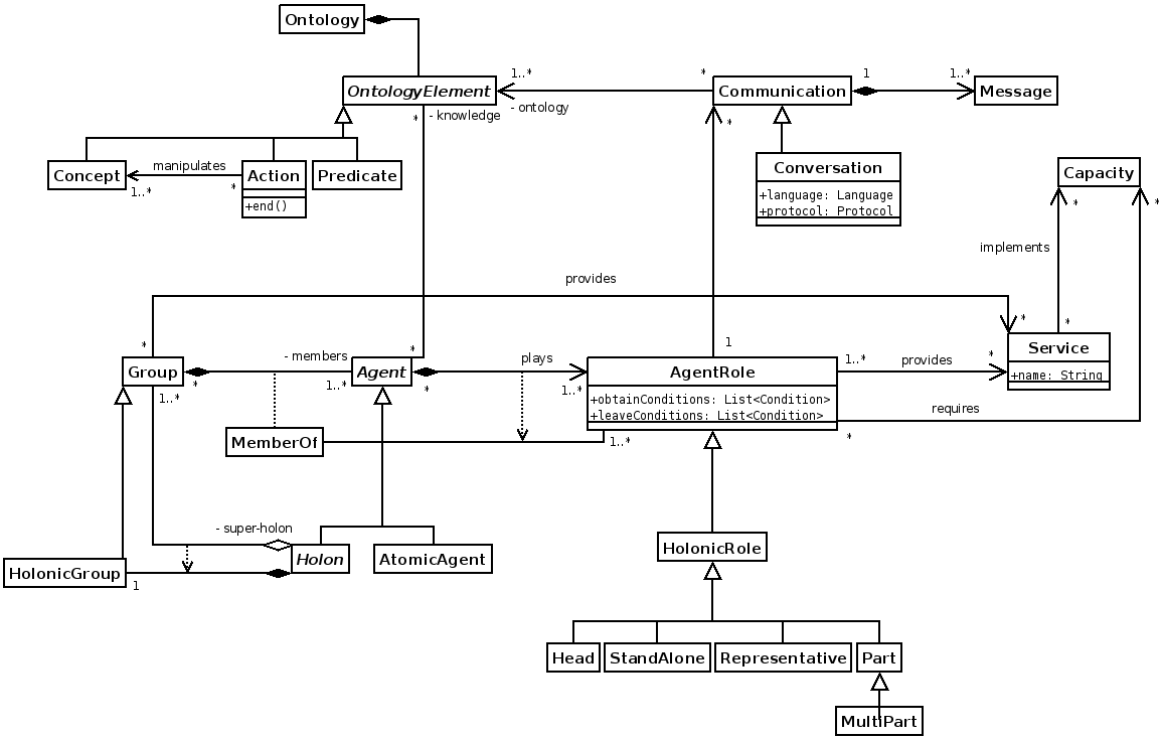


Figure 3.9: Diagramme UML du domaine Agent du métamodèle CRIO

3.4.1 De l'organisation au groupe

Au sein du *domaine agent*, les organisations issues du modèle du problème sont instanciées sous forme de *groupes*. Les rôles qui composaient ces organisations sont eux-aussi instanciés sous forme de *rôles d'agent* (ou *"AgentRole"*). Ces concepts de *groupe* et de *rôle d'agent* sont définis de la manière suivante :

Définition 3.5 Groupe

Un groupe est une instance concrète d'une organisation. Il modélise un groupe d'agent en interaction, coopérant pour satisfaire un ou plusieurs objectifs.

Deux agents ne peuvent communiquer que si ils jouent un rôle dans un groupe commun. Un agent, lorsqu'il joue un rôle dans un groupe donné, se doit de respecter le comportement de ce rôle. Le comportement global du groupe doit suivre le schéma spécifique d'interaction décrit par l'organisation qu'il instancie.

Définition 3.6 Rôle d'agent

Un rôle d'agent est une instance concrète d'un rôle. Il décrit un comportement dans le contexte défini par un groupe. Il confère à l'agent un statut dans ce groupe et les moyens d'interagir avec les autres agents jouant des rôles au sein du même groupe.

Un rôle d'agent est local à un groupe. L'accès à un tel rôle n'est pas automatique : il doit être demandé par l'agent qui désire le jouer. Cet accès est soumis à des conditions que doit valider l'agent pour accéder effectivement au rôle. Les conditions d'obtention d'un rôle sont nommées *obtainConditions*, et l'une d'elles spécifie que l'agent doit disposer de toutes les capacités requises par le rôle. Ces conditions d'obtention constituent un moyen de vérifier que l'agent valide effectivement un certain nombre de pré-requis nécessaires au rôle et notamment qu'il dispose de toutes les compétences requises pour mettre en œuvre le comportement du rôle. Un rôle ne peut pas être libéré directement, car sa libération est également soumise à des conditions. Les conditions de libération, nommées *leaveConditions*, permettent par exemple d'empêcher un agent de quitter un groupe alors qu'il avait la charge d'effectuer une tâche donnée.

Ces différentes conditions permettent de fixer le niveau d'engagement d'un agent dans un groupe. En somme, lorsqu'un agent entre dans un groupe, il accepte non seulement de se plier aux comportements définis dans les rôles qu'il joue, mais également de mettre à disposition tout ou partie de ses compétences. Ces aspects sont directement liés au statut du rôle qui définit les droits dont dispose l'agent, mais également les obligations auxquelles il doit se plier.

Un agent doit jouer au moins un rôle dans un groupe pour pouvoir communiquer, mais il peut également jouer plusieurs rôles au sein d'un même groupe ou plusieurs rôles au sein de groupes différents.

Au cours du processus d'instanciation, une même organisation peut être instanciée plusieurs fois. De manière analogue un rôle peut être instancié sous forme de plusieurs rôles d'agent, plusieurs fois au sein d'un même groupe ou au sein de groupes différents. La relation entre les concepts d'organisation et de groupe, et celle entre rôle et rôle d'agent peuvent être considérées comme analogues à la relation qui lie les concepts de *Classe* et d'*Objet* dans les métamodèles orientés-objet. Les définitions et les relations entre les concepts d'agent, de groupe et de rôle d'agent au sein du *domaine agent* sont très proches de celles fournies dans le métamodèle AGR [Ferber et al., 2004].

Le processus d'instanciation d'organisation est illustré par la figure 3.10. L'organisation de *Gestion de projet*, précédemment décrite par la figure 3.4, est instanciée en deux groupes distincts $g1$ et $g2$, et chacun d'entre eux dispose d'un nombre différent de participants. Deux invariants OCL ont été ajoutés précisant que tout groupe qui instancie l'organisation de *Gestion de projet* contient un unique *Manager* et au moins un *Participant*.

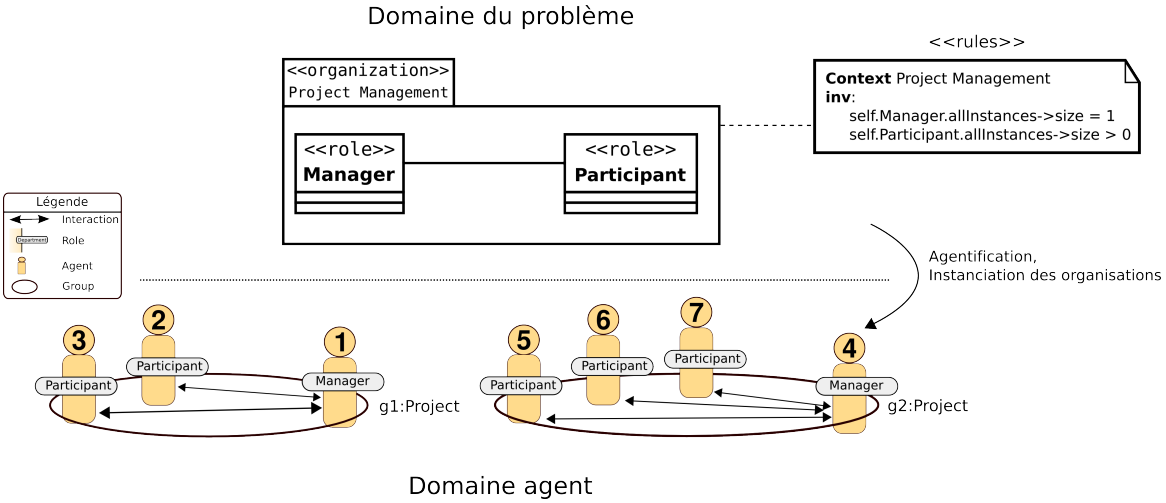


Figure 3.10: Processus d'instanciation d'une organisation en deux groupes g_1 et g_2

3.4.2 De l'interaction à la communication

Le modèle de communication utilisé entre les rôles d'agent présuppose que deux rôles, qui interagissent, partagent au moins le savoir lié au contexte de leur groupe, mais également celui spécifiquement lié à l'information échangée au cours de l'interaction.

Une communication est une spécialisation d'une interaction dans laquelle les connaissances partagées par les participants sont représentées par un ensemble d'éléments de l'ontologie. Une conversation est une spécialisation d'une communication dans laquelle le contenu (langage, ontologie, et encodage) et le contrôle de la communication (protocole) sont explicitement détaillés. Une conversation est principalement composée d'actes de langage [FIPA ACL, 2002, FIPA Com. Act, 2002]. Un protocole définit une séquence de messages attendus et représente un schéma commun de conversation utilisé pour exécuter une tâche, une stratégie de haut niveau gouvernant l'échange d'information entre les agents. Chacun des participants connaissant ce schéma, leur dialogue en est ainsi facilité.

3.4.3 Agent

Le cœur du *domaine agent* tient évidemment dans la définition des entités qui seront à la base de la solution du problème traité : les agents et les holons.

Définition 3.7 Agent

Un agent est une entité jouant un ensemble de rôles qui peuvent être définis dans plusieurs groupes. L'agent est caractérisé par ses rôles, ses connaissances et ses capacités.

Derrière cette définition minimale, on retrouve un concept relativement proche de la définition fournie par [Ferber, 1995] (cf. section 2.2.2, chapitre 2) ; cependant, une partie

de ce qui était attribué directement à l'agent est désormais externalisée dans ses rôles ou ses capacités.

En effet, chaque agent dispose de la capacité de base qui lui permet de jouer des rôles et donc de communiquer avec d'autres agents. Chaque agent possède ensuite des capacités ou compétences spécifiques, qu'il peut mettre au service d'autres agents, soit en partageant un groupe commun, soit par la publication d'un service (cf. section 3.4.5).

Le comportement global d'un agent résulte de la combinaison des rôles, connaissances et capacités dont il dispose à un instant donné. Chaque agent tend à satisfaire les objectifs qui ont été attribués aux rôles qu'il joue, et qui sont spécifiés dans leur comportement. Mais le comportement d'un rôle peut indirectement être influencé par les caractéristiques propres de l'agent telles ses accointances ou ses tendances, si l'implantation d'une des capacités qu'il requiert, est basée sur ces éléments. L'accès aux caractéristiques propres de l'agent ne s'effectue généralement que par l'intermédiaire d'une implantation particulière d'une capacité requise par le rôle. Cette approche garantit un certain niveau de généricité du rôle en faisant abstraction de l'architecture interne de l'agent.

Un aspect important à retenir dans la définition 3.7 est qu'elle autorise, voire favorise, une dynamique importante des rôles au sein de l'agent, et du système dans sa globalité. Un agent jouera simultanément et/ou successivement au cours de son exécution un grand nombre de rôles. Si l'on considère l'exemple d'un maître de conférence dans une université, ce dernier exerce généralement au moins deux rôles. En effet, en qualité d'enseignant, il est habilité à donner des cours à des étudiants, mais il pratique généralement en parallèle une activité de recherche au sein d'un laboratoire spécialisé. La figure 3.11 illustre l'exemple d'un agent Enseignant-Chercheur qui joue simultanément plusieurs rôles dans des groupes différents. L'agent A_1 joue ainsi le rôle d'enseignant dans le groupe $g1 : Lecture$. Parallèlement, il participe à un projet de recherche au sein du groupe $g2 : Research\ Project$, tout en prenant également part aux décisions du conseil du département Génie informatique de son université, modélisé dans l'exemple par le groupe $g2 : Council$. Cette dernière dénomination signifie que le groupe $g2$ est une instance de l'organisation *Council*. Quelques-unes des organisations impliquées dans la modélisation d'une université sont présentées dans la partie haute de la figure.

3.4.4 Holon

En sus de la notion d'agent, CRIO introduit un second type d'entité incarné par la notion de holon. Cette section est dédiée à la présentation des multiples facettes de ce concept et à la manière de le représenter en adoptant une approche organisationnelle.

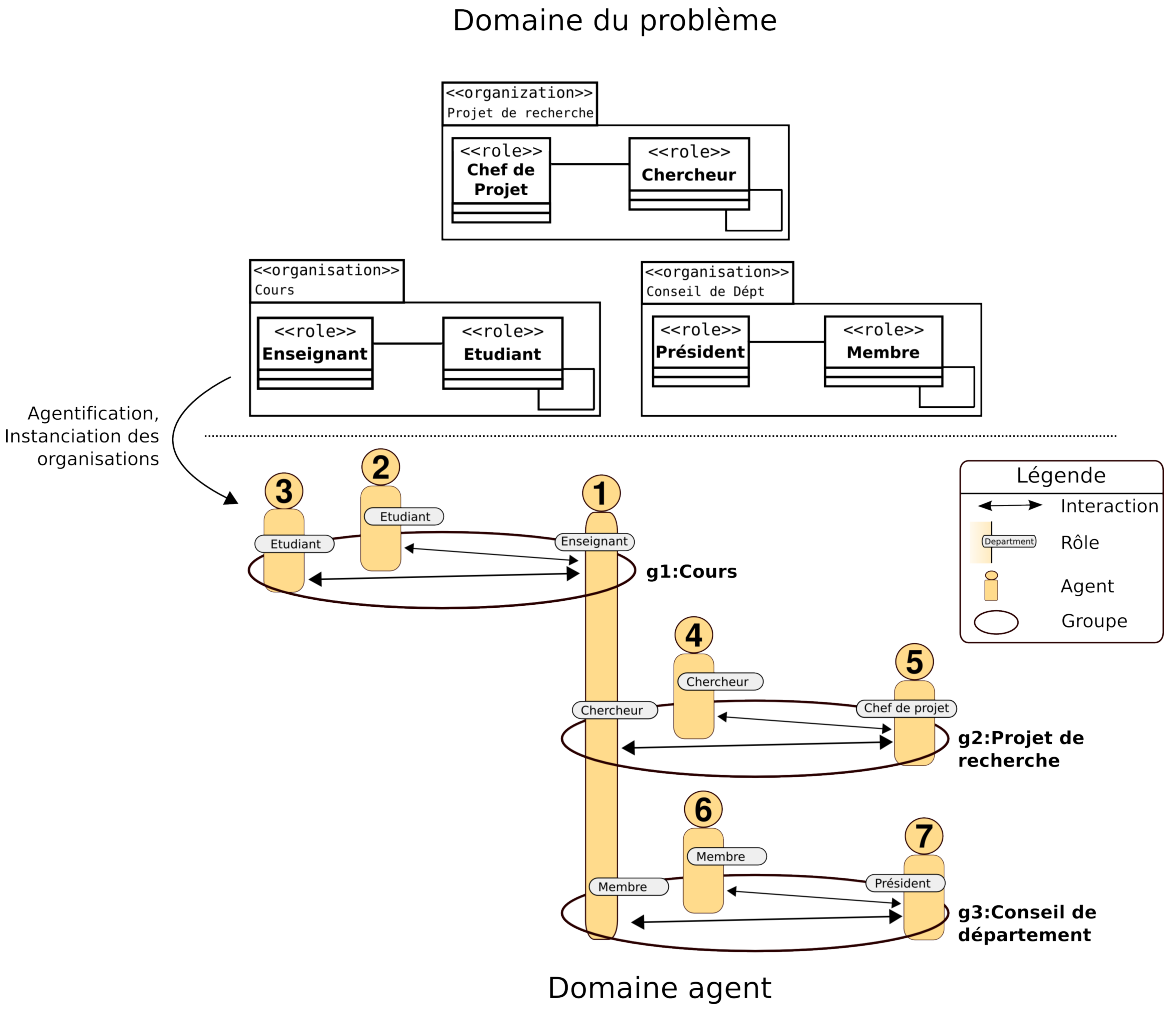


Figure 3.11: Exemple d'un agent jouant simultanément deux rôles dans trois groupes distincts

3.4.4.1 Introduction et terminologie

Un holon est une entité auto-similaire composée de holons comme sous-structures. La structure hiérarchique composée de holons est appelée *holarchie*. Un holon peut être vu, en fonction du niveau d'observation, tantôt comme une entité atomique, tantôt comme un groupe de holons en interaction. De la même manière, un ensemble constitué de différents holons peut être considéré comme un ensemble d'entités en interaction ou comme des parties d'un holon de niveau supérieur. Cette dualité est parfois appelée l'*effet Janus*[12], en référence à ces deux faces du holon. La figure 3.12 décrit cet aspect dual de la notion de holon. À un niveau d'observation donné, le holon composé est qualifié de super-holon. Les holons qui composent un super-holon sont appelés sous-holons ou holons membres.

De nombreux exemples de holarchies peuvent être trouvés dans la vie quotidienne : le

[12]Janus est une divinité romaine dieu des portes, gardien des portes de l'enfer. Il est représenté avec deux visages, l'un tourné vers le passé et l'autre tourné vers le futur. Pour plus de détails, se référer à : `http://fr.wikipedia.org/wiki/Janus_(mythologie)`

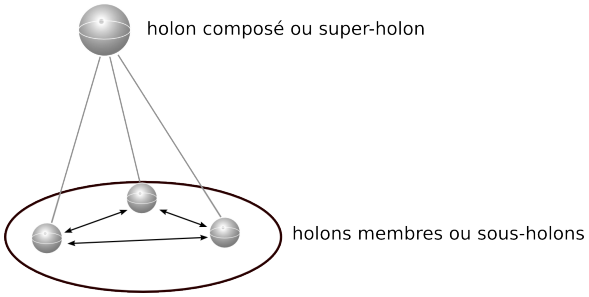


Figure 3.12: Un holon composé de trois holons membres

corps humain, les villes, les entreprises ou encore les galaxies. Si l'on considère une université d'un point de vue holonique, on peut alors la décomposer en différents départements et laboratoires. De la même manière, un département peut être décomposé en un ensemble d'organisations (conseil de département, cours, etc) peuplées d'enseignants et d'étudiants. Cette holarchie de l'université est décrite dans la figure 3.13. Au niveau le plus haut (niveau $n + 2$) se trouve le holon université, qui dans notre exemple est composé de trois sous-holons : le département informatique, le département mécanique et un laboratoire (niveau $n + 1$). Au niveau le plus bas se trouvent les enseignants, les étudiants et les chercheurs (niveau n).

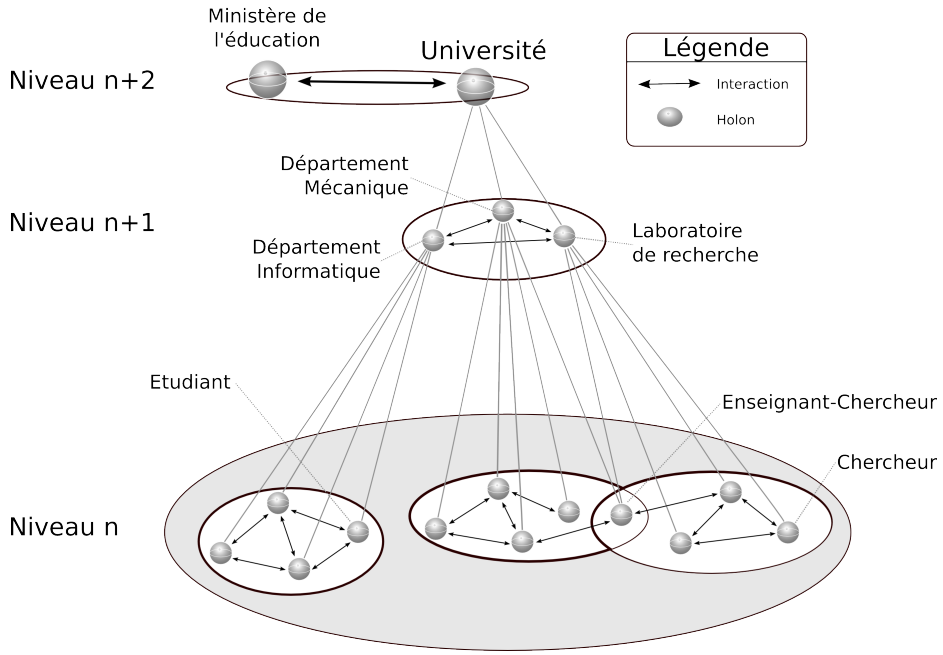


Figure 3.13: Exemple de la structure holonique d'une université

Un holon est à la fois un tout composé d'autres holons, et une partie composante d'un (ou plusieurs) holons de plus haut niveau. En cela, la notion de holon peut être rapprochée de celle d'organisation en tant que comportement à part entière ou composante d'un comportement de plus haut niveau. Ce parallèle entre ces deux concepts facilitera d'autant le processus d'agentification du modèle issu de la modélisation du problème.

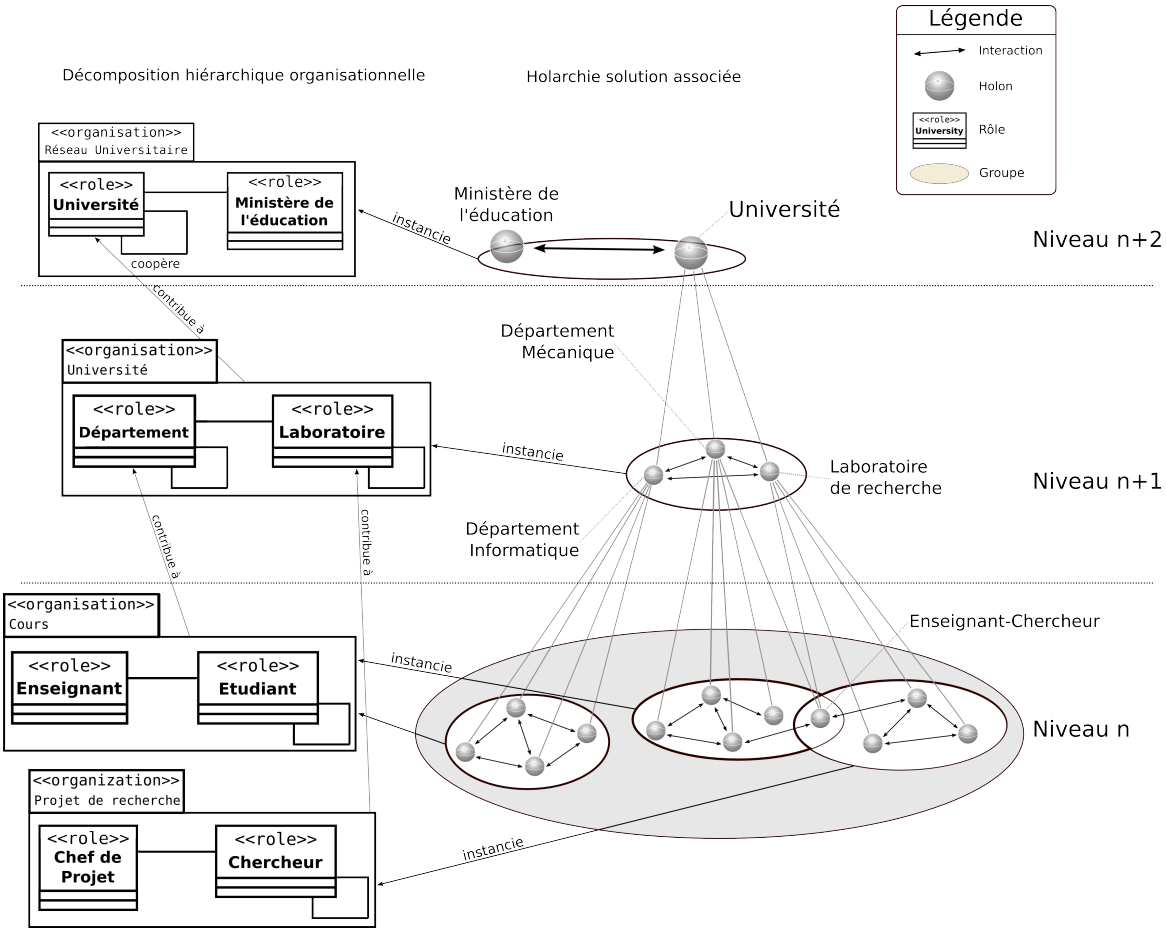


Figure 3.14: Lien entre décomposition hiérarchique organisationnelle et holarchie d'exécution

La figure 3.14 illustre cette association entre la décomposition hiérarchique organisationnelle d'un problème et la création d'une holarchie solution associée à ce problème. En respectant l'approche décrite dans le domaine du problème, le système est décomposé en différents niveaux d'abstraction. Chaque niveau est composé d'un ensemble d'organisations, lesquelles, situées à un niveau d'abstraction donné (niveau n), peuvent contribuer aux comportements de rôles définis à un niveau supérieur (niveau $n + 1$). Pour construire la holarchie, les organisations sont instanciées sous forme de groupes. Un ensemble de holons est ensuite créé à chaque niveau, chacun d'eux jouant un ou plusieurs rôles dans un ou plusieurs groupes du niveau considéré. Les relations de composition entre super-holons et sous-holons sont ensuite définies en accord avec les contributions entre organisations dans la hiérarchie organisationnelle. Par exemple les super-holons de niveau $n + 1$ jouent des rôles dans les groupes de niveau $n + 1$. Les membres respectifs de ces super-holons jouent des rôles dans les différents groupes de niveau n, qui contribuent au comportement des rôles de niveau $n + 1$ joués par leur super-holon. Les aspects individuel et collectif d'un holon sont ainsi pleinement exploités.

3.4.4.2 Définition

À un niveau d'observation donné, un super-holon peut être considéré comme un ensemble de sous-holons en interaction. Ces interactions définissent la structure et le comportement global du super-holon. Afin de modéliser de manière ordonnée et modulaire les interactions et les comportements des membres d'un super-holon, ces derniers sont regroupés par contexte commun via un ensemble de groupes. Un holon peut alors être défini de la manière suivante :

Définition 3.8 Holon

Un Holon $\mathcal{H}_n$ de niveau n peut être défini par le quadruplet suivant :

$$\mathcal{H}_n =< R_n, H_{n-1}, OP, \psi >$$

avec :

R_n **:** l'ensemble des rôles joués par $\mathcal{H}_n$, $R_n = 2^{roles(O_n)}$ avec O_n l'ensemble des groupes où $\mathcal{H}_n$ joue au moins un rôle.

H_{n-1} **:** l'ensemble des sous-holons membres du super-holon $\mathcal{H}_n$.

OP **:** l'ensemble des groupes qui participe à la vie et au fonctionnement du super-holon $\mathcal{H}_n$ et qui contribue notamment à la satisfaction des objectifs liés aux rôles de R_n.

$\psi : H_{n-1} \rightarrow 2^{roles(OP)}$ **:** fonction associant un sous-holon membre à l'ensemble des rôles qu'il joue dans les groupes définis au sein de $\mathcal{H}_n$, tel que $\forall\, h_i \in H_{n-1}, \psi(h_i) \neq \varnothing$ et $card(\psi(h_i)) \geq 2$. La fonction *roles* fournit l'ensemble des rôles définis dans les groupes de *OP*.

Cette définition implique que des holons ne peuvent générer un super-holon que s'ils interagissent. Un super-holon n'est pas uniquement défini par ses membres, mais également par leur manière d'interagir, leur schéma d'interaction. Il en découle que deux super-holons peuvent être créés depuis le même ensemble de sous-holons si leurs manières d'interagir et d'influencer les comportements des super-holons diffèrent.

Un point important à souligner est que, lorsqu'un holon devient membre d'un super-holon, il s'engage auprès de ce dernier et également auprès des autres membres. Cet engagement se traduit par la mise à disposition de tout ou partie des capacités que possède un holon, qu'elles soient requises ou non dans l'immédiat par les rôles qu'il joue au sein de son super-holon. À la création du super-holon le niveau d'engagement de chaque membre doit être spécifié. À l'intégration d'un nouveau membre, le niveau d'engagement de ce dernier est également précisé. Quand un holon prend un rôle au sein d'un super-holon, il accepte d'honorer le comportement associé à ce rôle. Les engagements de chaque membre caractérisent ainsi la structure du super-holon et garantissent la stabilité de ce dernier.

3.4.4.3 Structure d'un super-holon

CRIO est basée sur une approche organisationnelle de sorte à minimiser l'impact de l'architecture sous-jacente des entités impliquées dans la solution développée. Or, pour conserver cette généricité, deux aspects se superposant dans la notion de holon doivent être clairement distingués :

Administration — Gouvernement et structure : ce premier aspect est directement lié à la nature holonique de l'entité : une entité composée d'entités. Il décrit les processus généraux de prise de décision et de gestion au sein d'un super-holon. Il précise notamment la répartition des pouvoirs entre les différents membres. Cet aspect, commun à tous les holons composés, est qualifié d'*aspect holonique*. Il est modélisé par une organisation spécifique : l'*organisation holonique*, qui sera détaillée dans la section suivante (cf. section 3.4.4.4).

Production — Les interactions dépendantes des objectifs du problème : ce second aspect est dépendant de l'application. Il concerne les mécanismes de coordination et d'interaction entre les membres visant à satisfaire les objectifs du super-holon, les tâches à effectuer et éventuellement la prise de décision pour un objectif particulier. Cet aspect est relatif aux organisations utilisées pour modéliser un système donné, il est par conséquent nommé *aspect production*. Pour clairement distinguer les organisations liées à cet aspect de l'organisation holonique, celles-ci sont qualifiées d'*organisations de production*.

Chacun de ces aspects est traité et modélisé de manière séparée. Le lecteur pourra trouver davantage de détails sur l'approche utilisée pour modéliser les systèmes holoniques dans la thèse de [Rodriguez, 2005]. Mais avant d'étudier chacun de ces aspects, la précédente définition 3.8 de la notion de holon doit être raffinée pour distinguer les deux aspects.

Un holon est une spécialisation d'agent. Il s'agit donc d'une entité jouant des rôles qui peuvent être définis dans plusieurs groupes. Un holon peut être composé d'un ensemble de groupes qui précisent comment ses membres sont organisés et interagissent en son sein pour satisfaire les objectifs qui lui sont assignés. Un holon non composé est considéré comme un agent atomique. Un holon composé contient :

- un et un seul *groupe holonique*, instance de l'*organisation holonique* qui précise comment les membres sont organisés et gèrent le super-holon.
- un ou plusieurs *groupes de production*, instances d'organisations qui décrivent comment les membres interagissent et se coordonnent pour satisfaire les objectifs et les tâches assignés à leur super-holon. La définition de ces organisations est dépendante du problème traité.

Un holon composé peut contenir plusieurs instances de la même organisation de production. La table 3.1 détaille les objectifs associés aux aspects holonique et de production d'un holon et les principales stratégies possibles et disponibles dans CRIO pour les satisfaire.

	Objectifs	**Stratégies**
Gouvernement Administration	– Définir les objectifs et attribuer les tâches – Contrôler le processus d'auto-organisation – Superviser le processus de prise de décision – Filtrer et/ou traduire l'information – Recruter ou bannir des membres	– Un rôle – Une élection d'un agent ou d'un groupe d'agents – Un comité
Production	– Accomplir des tâches – Produire des services	– une capacité – un service – orchestrer des capacités ou des services élémentaires

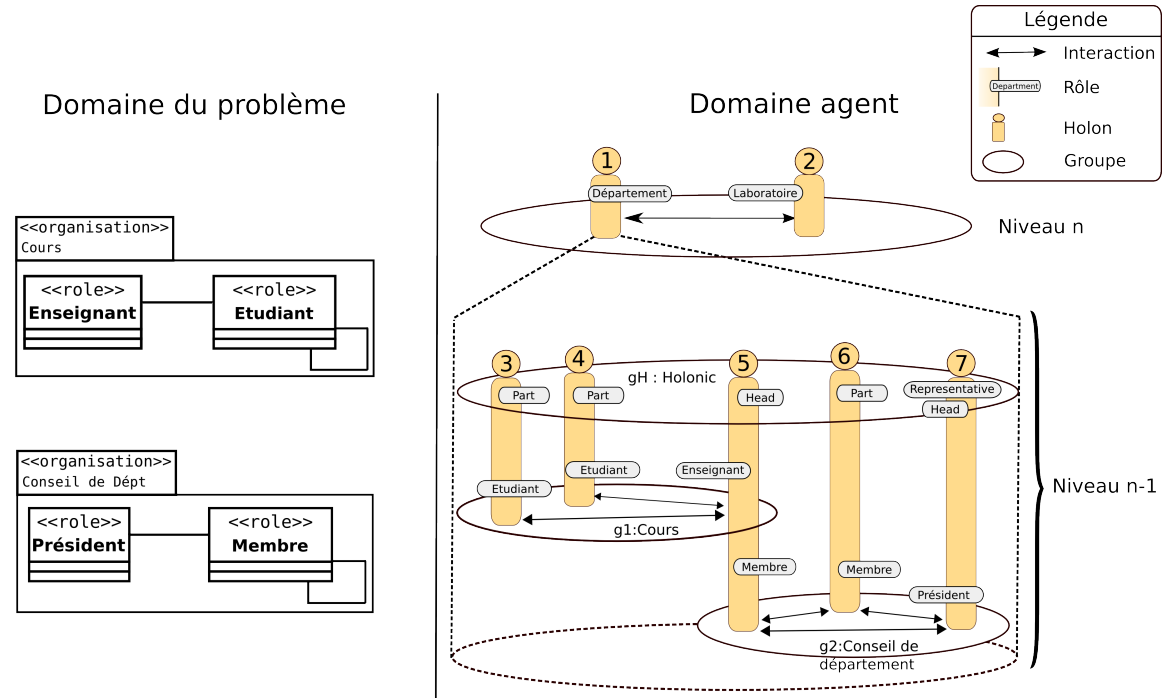
Table 3.1: Description des objectifs et stratégies associés aux deux aspects se superposant au sein d'un super-holon

La figure 3.15 illustre cette distinction entre les aspects holonique et production au sein d'un super-holon. La holarchie présentée détaille la modélisation holonique (à l'aide de CRIO) de l'exemple précédent de l'université. Le holon H_1 joue le rôle *département* au niveau n où il interagit avec les autres instances de l'université telles qu'un laboratoire de recherche par exemple. Mais le département est également composé du conseil en charge de le diriger et des différents cours qu'il propose. Le holon H_1 est ainsi composé de deux groupes de production $g1$ et $g2$, et du groupe holonique gH au niveau $n-1$. Le groupe de production $g1$ peut par exemple représenter un cours de mathématique mêlant des étudiants et un enseignant (holon H_5), le groupe $g2$ correspond au conseil du département. Le holon H_5 joue ainsi le rôle d'enseignant dans $g1$ et le rôle de membre du conseil dans $g2$.

3.4.4.4 Modélisation de l'aspect holonique d'un super-holon

Un super-holon est une communauté de sous-holons qui coopèrent pour satisfaire des objectifs communément acceptés. Chaque membre de la communauté s'engage auprès des autres. Au sein d'une communauté, de nombreuses structures peuvent être utilisées pour clarifier les responsabilités et les engagements de chacun. Cette section décrit une manière de définir le statut de chaque membre ainsi que les mécanismes de gestion et de prise de décision au sein d'un super-holon. Quelques structures définisssant la répartition des pouvoirs au sein d'un super-holon et considérées comme caractéristiques sont également présentées.

L'organisation *holonique* a été initialement conçue pour décrire le mode de gestion et la structure d'un super-holon en termes de répartition d'autorité et de pouvoirs. Elle est inspirée de la notion de *Groupe Modéré* [Gerber et al., 1999] et fut choisie pour sa souplesse et le large panel de configurations qu'elle offre, en modifiant simplement le degré



(a) Organisations de production (b) Groupe Holonique et de Production

Figure 3.15: Un exemple de la structure de la holarchie Université

d'engagement des holons membres envers leur(s) super-holon(s). Dans un groupe modéré, un sous-ensemble des membres représente les autres à l'extérieur du groupe. Pour représenter un groupe modéré avec une approche organisationnelle, cinq rôles, qualifiés de rôles *holoniques*, ont été identifiés :

Head : représente un statut privilégié auquel les membres peuvent conférer un certain degré d'autorité et certains droits particuliers. Il prend part au processus de prise de décision et, en contrepartie, il assure une partie de la charge d'administration du super-holon. Cette charge dépend de la configuration choisie et peut varier au cours de la vie du super-holon.

Representative : il fait partie de l'interface visible du super-holon. Il joue le rôle d'interface entre l'intérieur et l'extérieur du super-holon. Il assure la redistribution et l'éventuelle traduction de l'information arrivant de l'extérieur. Il représente les autres membres à l'extérieur du super-holon. Plusieurs *Representatives* peuvent être en charge de représenter les autres membres.

Part : identifie les membres d'un unique super-holon. Ils sont normalement en charge de l'exécution des tâches qui leur sont affectées par les *Heads*. Ils peuvent éventuellement prendre part au processus de décision. Cette participation à la prise de décision dépend de la structure de gouvernement choisie pour le super-holon.

MultiPart : extension du rôle *Part*, il identifie les membres partagés entre plusieurs super-holons. Le fait qu'un membre puisse être partagé entre plusieurs super-holons peut

éventuellement générer des conflits d'intérêt ou d'autorité. Si au cours de la vie du super-holon, un de ses membres *Part* rejoint un autre super-holon, il devra changer de rôle pour devenir *Multipart*.

Stand-Alone : Ce rôle représente le statut attribué aux holons non-membres. Tous les holons non-membres sont extérieurs au super-holon, et sont perçus par les membres à travers le statut *Stand-Alone*. Ce rôle a été ajouté pour gérer le recrutement de nouveaux membres au sein d'un super-holon.

Les quatre premiers rôles holoniques décrivent le statut d'un membre au sein d'un super-holon et participent à la définition de l'organisation *holonique*. Chacun de ces rôles peut être joué par un ou plusieurs membres, sachant que tout super-holon doit disposer d'au moins un *Representative* et un *Head*. Les rôles *Head*, *Part* et *Multi-Part* sont exclusifs entre eux, alors que *Representative* peut être joué simultanément avec l'un des trois autres.

Si l'on ne considère que les membres d'un super-holon, l'ensemble de ces membres forme une communauté qui est représentée au niveau supérieur par son super-holon. Les interactions, au sein de cette communauté, relatives à l'aspect holonique sont décrites par l'organisation holonique. Une instance de cette organisation est nommée *groupe holonique*. Chaque membre d'un super-holon doit jouer au moins un rôle dans ce groupe, et tout holon qui rejoint le super-holon après sa création, devra également jouer un rôle dans ce groupe.

La figure 3.16 détaille l'exemple d'une holarchie d'entreprise. Le super-holon H_1 est en charge du comportement d'une société qui gère un ensemble de projets. Chaque projet est lui-même associé à un super-holon composé des différents membres de l'équipe projet. Le super-holon H_2 est ainsi composé d'un groupe de production appelé *g1 :Project* et d'un groupe holonique. Ce groupe *g1* est une instance de l'organisation *Project* précédemment décrite dans la figure 3.4 (page 74). Chaque holon qui appartient à ce groupe joue l'un des rôles définis dans cette organisation et chaque équipe projet est dirigée par un manager (holon H_6 et H_7) qui prend les décisions et représente les membres de l'équipe. Le holon H_4 dispose d'un statut particulier puisqu'il est partagé entre les super-holons H_2 et H_3, il joue donc le rôle *Multi-part* dans les groupes holoniques de ces deux super-holons.

L'organisation holonique définit de manière générale la répartition des pouvoirs au sein d'un super-holon, et précise comment les décisions sont prises. Il est également possible de surcharger ce comportement global en créant une organisation de production spécifique pour préciser comment seront prises les décisions pour certains objectifs ou certaines tâches particulières. L'organisation holonique offre cependant un large panel de configurations possibles pour définir le gouvernement d'un super-holon. Cette section ne se veut pas une présentation exhaustive des différents types de processus de prise de décision, mais illustre simplement la grande palette de configurations autorisées.

En considérant seulement le mode de répartition des rôles holoniques au sein des membres d'un super-holon, plusieurs types de gouvernement peuvent être distingués. Quatre types, considérés comme caractéristiques, sont détaillés ci-dessous et dans la table

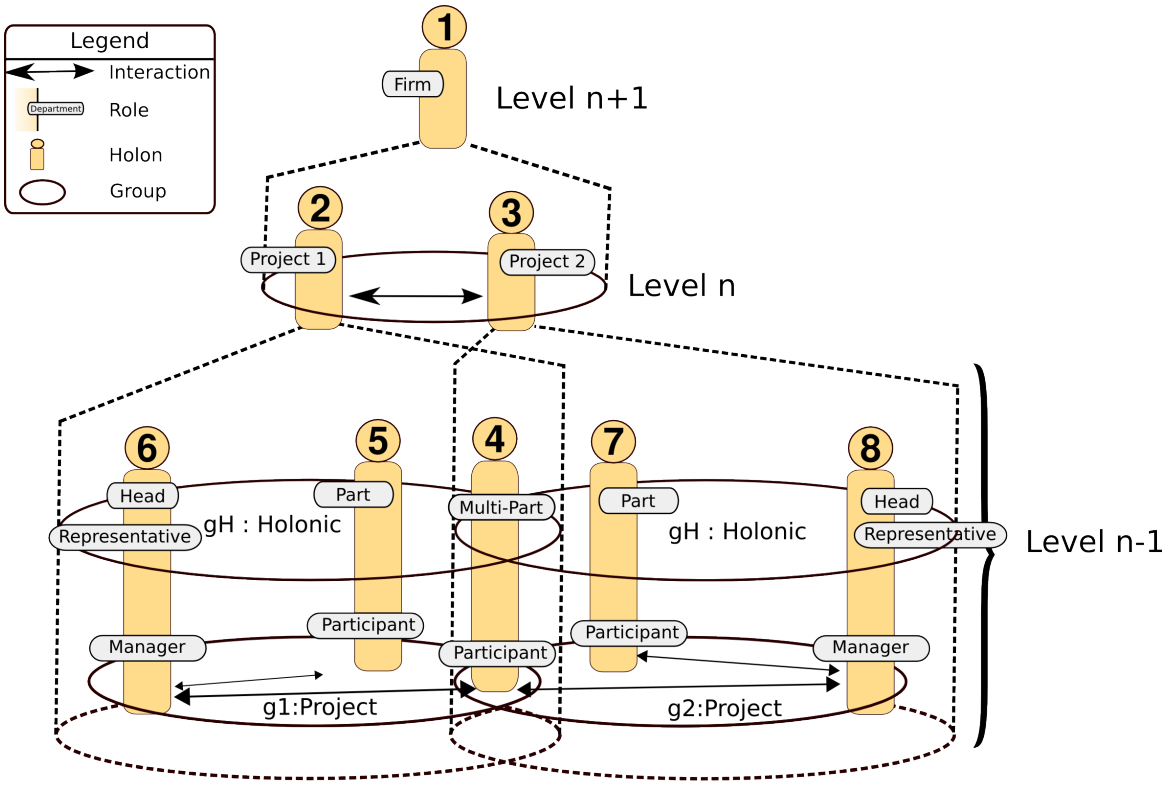


Figure 3.16: Un exemple de la structure de la holarchie Entreprise

3.2 :

Monarchie : Le commandement est centralisé au niveau d'un *Head* unique. La notion de monarchie ne réfère pas ici au mode de nomination du *Head*, mais uniquement à la répartition des pouvoirs au sein de la communauté et au fait qu'un unique *Head* contrôle l'intégralité du processus de prise de décision. Le mode de nomination est une partie intégrante de la dynamique de création d'un super-holon et se doit d'être spécifié séparément.

Oligarchie : Un petit groupe de *Heads* partage le commandement sans en référer aux autres membres de statut *Part*.

Polyarchie [13] : Un petit groupe de *Heads* partage le commandement, mais ils peuvent se référer aux *Parts* pour certaines décisions (via un vote par exemple).

Apanarchie : Le commandement est entièrement partagé entre tous les membres du super-holon. Toute la communauté est impliquée dans le processus de prise de décision.

[13] Nous empruntons ici le terme inventé par Robert A. Dahl pour décrire un type spécifique de gouvernement démocratique.

Nom	Configuration
Monarchy	un Head, un Votant, pas de Part votant
Oligarchy	n Heads, n Votants, pas de Part votant
Polyarchy	n Heads, $n + k$ Votants, k Part votant
Apanarchy	Tout le monde est Head, tout le monde vote

Table 3.2: Les formes caractéristiques de gouvernement d'un super-holon [Rodriguez, 2005]

3.4.4.5 Création d'un super-holon et Intégration de nouveaux membres

Cette section décrit un sous-ensemble des mécanismes qui gouvernent la dynamique d'un super-holon. Seuls les aspects relatifs à la création d'un super-holon et au recrutement d'un nouveau membre sont abordés. La thèse de [Rodriguez, 2005] fournit davantage de détails sur la dynamique et sur les processus d'auto-organisation des systèmes multi-agents holoniques.

Deux approches sont généralement distinguées pour créer un nouveau super-holon :

Montante (bottom-up) : la fusion (*"merging"*), un ensemble de holons s'associent pour créer un nouveau super-holon en charge de satisfaire un objectif donné.

Descendante (top-down) : la sub-division (*"splitting"*), un holon dont les tâches deviennent trop complexes décide de créer un ensemble de groupes de production pour exécuter ses tâches et distribuer les coûts de calcul. Pour peupler ces différents groupes, le super-holon pourra ensuite recruter des membres disposant des capacités requises ou créer de nouveaux holons.

Les objectifs et les tâches d'un super-holon peuvent évoluer au cours de sa vie. Il peut par conséquent avoir besoin de recruter de nouveaux membres pour satisfaire aux nouveaux objectifs.

À un niveau d'abstraction donné, le holon est considéré comme un ensemble de groupes et donc de membres en interaction. Cependant, depuis l'extérieur du super-holon, ces membres sont invisibles par le reste du système. Aucun holon non membre ne peut directement interagir avec les membres, excepté avec le ou les représentants de ceux-ci, les *Representatives*. Un holon peut demeurer seul, et dans ce cas, ces décisions ne sont pas restreintes et ne dépendent que de ses objectifs propres. Un holon demeure généralement dans cet état tant qu'il est satisfait. En accord avec ses besoins, un holon peut décider de rejoindre un super-holon existant afin de satisfaire des objectifs communs. Pour ce faire, il doit demander son admission à l'un des représentants des membres, lequel sert d'interface entre la communauté du super-holon et le candidat à l'intégration. Les *Heads* du super-holon décident ensuite, en fonction des capacités du candidat. Si son admission est acceptée, le holon devient un membre à part entière ; dès lors, et jusqu'à la destruction du super-holon ou le départ du nouveau membre, ce dernier peut interagir directement avec les

autres membres. Ce nouveau membre intègre le groupe holonique du super-holon ainsi que les groupes de production qui participent à satisfaire les objectifs qu'ils ont en commun.

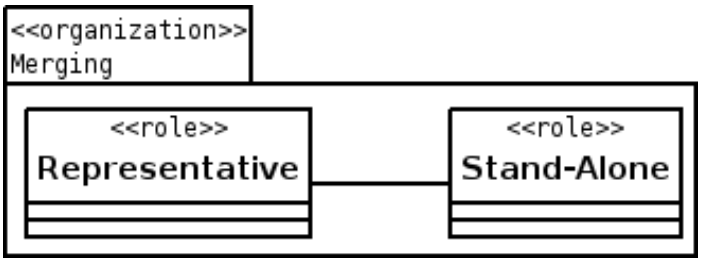


Figure 3.17: L'organisation *Merging* pour gérer l'intégration de nouveaux membres au sein d'un super-holon

Pour gérer ce processus d'intégration de nouveaux membres, une organisation spécifique, nommée *Merging*, a été créée. Elle est décrite dans la figure 3.17. Cette organisation définit deux rôles : *Representative* joué par l'un des représentants des membres du super-holon, et *Stand-Alone* joué par le candidat au recrutement. Lorsqu'un candidat demande son intégration à un super-holon existant, cette organisation est instanciée sous forme de groupe. Le candidat obtient le rôle *Stand-Alone* et l'un des représentants du super-holon intègre le groupe, lequel constitue le support nécessaire aux négociations de recrutement. La figure 3.18 présente un exemple d'instanciation de l'organisation *Merging* au sein d'un super-holon.

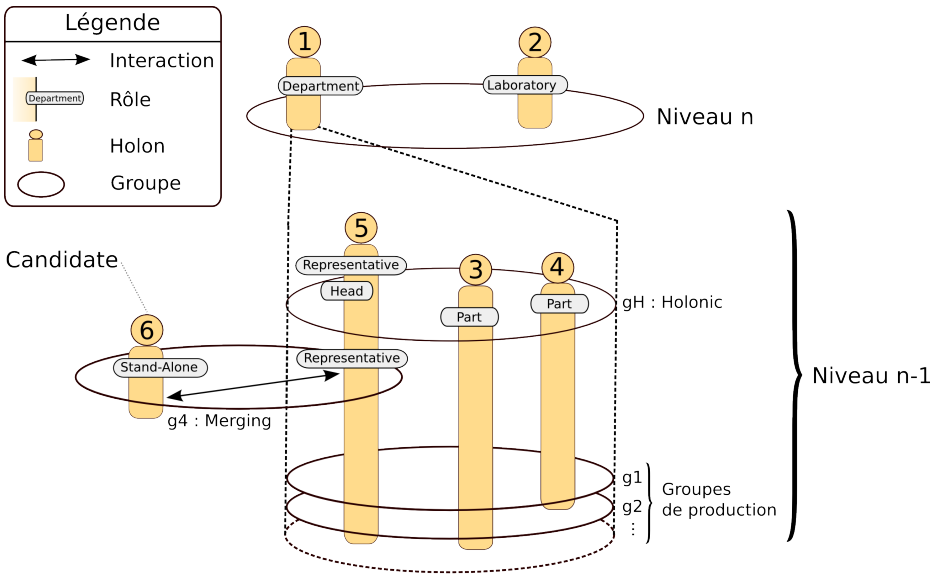


Figure 3.18: Le groupe *Merging* pour gérer l'intégration de nouveaux membres au sein d'un super-holon

Un holon peut changer ses rôles en accord avec ses besoins et ses objectifs, et tenter de fusionner avec des holons ou d'intégrer des super-holons existants.

Il est alors possible de déterminer si deux holons sont ou non prédisposés à collaborer en accord avec leurs objectifs respectifs et les capacités dont ils disposent. Un holon peut ainsi calculer l'affinité qu'il a avec d'autres holons en comparant leurs objectifs. Cette affinité peut ensuite être utilisée pour décider de fusionner ou non au sein d'un super-holon. La mesure de l'affinité entre holons constitue la base du processus de création des super-holons et des holarchies.

La notion d'affinité est dépendante des objectifs à satisfaire et donc de l'application étudiée. Diverses définitions ou formulations peuvent être fournies selon la nature du problème. Dans un sens général, l'affinité mesure, en accord avec les objectifs de l'application, la propension de deux holons à collaborer pour satisfaire un objectif commun. Cette définition d'affinité est ensuite affinée dans le contexte de chaque problème. Un exemple concret, détaillant la manière de calculer l'affinité entre deux holons, sera présenté au chapitre 7 dans le cadre d'une simulation de piétons en environnement urbain virtuel (cf. section 7.2, page 186).

3.4.5 Relations entre les concepts de capacité et de service

Il était nécessaire, pour aborder sereinement la partie qui suit, de comprendre les fondements de la vision proposée pour la notion de holon ainsi que la manière adoptée pour structurer les membres d'un holon composé. Il est maintenant possible de s'intéresser aux relations qui existent entre les concepts de service et de capacité pour montrer comment un super-holon peut pleinement exploiter les compétences de ses membres pour accéder à des rôles qui leur sont inaccessibles. La section présente se focalise en effet sur l'exploitation des compétences émergentes d'un groupe de holons en interaction. La clef d'un tel mécanisme repose sur le fait qu'un groupe d'agent est en mesure de fournir un service. Dans l'approche proposée, la notion de service se définit de la manière suivante :

Définition 3.9 Service

Un service est une fonctionnalité qui représente la possibilité d'exécuter des tâches et ce au nom de son propriétaire : agent, holon, rôle ou groupe. Cette fonctionnalité logique est généralement associée à une ou plusieurs capacités qu'elle est capable de réaliser.

D'après la définition précédente, un agent ou un groupe est en mesure de fournir un service qui peut réaliser une capacité. Si on prend le cas d'un rôle, cette définition prend un sens tout particulier. En effet, un rôle requiert des capacités pour définir son comportement, mais il peut également les mettre à la disposition d'autres rôles en publiant un service. Les différents rôles d'un groupe peuvent alors exploiter leurs capacités respectives par échanges de services. Un groupe est également capable de fournir une capacité *« collective »*, qu'il

peut partager avec les autres groupes en publiant un service à son tour. Ces capacités de groupe résultent de la collaboration entre les différents membres du groupe.

Cette section montre comment exploiter dans le cadre des systèmes multi-agents holoniques, ces compétences qui *émergent* des interactions entre les différentes parties d'un système. En effet, le groupe est considéré comme capables de fournir un service. Or, un service peut être vu comme une manière possible de réaliser une capacité. Dès lors, un super-holon peut posséder une capacité, qui peut être réalisée par un service, fourni par tout ou partie de ses membres. Ce super-holon est alors en mesure d'accéder à des rôles inaccessibles à ses membres.

Si l'on considère à nouveau l'exemple de la capacité *TrouverPlusCourtChemin*, présentée au début de ce chapitre, l'organisation *Colonie de fourmis* est capable de fournir la capacité *TrouverPlusCourtChemin* (voir figure 3.8, page 81). Cette capacité est ensuite requise pour définir le comportement du rôle *Route Provider* de l'organisation *Détermination de route dans un graphe*. Ces deux organisations se situent à deux niveaux d'abstraction différents.

Comme l'illustre la figure 3.19, l'approche holonique permet de représenter simplement ces deux niveaux en créant un super-holon dont les membres seront les fourmis. Les deux organisations précédentes sont instanciées sous forme de groupes $g0$ et $g1$. Le groupe $g0$ est composé de deux membres H_1 et H_2 de niveaux n. Le holon H_1 joue le rôle *Route Requester* ; il désire obtenir la route la plus courte pour atteindre sa destination, et pour cela, il interroge le holon H_2 jouant le rôle *Route Provider*. Le comportement de ce rôle est basé sur la capacité *TrouverPlusCourtChemin*. Le holon H_2 doit donc posséder une réalisation de cette capacité. Supposons pour l'exemple qu'il opte pour la réalisation basée sur l'organisation *Colonie de fourmis*. Dans ce cas, le holon H_2 contient une instance de cette organisation, notée *g1 : Ant Colony*. Dès lors, les membres de H_1 intégrés à ce groupe jouent l'un des rôles définis dans le diagramme UML de l'organisation (cf. figure 3.8).

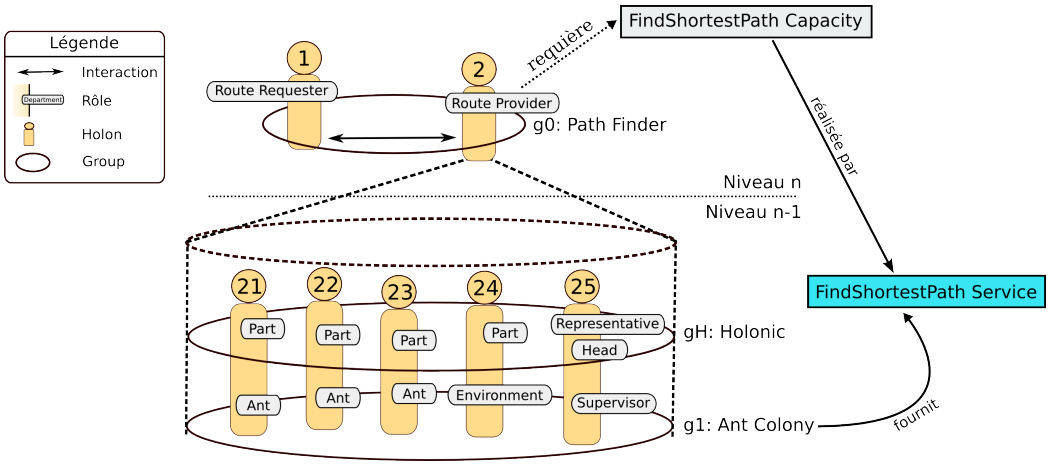


Figure 3.19: Structure d'un super-holon exploitant la capacité collective d'une colonie de fourmis

L'exemple précédent illustre le fait qu'un groupe peut réaliser une capacité via un service. Ce type de réalisation implique que deux niveaux d'abstraction doivent cohabiter au sein d'un même système. L'approche holonique permet de le modéliser de manière aisée.

En effet, le super-holon (ici H_2) peut exploiter les comportements émergents des interactions entre ses membres et ainsi jouer des rôles inaccessibles à ses membres. Un super-holon peut donc réaliser une capacité, soit en obtenant directement une réalisation sous forme d'algorithmes, soit en intégrant un groupe (de production) capable de fournir un service qui réalise la capacité dont il a besoin.

Il demeure encore un aspect à détailler lorsque l'on considère une réalisation collective d'une capacité : comment rediriger les stimulii externes du super-holon vers ses membres. Le rôle holonique *Representative* constitue la solution à ce problème. En effet, les membres jouant ce rôle font partie de l'interface du super-holon. Ils sont en charge de la redistribution et de l'éventuelle traduction de l'information arrivant de l'extérieur du holon vers les autres membres. Ce mode de transfert de l'information requiert parfois l'adaptation des organisations avant leur intégration au modèle holonique. C'est pourquoi dans l'organisation *Colonie de fourmis*, un rôle nouveau est introduit : le *Supervisor*. Ce rôle impose que l'agent qui le joue, assume également le rôle *Representative*. L'information peut alors transiter entre ces rôles[14]. Le rôle *Supervisor* observe l'environnement jusqu'à ce que le plus court chemin soit disponible et le transmet alors au super-holon. Le plus court chemin émergent des interactions entre les fourmis, la présence d'un observateur est donc impérative pour déterminer la disponibilité de ce résultat.

Une organisation peut fournir un service de différentes manières. On peut distinguer trois approches possibles :

Atomique : le service est fourni par l'un des rôles de l'organisation. Le rôle possède une capacité disposant d'une description compatible avec celle du service.

Composé : le service est obtenu depuis les interactions d'un sous-ensemble des rôles de l'organisation en suivant un protocole connu. Ceci est un scénario de composition de service où le plan d'obtention est a priori connu et où les performances peuvent être connues et assurées.

Émergent : le service est obtenu grâce aux interactions d'un sous-ensemble de rôles de l'organisation, mais le plan d'obtention n'est pas connu a priori. Le résultat du service est émergent.

Dans ce dernier cas, la gestion du service fourni par le comportement global de l'organisation est généralement assurée par l'un des rôles de l'organisation, comme c'est le cas dans la colonie de fourmis avec le rôle *Supervisor*.

La section suivante s'intéresse au processus d'acquisition dynamique de capacité pour un holon. Elle s'attachera notamment à démonter l'intérêt dans ce cadre des relations entre

[14] La transition de l'information entre des rôles définis dans des organisations différentes, mais joués par un même agent sera explicité au chapitre 5.

les concepts d'organisation et de capacité.

3.4.6 Acquisition dynamique de capacité

L'un des aspects fondamentaux de la notion de capacité est qu'elle permet à un agent de pouvoir raisonner sur ses propres compétences et sur celles des autres agents. Un agent possède à l'origine un ensemble de capacités basiques qu'il peut faire évoluer dynamiquement.

Pour acquérir une nouvelle capacité, un holon peut intégrer un nouveau groupe fournissant un service capable de la réaliser. La procédure d'acquisition de nouvelles capacités par l'intégration de nouveaux groupes peut être résumée par les étapes suivantes :

1. Le holon compare les capacités fournies par les différentes organisations connues dans le système avec celles requises par le rôle qu'il désire jouer. Pour effectuer cet appariement (*"matchmaking process"*) dans un système ouvert, un langage commun de description des capacités et des services est requis. [Sycara et al., 1999a,b] proposent un modèle permettant d'effectuer dynamiquement cette opération qui peut aisément être adapté au cas présent.
2. Si des organisations candidates ont été identifiées, le holon doit choisir celle qui lui correspond le mieux. Ce choix est en fait essentiellement guidé par la comparaison des capacités requises par les rôles de l'organisation et des capacités possédées par les membres actuels du holon.
3. Quand une organisation a été choisie, elle est instanciée sous forme de groupe interne au holon. Les rôles qu'elle contient sont instanciés et distribués aux membres du holon. Si l'un des rôles ne peut être joué (par manque de membres, ou de capacités requises), une procédure de recrutement de nouveaux membres est lancée (ou éventuellement une procédure d'acquisition de capacité pour l'un des membres).
4. Lorsque chacun des rôles, définis dans le groupe nouvellement créé, est associé à un agent (en accord avec le nombre d'instances requis pour chaque rôle), le superholon est alors en mesure d'obtenir la nouvelle capacité et ainsi acquérir le rôle qu'il convoitait.

La figure 3.20 illustre ce scénario d'acquisition sur l'exemple de la capacité *TrouverPlusCourtChemin*. Le groupe g_0, instance de l'organisation *Détermination de route dans un graphe*, contient déjà deux holons H_1 et H_2 ; le premier jouant le rôle *Route Requester* et le second *Route Provider*. Le holon H_2 possède la réalisation de la capacité *TrouverPlusCourtChemin* basée sur l'algorithme de *Dijkstra*. Le holon H_3 souhaite intégrer le groupe et jouer le rôle *Route Provider* ; or il ne possède pas la capacité (cf. figure 3.20, étape 1). Il dispose alors de trois possibilités, soit acquérir une réalisation de type *Dijkstra*, soit recruter un nouveau membre qui possède déjà cette capacité, ou enfin intégrer une organisation capable de réaliser la capacité requise. Il est supposé ici que l'agent opte pour cette dernière

alternative. L'organisation *Colonie de fourmis* est identifiée comme candidate via le processus d'appariement (cf. figure 3.20, étape 2). Elle est instanciée sous forme de groupe et celui-ci est intégré au holon H_3 (cf. figure 3.20, étape 3). Disposant finalement de la capacité requise, H_3 peut alors intégrer le groupe g_0 et jouer le rôle désiré (cf. figure 3.20, étape 4).

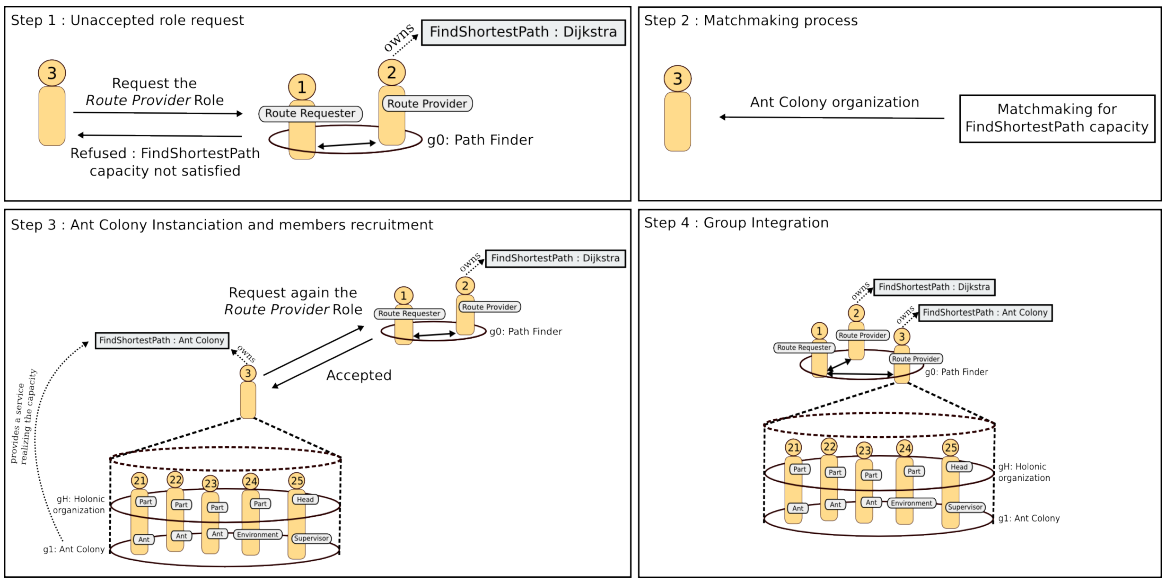


Figure 3.20: Processus d'acquisition d'une nouvelle capacité par l'intégration d'une nouvelle organisation interne

3.5 Conclusions et perspectives

Dans ce chapitre, le métamodèle CRIO a été introduit pour faciliter l'analyse et la conception de systèmes complexes. Celui-ci est basé sur l'approche de développement dirigée par modèles (MDD) et fournit en fait trois métamodèles ; chacun étant associé à l'un des niveaux de modélisation de l'approche MDD : CIM, PIM et PSM. Seuls les deux premiers niveaux ont été détaillés dans ce chapitre ; le dernier lié à la plate-forme *Janus*, sera décrit au chapitre 5.

Le *domaine du problème* est en charge de la description d'un système en termes d'organisation, de rôle, d'interaction, de capacité et d'ontologie. Il a la charge de collecter les connaissances liées au système, de délimiter son périmètre et d'identifier les besoins à satisfaire. Il fournit également l'ensemble des moyens nécessaires pour construire une hiérarchie d'organisations capable de satisfaire ces besoins.

Le *domaine agent* décrit le modèle d'une société d'agents, éventuellement holonique, en charge d'offrir une solution au problème précédemment modélisé. Il définit les concepts d'agent, de holon, de groupe et de service. Il fournit en somme l'ensemble des moyens nécessaires pour donner vie à la hiérarchie organisationnelle du système.

CRIO permet ainsi la modélisation d'un système avec un nombre quelconque de niveaux d'abstraction grâce au mécanisme de décomposition organisationnelle hiérarchique et à la définition intrinsèquement récursive du concept de holon. L'approche organisationnelle garantit que le métamodèle pourra être aisément étendu par l'introduction de n'importe quel mécanisme pouvant être modélisé sous la forme d'organisation.

Les mécanismes de communications inter- et intra- niveaux d'abstraction sont également fournis : (i) intra-niveau lorsque des holons de même niveau interagissent, ces interactions sont décrites dans le cadre des organisations autorisant ainsi n'importe quel type de communication ; (ii) inter-niveau via la possibilité de réaliser une capacité requise par un rôle de niveau n, par un service fourni par une organisation de niveau $n-1$. La notion de *Representative* permet également de modéliser cette communication inter-niveau puisque le *Representative* est considéré comme l'interface d'un super-holon et de ce fait capable de transmettre, traduire, et rediriger l'information aux *Heads* pour générer la réaction appropriée.

CRIO encourage la réutilisation de solutions et de modèles en permettant la définition d'organisations génériques sur la base du concept de capacité.

Grâce à l'ontologie, considérée comme la base de connaissances communes transversale au processus de modélisation, l'ensemble des connaissances du *domaine du problème* sont regroupées. Ceci permet de faciliter leur partage, leur réutilisation, et leur extension de manière modulaire.

CRIO offre également des mécanismes de raisonnement dynamique et d'extension sur les capacités d'un agent (acquisition dynamique de nouvelles capacités). Il propose également une manière d'exploiter les capacités *« collectives »* qui émergent d'un groupe d'entités en considérant un groupe comme capable de fournir un service ; service qui peut lui-même réaliser une capacité requise par un rôle de niveau d'abstraction supérieur. Ce mécanisme permet à un super-holon de jouer des rôles inaccessibles à ses membres.

CRIO adopte une approche de modélisation distribuée de l'environnement d'un système. En effet, l'environnement est modélisé selon le point de vue des organisations qui le manipulent. La notion de rôle frontière (*boundary role*) permet ainsi de définir les limites du système à développer et de modéliser les interactions avec cet environnement.

Le métamodèle proposé dans ce chapitre tente globalement de satisfaire les différents principes de conception exposés au chapitre précédent. Le chapitre suivant est consacré à la description du processus de développement logiciel ASPECS qui permet d'exploiter les abstractions fournies par CRIO pour produire des solutions à des problèmes complexes.

« Une circonférence est le lieu géométrique de tous les points équidistants d'un point donné. Pour construire une circonférence, tournez un compas, une de ses pointes restant fixe, jusqu'à ce que l'autre soit revenue à son point de départ. Il est implicite, d'après Euclide, que si vous mettez en œuvre le processus défini par la deuxième phrase, vous produirez un objet qui satisfera à la définition de la première. La première phrase est une description d'état de la circonférence, la seconde une description processus.

Ces deux modalités d'appréhension des structures constituent à la fois la chaîne et la trame de notre expérience. Les tableaux, les schémas, bien des diagrammes, les formules des structures chimiques sont des descriptions d'états. Les recettes, les équations différentielles, les équations des réactions chimiques sont des descriptions de processus. Les premières caractérisent le monde tel que nous le percevons ; elles nous donnent un critère pour identifier les objets souvent en modélisant les objets eux-mêmes. Les secondes caractérisent le monde dans lequel nous agissons. Elles nous donnent les moyens pour produire ou pour engendrer des objets ayant les caractéristiques désirées. »

H.A. Simon

La science des systèmes, science de l'artificiel,

1969, trad. française 1974, p. 133

CHAPITRE 4

ASPECS : UN PROCESSUS D'INGÉNIERIE LOGICELLE POUR L'ANALYSE, LA CONCEPTION, L'IMPLANTATION ET LE DÉPLOIEMENT DE SYSTÈMES COMPLEXES

LE MÉTAMODÈLE CRIO fournit les abstractions nécessaires pour décrire la décomposition hiérarchique d'un système complexe en un nombre quelconque de niveaux d'abstraction. Dans ce chapitre ce métamodèle est doté d'un processus méthodologique pour guider le développement de solutions sur la base du paradigme des systèmes multi-agents holoniques.

Sommaire

4.1 Introduction

D'après [Simon, 1996], les systèmes complexes exhibent souvent une configuration hiérarchique[1]. L'idée que l'architecture d'un système complexe puisse être décrite, expliquée et comprise en utilisant des *structures organisationnelles hiérarchiques*[2] est partagée par un nombre important de scientifiques [Edwards, 2003, Kofman, 2001, Simon, 1996, Wilber, 1995]. De nombreux métamodèles et méthodologies ont déjà été proposés pour les SMA, mais la plupart d'entre eux considèrent les agents comme des entités atomiques. Considérer les agents comme des entités composées permet de faciliter la modélisation des hiérarchies de composition et de leurs dynamiques. Une telle approche est donc susceptible d'offrir un moyen plus adapté à la modélisation des systèmes complexes.

Partant de ce constat général, l'approche défendue dans cette thèse propose d'utiliser les systèmes multi-agents holoniques pour faciliter le développement de systèmes logiciels complexes. Ce chapitre introduit un processus de développement orienté-agent nommé ASPECS[3]. Il est basé sur le métamodèle CRIO décrit au chapitre précédent et capitalise ainsi l'expérience acquise dans la conception organisationnelle et holonique de RIO et de ses extensions [Hilaire et al., 2000, Rodriguez, 2005]. ASPECS peut également être considéré comme l'extension des méthodologies PASSI[4] et AgilePASSI [Chella et al., 2004, 2006, Cossentino et al., 2004, Cossentino, 2005] dont la philosophie et les grandes étapes du processus ont constitué l'une des bases de la conception de ce nouveau processus. Comme ces deux méthodologies, le cycle de développement d'ASPECS est basé sur un processus itératif. La construction de ce nouveau processus de développement a été effectuée sur la base du paradigme situationnel d'étude des méthodes[5] [Brinkkemper et al., 1996, Henderson-Sellers, 2003, Rolland and Prakash, 1996] et sur les applications de ce paradigme aux méthodologies orientées-agent [Cossentino and Seidita, 2005, Cossentino et al., 2006b, 2007, Seidita et al., 2006]. Cette approche spécifique prescrit tout d'abord la définition d'un métamodèle pour la conception de SMA. Elle considère ensuite ce métamodèle comme une nouvelle source de contraintes et de besoins pour la conception du processus de développement. Le métamodèle CRIO a été utilisé comme point de référence pour identifier les différentes activités du processus ASPECS. La description complète de la méthode adoptée pour construire ASPECS à partir du métamodèle CRIO sort du cadre de ce document.

ASPECS fournit un guide complet, depuis l'analyse des besoins jusqu'à l'implantation et au déploiement, permettant la modélisation d'un système à différents niveaux de détails, en procédant par raffinements successifs. L'approche holonique offre la possibilité

[1] Hierarchie signifie ici hiérarchie faible ou *"loose hierarchy"*, une hiérarchie d'après Simon est un système composé de sous-systèmes inter-connectés, eux-mêmes structurellement hiérarchiques, et ce, jusqu'à ce que l'on atteigne le niveau le plus bas composé de sous-systèmes considérés comme élémentaires. Dans la suite de ce document, le terme hiérarchie sera toujours relatif à cette définition fournie par Simon.

[2] *"Hierarchical organization structure"*, [Simon, 1996].

[3] ASPECS : *"Agent-oriented Software Process for Engineering Complex Systems"*.

[4] PASSI : *"Process for Agent Societies Specification and Implementation"*.

[5] *"Situational method engineering paradigm"*.

au concepteur de modéliser un système avec des entités de granularités différentes. Il peut ainsi récursivement décomposer un système en sous-systèmes, jusqu'à atteindre un niveau où la complexité des tâches demandées est suffisamment faible, pour être exécutée par des entités considérées comme atomiques et faciles à implanter.

Ce chapitre est organisé de la manière suivante : après une courte description générale du processus ASPECS (section 4.2), le processus de développement en lui-même est détaillé phase par phase (sections 4.3 à 4.6). Ces principales contributions sont finalement récapitulées à la section 4.7.

4.2 Description générale du processus ASPECS

ASPECS est un processus d'ingénierie logicielle qui décrit pas à pas les étapes à suivre pour le développement de logiciels, depuis l'analyse des besoins jusqu'à la production du code et au déploiement de celui-ci sur une plate-forme spécifique. Il est basé sur le métamodèle CRIO qui définit les principaux concepts pour l'analyse, la conception et l'implantation des systèmes multi-agents holoniques.

Le langage de modélisation adopté est UML. Afin de pleinement satisfaire les objectifs et les besoins spécifiques à l'approche orientée-agent, la sémantique et les notations d'UML ont été étendues, et de nouveaux *"profiles"* UML ont notamment été introduits.

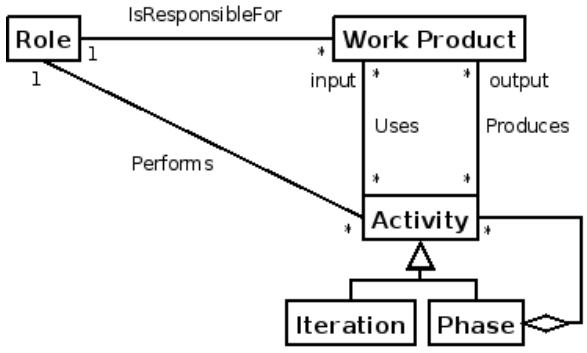


Figure 4.1: Un fragment du métamodèle SPEM

La description du processus ASPECS et de ses différentes composantes est basée sur le métamodèle de conception de processus de développement logiciel [SPEM, 2007] proposé par l'OMG. Le métamodèle SPEM considère le processus de développement comme une collaboration entre des entités abstraites actives, appelées *Rôles*, qui effectuent certaines opérations, appelées *Activités*, sur des entités concrètes et tangibles appelées *Produits* (*"Work Products"*). La figure 4.1 présente les relations entre ces différents concepts. SPEM distingue clairement les éléments réutilisables de leur application dans un processus donné. S'appuyant sur le métamodèle UML 2.0 [UML, 2003], SPEM fournit également un ensemble de *"profiles"* UML et d'icônes pour la description des processus de développement de lo-

giciels à l'aide de diagrammes UML, notamment les diagrammes de classes et d'activités (cf. figure 4.2). En accord avec le métamodèle SPEM, quatre composantes principales sont distinguées dans le processus de développement ASPECS : la *Phase*, l'*Activité*, la *Tâche* et l'*Étape*.

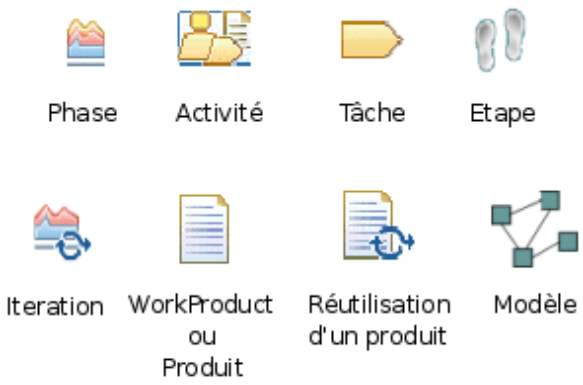


Figure 4.2: Sémantique associée aux icônes définies dans SPEM

Une *Phase* est composée de plusieurs *Activités*, et chaque activité est elle-même composée de plusieurs *Tâches*. Une *Tâche* peut à nouveau être décomposée en *Étapes*, considérées comme l'unité atomique de conception. Une *Phase* délivre un *Produit* (*"Work Product"*). Une *Activité* délivre un *Produit* principal tel un diagramme ou un document texte, et une *Tâche* contribue à la production d'une partie de ce *Produit* principal.

Le cycle de développement d'ASPECS est composé des quatre phases décrites ci-dessous :

1. L'*analyse des besoins* vise à fournir une description organisationnelle du système (décomposition hiérarchique du système). Elle doit également collecter les connaissances disponibles sur le domaine du problème et les organiser au sein d'une ontologie.
2. La *conception d'une société d'agents* cherche à construire le modèle d'un SMA, dont le comportement global doit être en mesure de fournir une solution au problème décrit dans la phase précédente. Les connaissances sur le système sont affinées et intègrent les éléments spécifiques à la solution proposée.
3. L'*implantation* de la solution décrit l'architecture des holons impliqués dans la solution et doit fournir le code source de l'application.
4. Le *déploiement* de la solution constitue la phase finale en charge du déploiement de l'application sur la plate-forme choisie.

Le lecteur, désirant davantage de détails, peut se reporter à la spécification complète du processus ASPECS[6]. Un exemple sera utilisé tout au long de ce chapitre afin d'illustrer les

[6]disponible à l'adresse suivante : `http://set.utbm.fr/index.php?pge=352&lang=fr`

différentes notations ainsi que les phases du processus de développement. Cet exemple est inspiré de la compétition de robots footballeurs organisée chaque année par la FIRA[7]. Le principe de cette compétition, créée en 1996, consiste à organiser des confrontations entre des équipes de robots autonomes qui jouent à un jeu proche du football [Kim, 1996]. Dans cet exemple, la coordination temps-réel et la collaboration entre les membres de chaque équipe est un point crucial pour réussir. Cet exemple est considéré comme un cas d'application et d'expérimentation classique, et comme un *"benchmark"* bien connu dans de nombreux domaines de recherche, tel que le traitement d'image ou les systèmes multi-agents.

Notre objectif consiste à concevoir un simulateur capable de simuler le comportement des équipes et à fournir ainsi une plate-forme de test pour les différentes stratégies de coordination entre les robots. Cet exemple a déjà été utilisé pour illustrer des approches de modélisation et d'implantation des SMA, notamment dans [Rodriguez et al., 2005b], MAGMA/Volcano [Ricordel, 2001], et pour comparer les approches AGR, MOISE et ISLANDER dans [Coutinho et al., 2005].

4.3 Phase d'analyse des besoins

La phase d'analyse des besoins doit fournir une description complète du problème sur la base des abstractions définies dans le *domaine du problème* du métamodèle CRIO. L'ensemble des activités qui composent cette première phase ainsi que leurs principaux produits sont décrits dans la figure 4.3.

Description des besoins du domaine Comme de nombreux processus de développement, ASPECS débute par une *description des besoins du domaine*. L'objectif consiste à dresser une première description du contexte de l'application et de ses fonctionnalités. Cette activité vise à identifier, classifier et hiérarchiser l'ensemble des besoins fonctionnels et non-fonctionnels des différentes parties prenantes du projet. Elle doit également fournir une première estimation du périmètre de l'application ainsi que de sa taille et de sa complexité. Une approche d'analyse des besoins, basée sur les diagrammes de cas d'utilisation UML, est adoptée. Une approche orientée-objectif ou -capacité peut également être employée [Van Lamsweerde, 2001].

La figure 4.4 présente les différents cas d'utilisation associés à l'exemple du simulateur de robots footballeurs pour la *FIRA Robot Soccer Cup*. Dans cet exemple, huit cas d'utilisation et un acteur ont été identifiés. L'acteur représente l'utilisateur qui peut simuler des matchs et affiner la stratégie associée à chacune des équipes. Simuler un match implique la simulation de deux équipes autonomes capables de choisir leurs propres stratégies. Chaque

[7] FIRA, Federation of International Robot-soccer Association, `http://www.fira.net`

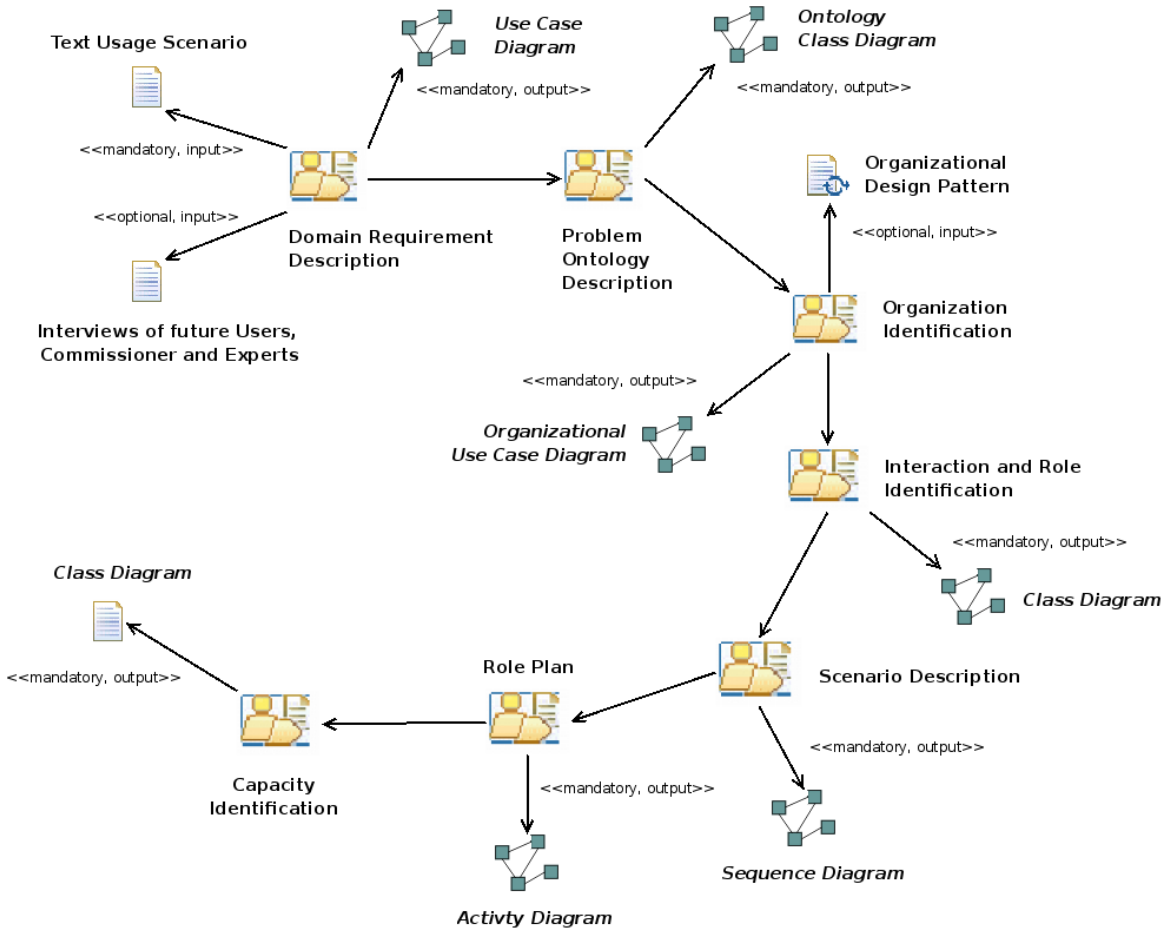


Figure 4.3: Phase d'analyse des besoins

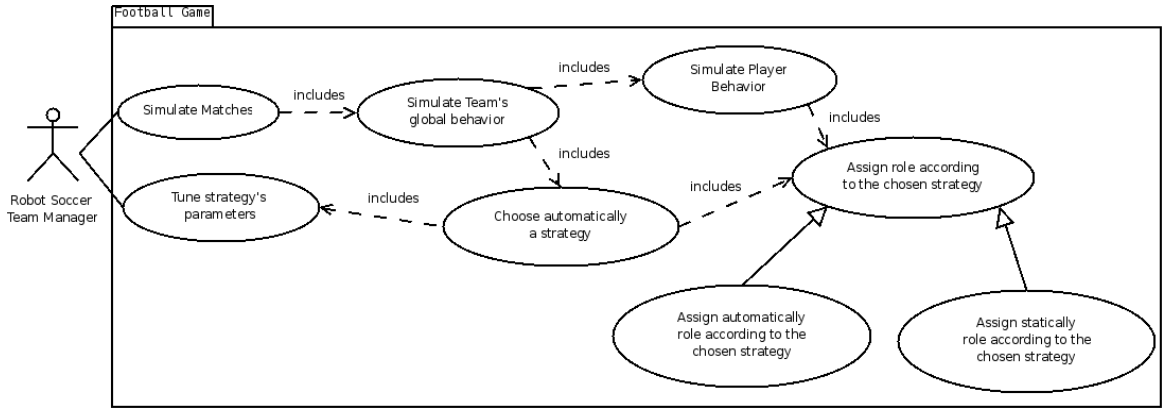


Figure 4.4: Diagramme de cas d'utilisation du simulateur de robots footballeurs

équipe est responsable de la simulation du comportement individuel de chaque robot footballeur.

Description de l'ontologie du problème Une fois les objectifs de l'application déterminés, l'ensemble des connaissances disponibles sur l'application et son contexte est capitalisé

dans l'ontologie du problème. La *description de l'ontologie du problème* doit fournir une première définition du contexte de l'application et du vocabulaire spécifique au domaine. Elle vise à approfondir la compréhension du problème, en complétant l'analyse des besoins et les cas d'utilisation, avec la description des concepts qui composent le domaine du problème, et de leurs relations. L'ontologie joue un rôle crucial dans le processus de développement ASPECS. En effet, sa structure sera un élément déterminant pour l'*identification des organisations*. L'ontologie est décrite ici en termes de concepts, d'actions et de prédicats. Elle est représentée à l'aide d'un *"profile"* UML spécifique pour les diagrammes de classes.

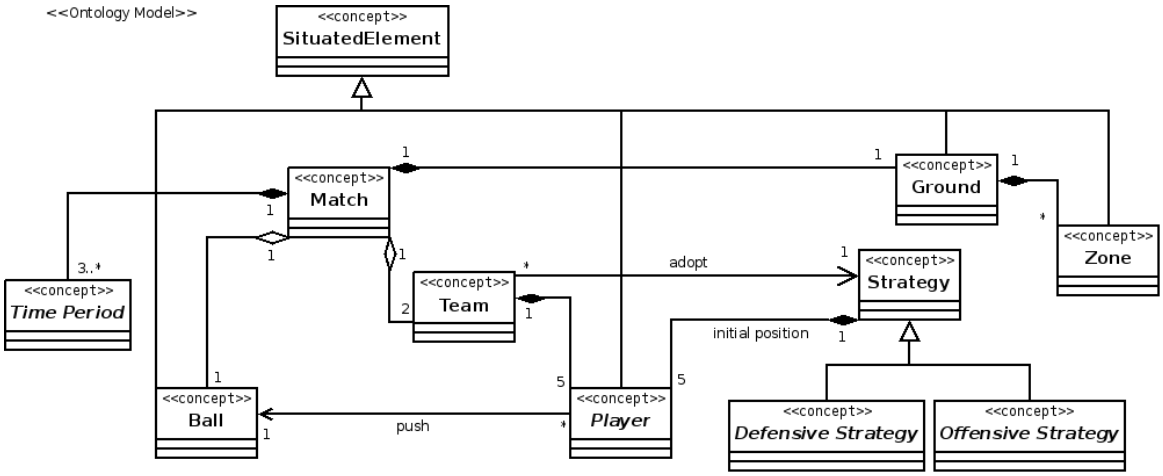


Figure 4.5: Fragment de l'Ontologie du problème du simulateur de robots footballeurs

L'ontologie du domaine associée aux robots footballeurs est partiellement décrite dans la figure 4.5. Cette ontologie représente les connaissances liées aux matchs de robots telles que la gestion des périodes temporelles, les différents éléments mobiles sur le terrain, les règles associées au jeu, etc. Dans CRIO et ASPECS, les règles (ou normes) associées à un domaine ou une organisation, ne sont pas explicitement représentées dans un modèle dédié, mais sont directement ajoutées soit dans l'ontologie du problème à l'aide de prédicats, soit dans les diagrammes des organisations correspondantes sous forme de contraintes OCL[8].

Identification des organisations L'*identification des organisations* doit établir une première décomposition organisationnelle du système et définir les objectifs de chaque organisation. Chacun des besoins, identifiés dans la première activité, se voit associer une organisation incarnant le comportement global en charge de le satisfaire ou de le réaliser. Le contexte de chacune de ces organisations est défini par un sous-ensemble des concepts de l'ontologie du problème. Le résultat de cette activité est ainsi incarné par un ensemble de triplets associant des concepts de l'ontologie, un ou plusieurs cas d'utilisation et une

[8] *"Object Constraint Language"* : langage de spécification de contraintes basé sur la logique et adaptée à la conception orientée-objet [OCL, 2006].

organisation. Les organisations, ainsi identifiées, sont directement ajoutées au diagramme de cas d'utilisation, sous la forme de *paquets* stéréotypés englobant les cas d'utilisation qu'elles sont en charge de satisfaire, comme décrit dans la figure 4.6. Cette étape de l'activité d'*identification des organisations* permet de fixer les objectifs d'un premier ensemble d'organisations.

ASPECS se basant sur un processus itératif, cet ensemble d'organisations peut ensuite être complété au fur et à mesure des itérations, afin de déterminer la hiérarchie organisationnelle représentant le système. Cette décomposition hiérarchique du système se poursuit jusqu'à un niveau où la complexité des comportements (rôles) est suffisamment faible pour être exécutée par des entités considérées comme atomiques et directement implantables. Les organisations sont alors représentées par des paquets stéréotypés dans les diagrammes de classes, conformément à la description fournie dans la figure 4.8, page 116. Les liens de compositions comportementale et hiérarchique entre organisations sont décrits par des contraintes dans les diagrammes de classe.

Différentes approches peuvent être adoptées pour partitionner les cas d'utilisation et ainsi faciliter l'*identification des organisations* du système. Dans l'exemple présenté, une étude structurelle du système basée sur l'ontologie du problème, est combinée avec une partition fonctionnelle des cas d'utilisation. (i) La première approche prend pour hypothèse que les membres d'une même organisation partagent et manipulent souvent les mêmes connaissances. Les cas d'utilisation liés aux mêmes concepts dans l'ontologie sont regroupés dans les mêmes organisations, ce qui fournit une décomposition essentiellement hiérarchique du système. (ii) La seconde approche considère que les membres d'une même organisation partagent les mêmes objectifs. Les cas d'utilisation, dont les fonctionnalités sont proches ou liées, sont regroupés dans une même organisation.

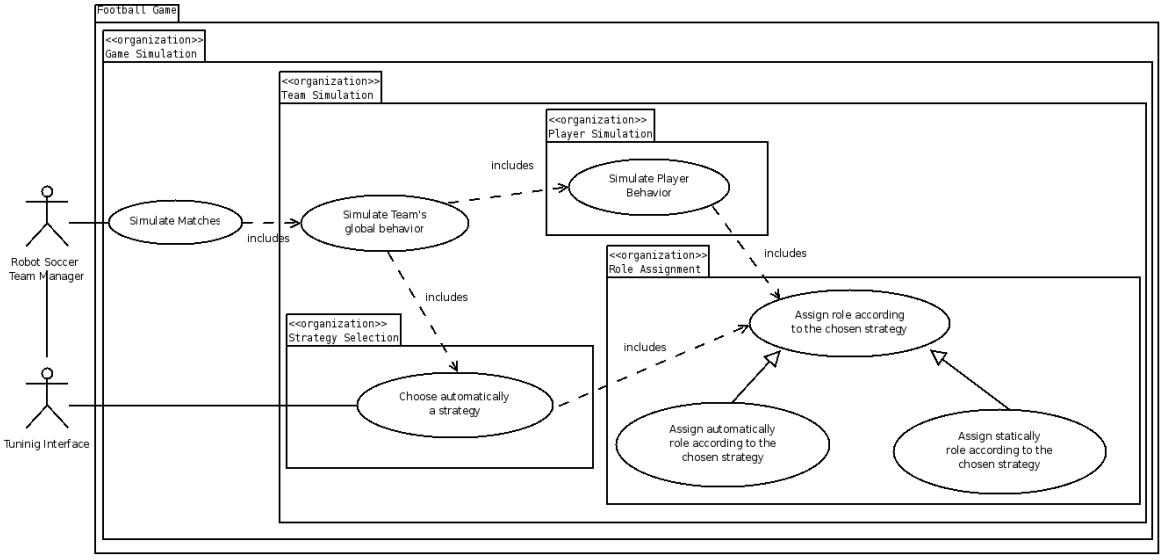


Figure 4.6: Quelques-unes des organisations du simulateur de robots footballeurs

En croisant ces deux points de vue sur le système et les partitions des cas d'utilisation

qui en découlent, on obtient le diagramme présenté sur la figure 4.6. Une première hiérarchie de cinq organisations est ainsi identifiée au sein du système (cf. figure 4.7) : (i) L'organisation *Game Simulation* correspond au comportement global du simulateur. (ii) L'organisation *Team Simulation* est en charge de simuler le comportement global d'une équipe de robots, depuis le choix de la stratégie à adopter jusqu'à son application sur le terrain. (iii) L'organisation *Player Simulation* simule le comportement des robots footballeurs d'une équipe. (iv) L'organisation *Strategy Selection* doit déterminer la meilleure stratégie à adopter en fonction de la situation courante du jeu. Une stratégie correspond à une distribution particulière de postes, tactiques ou stratégiques, à répartir entre les différents membres de l'équipe. On distingue généralement deux types de postes tactiques : les postes défensifs comme le gardien de but ou le défenseur central, et les postes offensifs comme l'ailier ou le tireur. (v) L'organisation *Role Assignment* doit appliquer la stratégie choisie, et associer à chaque membre de l'équipe l'un des postes tactiques définis dans la stratégie.

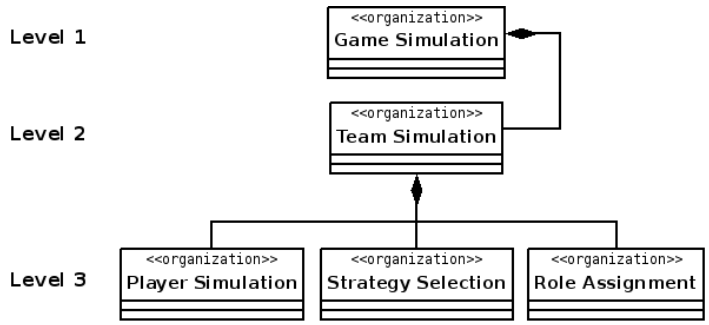


Figure 4.7: Première hiérarchie organisationnelle du simulateur de robots footballeurs

Identification des interactions et des rôles Le contexte et les objectifs de chaque organisation sont désormais connus. L'*identification des interactions et des rôles* vise à décomposer le comportement global incarné par une organisation en un ensemble de rôles en interaction.

Cette activité doit également décrire les responsabilités de chaque rôle dans la satisfaction des besoins associés à leurs organisations respectives. Chaque rôle est associé à un ensemble de concepts dans l'ontologie, généralement une sous-partie de ceux associés à son organisation. Grâce aux *Boundary Roles*[9], le périmètre de l'application et la frontière entre le système et son environnement sont ici affinés. Les rôles et les interactions composant chaque organisation sont ajoutés à leurs diagrammes de classe comme le décrit la figure 4.8 de la page 116. Un rôle est représenté par une classe stéréotypée, et une interaction entre deux rôles est représentée par une association entre les classes des rôles correspondants. Un ensemble de contraintes OCL est également ajouté au diagramme pour préciser le nombre d'instances autorisées pour chacun des rôles qui composent les différentes organisations du

[9]Concept du métamodèle CRIO défini au chapitre 3 à la page 78

système. À noter également que, dans cette figure, le lien *"contributes to"* signifie qu'une organisation contribue pour partie au comportement d'un rôle situé à un niveau d'abstraction supérieur.

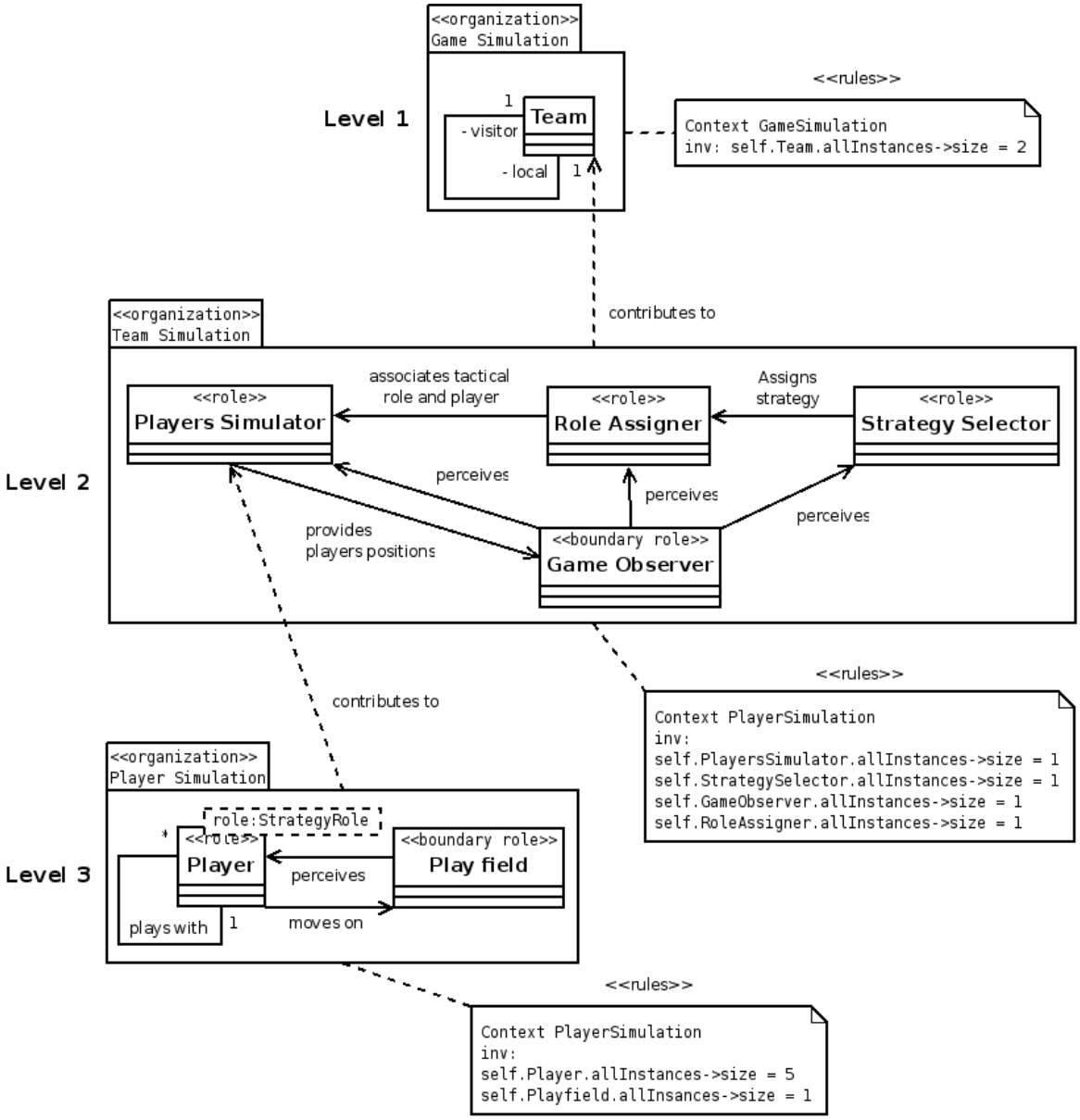


Figure 4.8: Description des interactions et des rôles de quelques organisations du simulateur de robots footballeurs

Toutes les organisations, qui composent le simulateur de robots footballeurs, ne seront pas détaillées, dans la suite de ce chapitre, l'étude se focalisera essentiellement sur l'organisation *Team Simulation*. Celle-ci est composée des quatre rôles suivants :

***Game Observer* :** C'est un rôle frontière (*Boundary Role*), il est chargé de maintenir l'état de la partie et de fournir les perceptions aux autres rôles. Il est situé à la frontière du système puisque l'état de la partie dépend d'éléments extérieurs aux comportements des robots traités par le simulateur (les règles du jeux, le terrain fixant les limites

aux déplacements des robots, etc). Ce rôle est ainsi nommé, car il devra observer les déplacements des robots des deux équipes et ceux de la balle pour assurer la mise à jour de l'état de la partie.

***Strategy Selector* :** À partir de la situation courante de la partie fournie par le rôle précédent, le *Strategy Selector* se doit de sélectionner la meilleure stratégie à adopter pour l'équipe.

***Role Assigner* :** Appliquer une stratégie implique d'associer à chaque membre de l'équipe l'un des postes tactiques définis dans la stratégie. Cette assignation est assurée par le rôle *Role Assigner* en fonction de la position courante de chacun des joueurs de l'équipe.

***Players Simulator* :** Ce dernier rôle est en charge de simuler le comportement des joueurs de l'équipe en fonction de la tactique choisie.

À chaque niveau de la hiérarchie organisationnelle, la complexité des rôles identifiés est étudiée. Si celle-ci est considérée comme trop importante pour être gérée par une entité atomique, une nouvelle organisation de niveau d'abstraction inférieur devra alors être identifée pour répartir la complexité du rôle. Cette identification engendre un nouvelle itération avec l'activité d'*identification des organisations*. L'objectif consiste à décomposer les tâches, considérées comme trop complexes au niveau n de la hiérarchie, en un ensemble de tâches plus simples, dont la résolution est répartie entre les rôles du niveau $n-1$. Chacun de ces rôles de niveau $n-1$ dispose ainsi d'une complexité inférieure au rôle de niveau n. Les tâches de niveau n sont en fait résolues de manière collaborative par un ensemble de rôles de niveau $n-1$. Le système est ainsi successivement décomposé niveau par niveau, jusqu'à atteindre un niveau où la complexité des tâches à effectuer est considérée comme suffisamment faible, pour être gérée par des entités atomiques faciles à implanter. Les contributions entre chaque niveau seront détaillées ultérieurement. Le processus de décomposition d'un système dans ASPECS est basé sur la définition récursive du concept d'organisation[10]. Selon le niveau d'abstraction considéré, une organisation peut être vue, soit comme un comportement unitaire, soit comme un ensemble de comportements en interaction. Cette dualité permet de briser la complexité d'un comportement de niveau n en la répartissant parmi un ensemble de comportements en interaction au niveau $n-1$.

Comme de nombreuses méthodologies, ASPECS intègre dans son processus des activités de description statique et dynamique du système. L'intégration de ces deux points de vue sur le système est à la base des méthodes de conception orientées-objet. Les organisations et leurs composantes ont été statiquement modélisées dans les deux activités précédentes. Les deux prochaines sont dédiées à la description de la dynamique du système.

Description des scénarios d'interaction Les objectifs, les rôles et les interactions de chaque organisation sont désormais identifiés. L'objectif de cette activité consiste à préci-

[10]Cette définition est fournie au chapitre 3, section 3.3.2, page 74.

ser les interactions entre les rôles pour donner naissance à un comportement de plus haut niveau. Un scénario d'interaction décrit les interactions entre les rôles définis au sein d'une organisation donnée et précise les moyens de coordination entre eux pour satisfaire les objectifs assignés à leur organisation. En conséquence, chaque organisation est associée à au moins un scénario ; ce dernier pouvant impliquer des rôles définis dans des organisations différentes. En effet, une organisation requiert généralement de l'information provenant d'autres organisations (situées au même niveau d'abstraction ou à un niveau différent). Les scénarios détaillent la séquence d'arrivée ou de transfert de ces informations. La *description des scénarios d'interaction* est supportée par un ensemble de diagrammes de séquences UML.

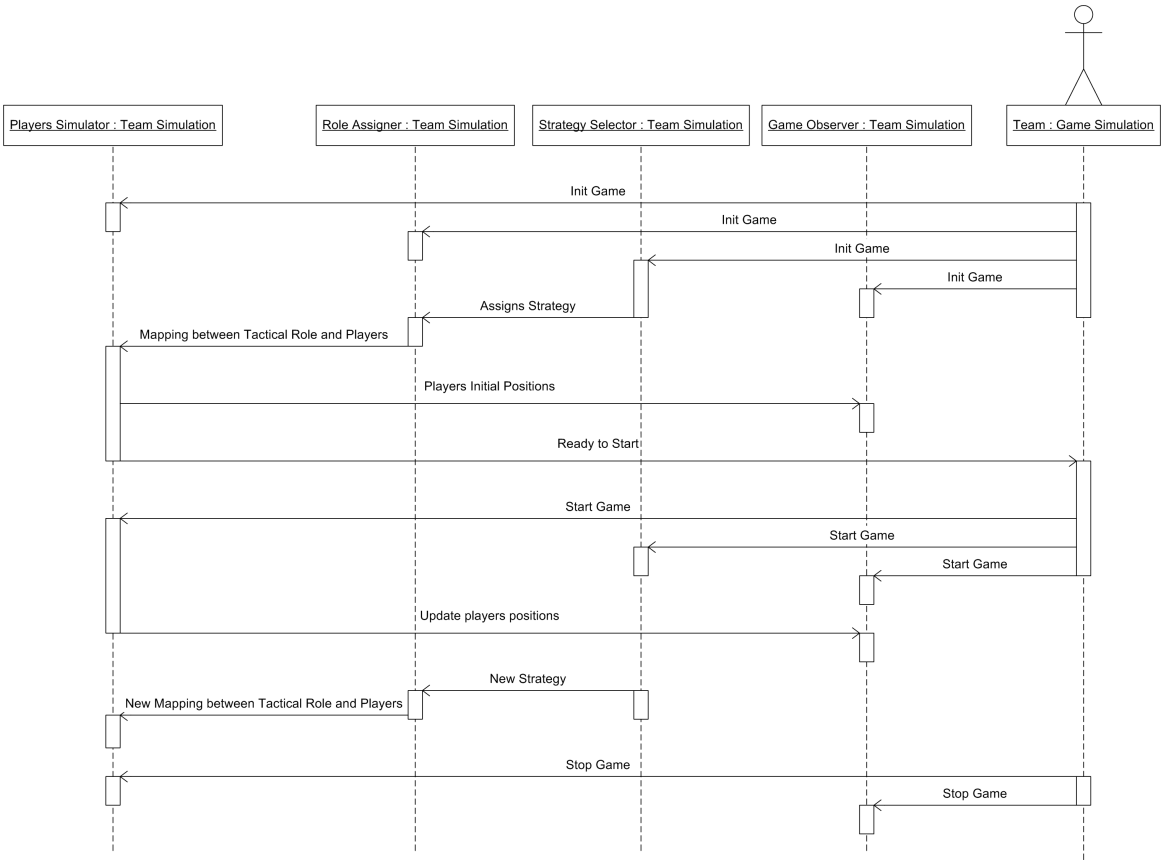


Figure 4.9: Description d'un des scénarios d'interactions de l'organisation *Team Simulation*

Le scénario d'interaction associé à l'organisation *Team Simulation* est décrit dans la figure 4.9. Chacun des quatre rôles de cette organisation dispose de sa propre ligne de vie. Le niveau supérieur de la hiérarchie organisationnelle est représenté dans ce diagramme par l'acteur *Team : Game Simulation* (manière utilisée ici pour représenter les interactions entre des organisations différentes ou situées à des niveaux hiérarchiques différents). Ce scénario débute par l'envoi des messages d'initialisation depuis l'acteur *Team*. Le rôle *Strategy Selector* décide ensuite de la stratégie à adopter qu'il transmet alors au *Role Assigner*. Ce dernier décide à son tour de la distribution des rôles tactiques parmi les joueurs. Les

robots sont ensuite positionnés sur le terrain et le rôle *Players Simulator* informe alors l'acteur *Team* que la simulation est prête à démarrer (message *ready-to-start*). La partie débute réellement quand l'utilisateur le décide. Cette information sera transmise via l'acteur *Team* grâce à l'événement *start*. La manière explicite de transférer l'information vers le rôle *Team*, situé au niveau d'abstraction supérieur, sera détaillée durant la phase de conception. Le transfert d'information est en fait basé sur la notion de service. Les interactions décrites dans le scénario d'interaction, faciliteront ensuite la spécification de l'interface du service correspondant.

Description des plans de comportement des rôles La *description des plans de comportement des rôles* vise à spécifier le comportement de chaque rôle en accord avec les objectifs qui lui ont été assignés et les interactions dans lesquelles il est impliqué. Chaque plan de comportement décrit la combinaison et l'ordonnancement des interactions, des événements extérieurs et des tâches qui composent le comportement de chaque rôle. Dans une seconde itération, la définition des tâches faisant usage de capacités pourra être affinée. Enfin, cette activité vise également à affiner l'association entre les rôles et les connaissances qu'ils manipulent (concepts de l'ontologie). Chaque plan de comportement est décrit par un diagramme d'activités UML ou par un statechart.

L'ensemble des plans de comportement des rôles de l'organisation *Team Simulation* sont décrits dans la figure 4.10. Le diagramme d'activités correspondant est partitionné en 5 lignes de vie, une pour chaque rôle, la dernière étant consacrée aux événements provenant de l'extérieur de l'organisation. Les interactions avec l'extérieur de l'organisation sont toutes représentées en utilisant une approche à base d'événements (*AcceptEventAction* et *SendSignalAction* des diagrammes d'activités, cf. [UML, 2007, p416-417]). Chacune des interactions précédemment identifiées dans le scénario d'interaction doit apparaître dans le plan comportemental, soit sous la forme d'un flux de contrôle ou de donnée, soit sous la forme d'un événement. Cette correspondance permet de vérifier que le plan de comportement en cours de modélisation satisfait effectivement les différents scénarios dans lesquels le rôle correspondant est impliqué.

Identification des capacités Cette activité vise à augmenter la généricité des comportements des rôles, en séparant clairement la définition de ces comportements des dépendances externes de leurs organisations. Elle consiste notamment à raffiner le comportement des rôles, pour faire abstraction de l'architecture des entités qui vont les jouer, et assurer leur indépendance à l'égard de tout élément extérieur à la définition du rôle lui-même. L'*identification des capacités* doit déterminer l'ensemble des compétences nécessaires à chaque rôle. La création d'une capacité dépend évidemment du niveau de dynamisme et de généricité que le concepteur souhaite apporter à une organisation. Le principe de base consiste à déterminer s'il faut laisser une ou plusieurs possibilités d'implantation pour une

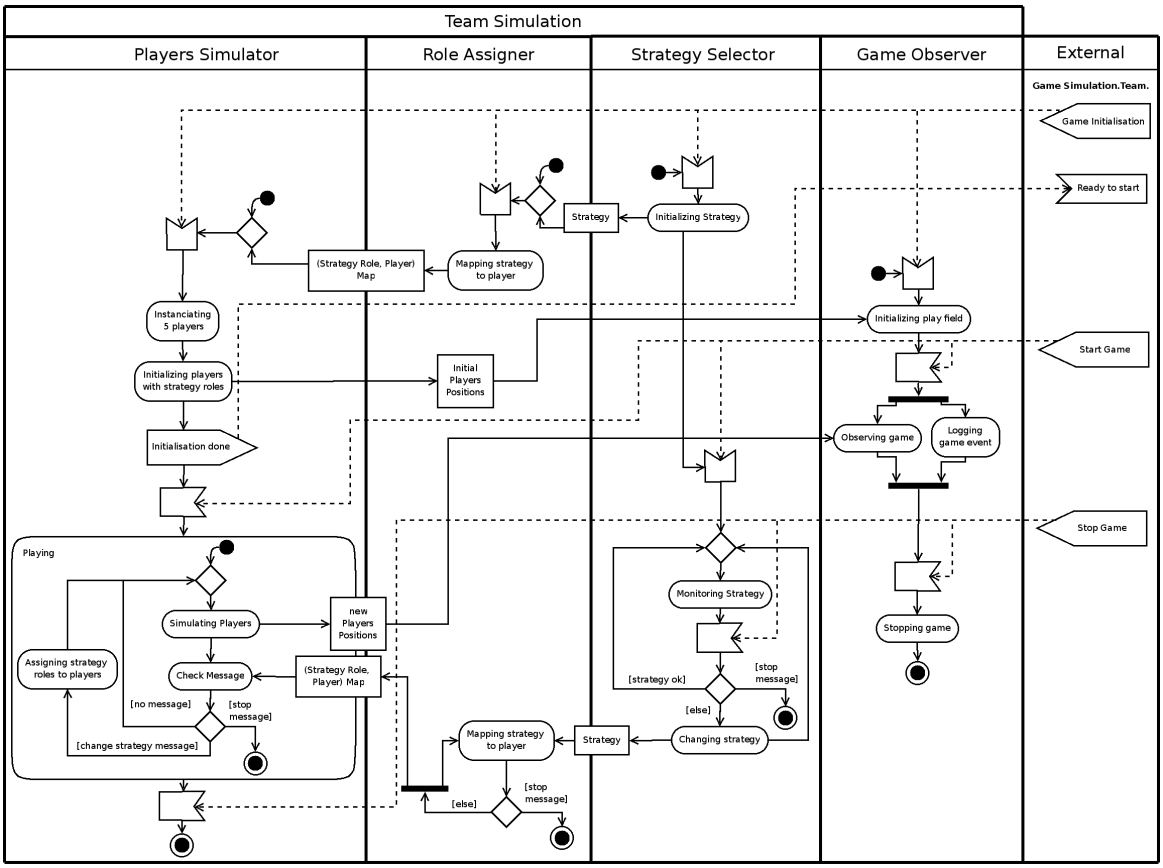


Figure 4.10: Description des plans de comportement des rôles de l'organisation *Team Simulation*

fonction donnée. Si c'est le cas, une capacité est alors requise. De même, tout accès à une ressource externe exige la création d'une capacité. Les capacités sont ajoutées sous forme de classes stéréotypées dans les diagrammes de classe UML des organisations concernées ; et elles sont reliées par une association aux rôles qui les requièrent (cf. figure 4.11).

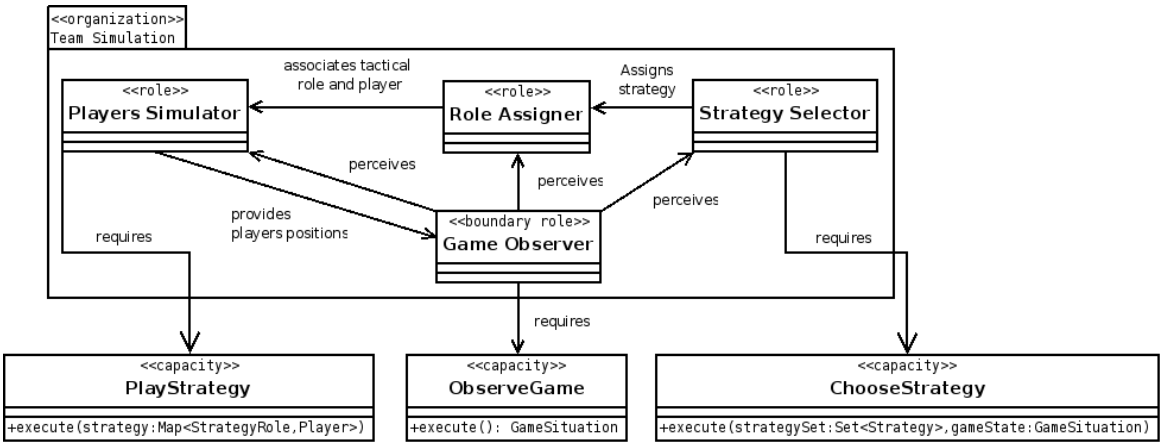


Figure 4.11: Identification des capacités requises par les rôles de l'organisation *Team Simulation*

Dans notre exemple, le rôle *Strategy Selector* est en charge de déterminer la meilleure stratégie à adopter en fonction de la situation courante du jeu. Le choix peut être effectué et être implanté de diverses manières. Cet aspect peut donc être extrait du comportement du rôle et modélisé via une capacité spécifique qui sera nommée par la suite *ChooseStrategy*. De la même manière, le rôle *Game Observer* doit maintenir l'information relative à l'état de la partie en cours (position des joueurs, de la balle, score, etc). Le mécanisme d'observation du jeu peut, lui aussi, être implanté de différentes manières et il requiert dans la plupart des cas des droits spécifiques. Une capacité, nommée *ObserveGame*, a également été conçue pour gérer cet aspect. La figure 4.11 décrit les différentes capacités associées à l'organisation *Team Simulation*.

4.4 Phase de conception de la société agent

À la fin de la phase d'analyse, le périmètre de l'application à développer, ainsi que la hiérarchie organisationnelle qui la compose, sont désormais identifiés et partiellement spécifiés. À chaque niveau de cette hiérarchie, les organisations ainsi que les rôles et les interactions qui la composent sont décrites. Une première spécification du comportement des rôles a déjà été effectuée et pour chacun d'eux un premier ensemble de capacités a déjà été identifié. L'objectif consiste désormais à élaborer le modèle d'une société d'agents dont le comportement global doit être en mesure de fournir une solution au problème décrit dans la phase précédente. Après avoir modélisé le problème en termes d'organisations, de rôles, d'interactions et de capacités, le modèle de la société d'agents est décrit en termes de communication et de dépendances entre entités (agents et holons). Cette description est essentiellement basée sur les concepts du domaine agent du métamodèle CRIO. Les connaissances sur le système sont affinées et intègrent les éléments spécifiques à la solution proposée. Le comportement des rôles est affiné sur la base des concepts nouvellement introduits, et associé aux agents en charge de les exécuter. Les interactions sont également raffinées pour décrire précisément leur contenu et leur éventuel protocole.

L'ensemble des activités qui composent cette phase de conception de la société agent, ainsi que les principaux produits associés, sont décrits dans la figure 4.12.

La première activité de la phase de conception est consacrée à la *description de l'ontologie de la solution*. Elle cherche à affiner l'ontologie du problème, afin d'y intégrer les nouvelles connaissances relatives à la solution en cours de développement et notamment celles qui seront utilisées pour décrire les informations échangées lors des communications entre rôles. Cette extension implique l'ajout de nouveaux concepts et le raffinement de concepts déjà présents. De nouvelles actions sont également ajoutées, en accord avec les capacités identifiées durant l'activité précédente, ainsi que de nouveaux prédicats pour spécifier les règles qui régissent les organisations du système.

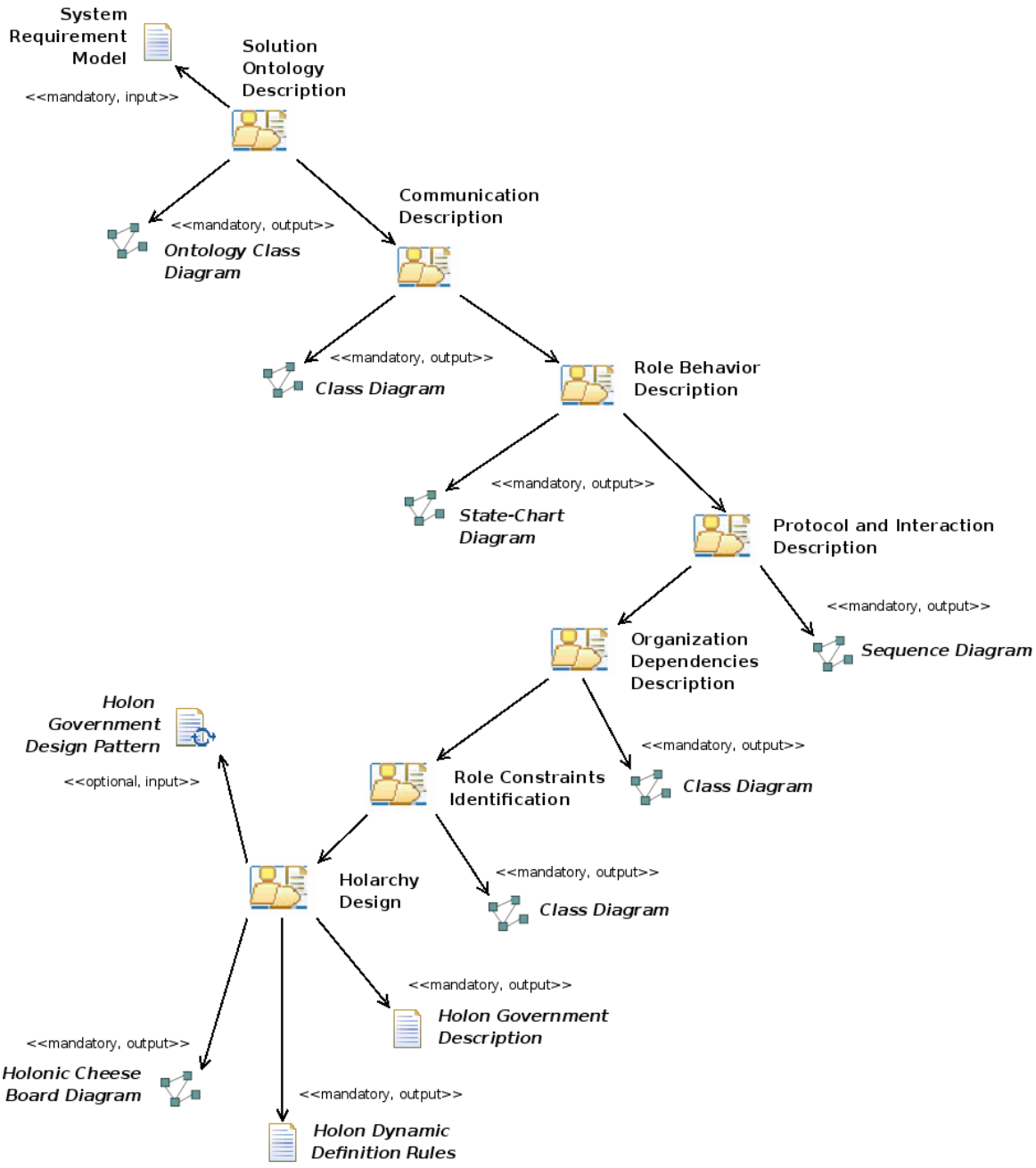


Figure 4.12: Phase de conception de la société agent

Description des communications Cette activité vise à décrire les communications et conversations entre les rôles de chacune des organisations du système. Chacune des interactions précédemment identifiées va être spécialisée dans cette activité en *communication* ou en *conversation*. Il est rappelé que les communications sont un moyen plus raffiné d'interagir, par rapport aux interactions entre les rôles définis dans le domaine du problème. Le modèle de la communication entre rôles est basé sur l'hypothèse que deux rôles qui interagissent partagent des connaissances communes et donc une ontologie commune. Ce savoir partagé est représenté dans chaque communication (et conversation) par un ensemble

d'éléments d'ontologie (Concept, Action, Prédicat). Une conversation, quant à elle, est réglementée par un protocole d'interaction et un langage défini. L'ensemble des protocoles est décrit en accord avec les spécifications de la FIPA [FIPA ACL, 2002]. Dans ces standards largement adoptés dans ASPECS, les conversations sont décrites sous la forme d'actes de langage [Searle, 1969]. L'activité de *description des communications* utilise les diagrammes de classe UML pour fournir une description statique des communications entre rôles : les rôles sont représentés par des classes et les communications par des associations (cf. figure 4.13). Les attributs des communications (ontologie, langage, protocole) sont, pour leur part, spécifiés dans des classes d'association.

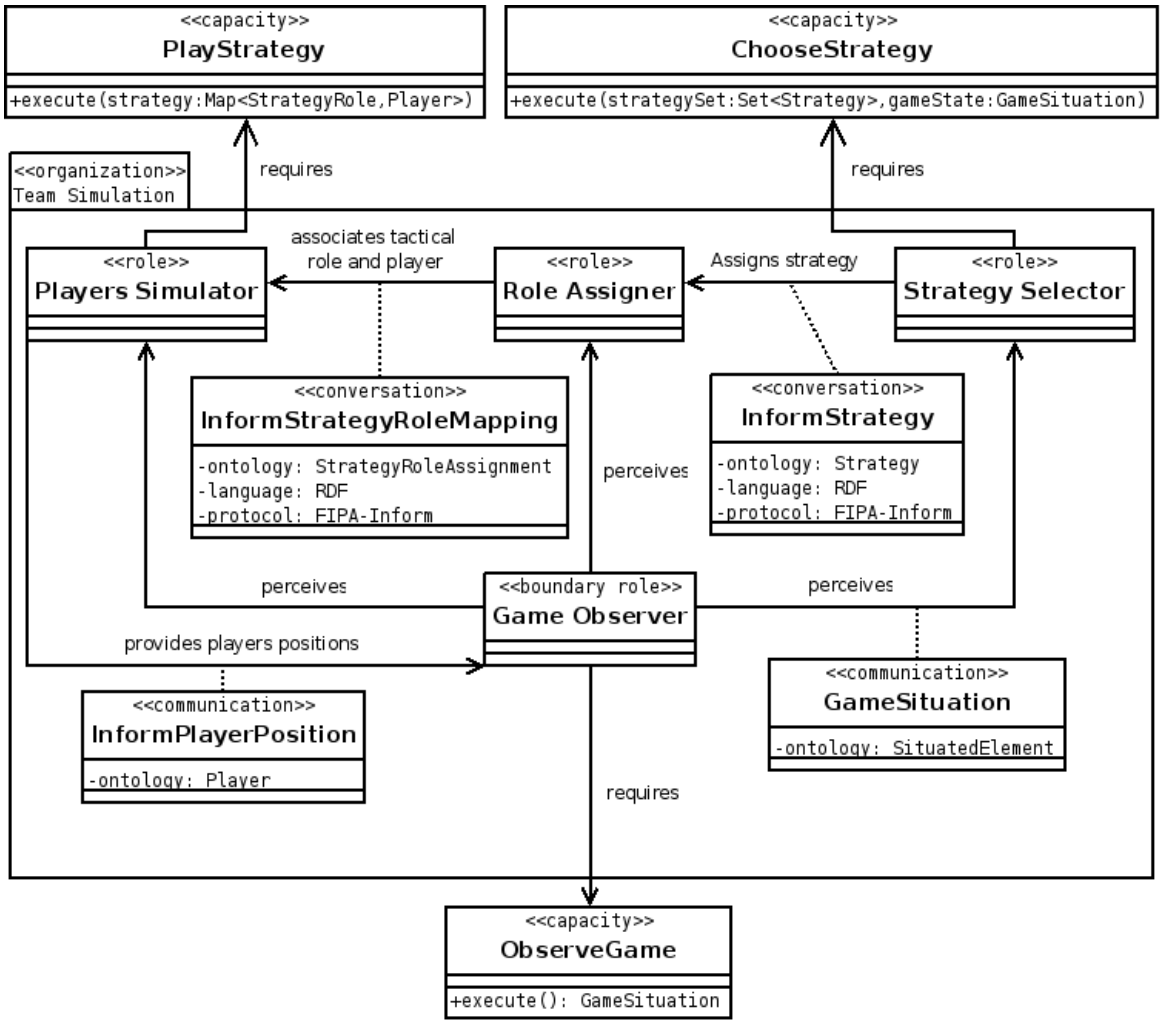


Figure 4.13: Description de quelques-unes des communications de l'organisation *Team Simulation*

Dans l'organisation *Team Simulation*, six interactions avaient précédemment été identifiées. Deux d'entre elles sont désormais modélisées par des conversations, les autres sont représentées par des communications. Par exemple, l'interaction entre les rôles *Players Simulator* et *Role Assigner* est désormais considérée comme une conversation respectant le protocole *FIPA Inform Communicative Act* [FIPA Com. Act, 2002], manipulant le prédicat

Role Assignment défini dans l'ontologie de la solution et adoptant le langage de contenu RDF [FIPA RDF, 2001]. La figure 4.13 résume la *description des communications* de l'organisation *Team Simulation*.

Description comportementale des rôles Cette activité vise à décrire le cycle de vie complet d'un rôle en intégrant les tâches, les capacités et les communications précédemment identifiées. À ce niveau d'abstraction, les tâches qui composaient le comportement des rôles, peuvent éventuellement être décomposées en éléments plus fins, représentés par la notion d'action. L'action est considérée comme l'unité atomique de spécification comportementale. Une des actions considérée comme les plus basiques consiste à invoquer une capacité. Dans cette activité, le comportement d'un rôle est décrit par un *"statecharts"* [Harel, 1987] ou une machine à états UML [UML, 2007]. Ces derniers sont en effet directement transformables en code source exécutable et sont déjà intégrés dans notre *"framework"* de description formelle des rôles basés sur le langage OZS [Gruer et al., 2004, Rodriguez et al., 2003, 2005a].

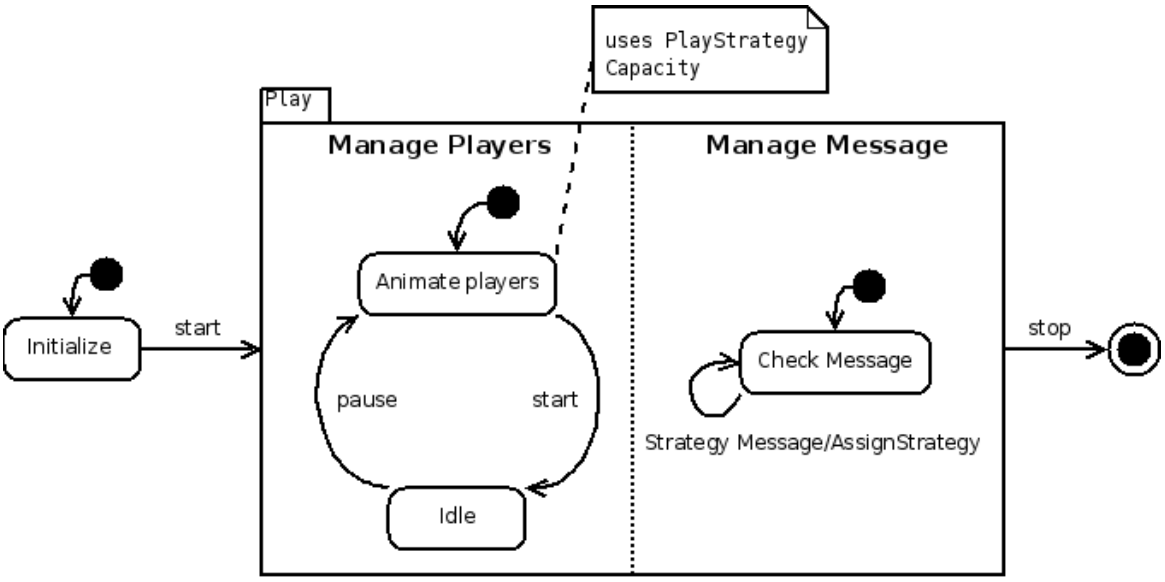


Figure 4.14: State-chart du rôle *Players Simulator* de l'organisation *Team Simulation*

La figure 4.14 présente le raffinement du plan de comportement du rôle *Players Simulator* fourni à la section 4.3, transcrit sous forme de statechart. L'état par défaut correspond à l'initialisation. Lorsque l'événement *start* est reçu, le rôle passe alors dans le super-état *Play* où il gère en parallèle l'arrivée des messages et la simulation des joueurs. Si un événement *pause* est reçu, la simulation du comportement des joueurs est suspendue. La gestion des messages consiste à attendre l'arrivée de messages fixant la nouvelle stratégie à adopter pour simuler les joueurs. L'animation des joueurs est également un super-état, il n'est pas détaillé dans cette figure, mais utilise la capacité *PlayStrategy* pour informer l'organisation de niveau inférieur lorsque les postes tactiques attribués aux joueurs changent.

L'activité de *description comportementale des rôles* peut éventuellement être suivi par une description des protocoles nouvellement introduits au cours du processus de développement. Cette activité est optionnelle, et n'est utilisée que lorsqu'aucun protocole existant répertorié par la FIPA[11], ne s'est avéré correspondre aux besoins de l'application en cours de développement. La description de ces nouveaux protocoles est basée sur le modèle de spécification utilisé par la FIPA [Odell et al., 2001] et sur les diagrammes de séquences AUML. Cette description vise notamment à faciliter la réutilisation de ces protocoles dans des applications ultérieures.

Description des dépendances entre organisations Cette activité se place dans la continuité de l'activité d'*identification des capacités* de la phase précédente. Ses objectifs sont d'identifier et de décrire l'ensemble des éléments extérieurs à une organisation, mais qui sont nécessaires à son bon fonctionnement. Les ressources de l'environnement manipulées par les rôles seront donc identifiées et décrites. Les capacités requises par les rôles et les services qu'ils fournissent seront également décrits. L'objectif est ensuite de dresser un annuaire complet, associant chaque capacité présente dans le système à l'ensemble des services susceptibles de les implanter. Cet annuaire pourra ensuite être utilisé pour informer les agents qui souhaitent obtenir dynamiquement une capacité donnée. L'association entre capacités et services permet également de valider et de compléter la spécification des contributions entre des organisations situées à des niveaux d'abstraction différents. Les services, les capacités et les ressources sont ajoutés aux diagrammes de classes des organisations qui leurs sont associés et sont reliés, entre eux et aux rôles correspondants, par des associations.

La figure 4.15 décrit les éléments externes à l'organisation *Team Simulation* nécessaires à son bon fonctionnement (capacités, services et ressources). Les contributions hiérarchiques entre organisations ont été détaillées grâce aux associations entre capacités et services. Dans l'organisation *TeamSimulation*, le rôle *Game Observer* maintient l'information relative à l'état de la partie en cours (position des joueurs, de la balle, score, etc). Afin de conserver une trace du déroulement complet d'une partie, il sauvegarde toutes les informations de son évolution dans un fichier journalisé (*"Logging File"*). Le rôle *Game Observer* accède à cette ressource via la capacité *LogCapacity*. Le rôle *Strategy Selector* est en charge d'assurer la gestion du service *SimulateTeamBehavior*, de même pour le rôle *Play Field* avec le service *PlayStrategy* dans l'organisation *Player Simulation*. Ces deux organisations exploitent le mécanisme composé de gestion des services, décrit au chapitre 3 (cf. section 3.4.5, page 99), où le service est obtenu depuis les interactions d'un sous-ensemble de rôles de la même organisation en suivant un protocole connu.

Identification des contraintes entre rôles Cette activité consiste à identifier les contraintes qui pourraient exister entre des rôles définis dans différentes organisations du système. Ces

[11] bibliothèque de protocoles d'interaction de la FIPA : `http://www.fipa.org/repository/ips.php3`

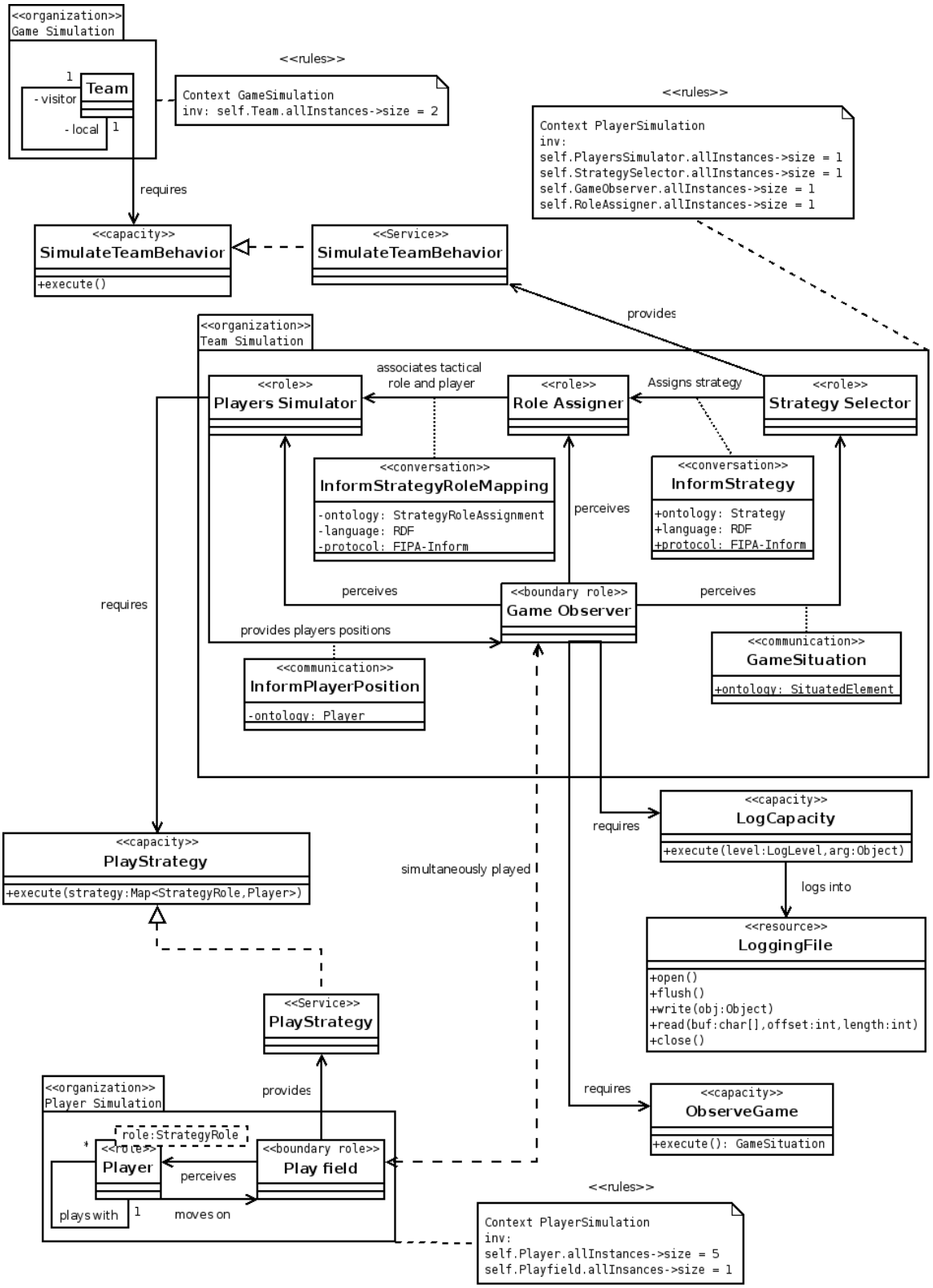


Figure 4.15: Description des dépendances externes de l'organisation *Team Simulation*

contraintes vont directement influencer la structure de la holarchie finale du système et contribuer à obtenir du système le comportement désiré et spécifié dans la phase précédente.

Identifier les contraintes entre les rôles implique concrètement d'identifier les rôles définis par des organisations différentes, mais qui permettent le transfert d'informations entre elles. En d'autres termes, ces contraintes représentent essentiellement les rôles qui doivent être joués simultanément. En effet, si deux rôles sont joués par le même agent, ils appartiennent alors tous deux au contexte d'interaction interne à l'agent et peuvent donc interagir, soit indirectement par modification des propriétés internes de l'agent, soit directement via un mécanisme de communication à base d'événements. Ce mécanisme constitue une alternative à l'utilisation des services et un moyen plus simple de transfert de l'information entre des organisations différentes. Les contraintes relatives au problème de simultanéité et de parallélisme sont également importantes, puisqu'elles influeront de manière notoire sur le choix futur des politiques d'ordonnancement des rôles. En somme, les contraintes qui empêchent des exécutions concurrentes ou qui forcent une séquence précise doivent être spécifiées.

Le résultat de cette activité est incarné par la liste détaillée de toutes les contraintes identifiées entre les rôles du système. Les contraintes majeures sont ensuite ajoutées aux diagrammes de classes des organisations impactées sous forme de contraintes UML ou OCL.

Dans l'exemple présenté, l'environnement est simulé en utilisant un unique agent. Cette décision est un choix du concepteur, et elle implique que tous les *boundary rôles* de l'application doivent être joués par un unique agent. Cette contrainte est traduite dans la figure 4.15, par une contrainte UML entre les deux *boundary rôles* : *Play Field* et *Game Observer*. Ce n'est pas une contrainte du système, mais une contrainte imposée par le concepteur ; toutefois, ces deux types de contraintes sont modélisés de manière analogue. Tous deux ont des conséquences importantes sur la structure finale de l'application qui sera détaillée dans la section suivante.

Conception des holarchies À ce stade du processus, l'ensemble des organisations du système, leurs rôles et les communications associées sont désormais complètement décrits et spécifiés. La *conception des holarchies* est la dernière activité de la phase de conception ; elle effectue une synthèse globale où les résultats de l'ensemble des travaux précédents sont combinés et résumés en un seul produit. Elle est consacrée à l'agentification de la hiérarchie organisationnelle et à la définition des entités en charge de l'exécuter. Son objectif consiste à définir les holons du système et à en déduire la structure de la holarchie.

Pour construire la holarchie de l'application, les organisations qui composent le système sont instanciées sous forme de groupes. Un ensemble de holons est ensuite créé à chaque niveau, chacun d'eux jouant un ou plusieurs rôles dans un ou plusieurs groupes du niveau considéré. Les relations de composition entre super-holons et sous-holons sont ensuite spécifiées en accord avec les contributions entre organisations définies dans la hiérarchie organisationnelle. Par exemple, les super-holons de niveaux $n + 1$ jouent des rôles dans les groupes de niveaux $n+1$. Les membres respectifs de ces super-holons jouent des rôles dans

les différents groupes de niveaux n, qui contribuent au comportement des rôles de niveau $n+1$ joués par leur super-holon. La hiérarchie organisationnelle est donc directement associée à une hiérarchie de holons (ou holarchie). Les règles qui régissent la dynamique de ces holons, ainsi que les types de gouvernement[12] de chaque type de holon composé, sont également décrits. Tous ces éléments sont ensuite synthétisés pour décrire la structure de la *holarchie initiale* du système. La notation utilisée pour représenter la structure statique d'une holarchie est inspirée des diagrammes *"cheese-board"* proposés par [Ferber et al., 2004]. Elle fut toutefois adaptée pour mieux représenter l'approche holonique.

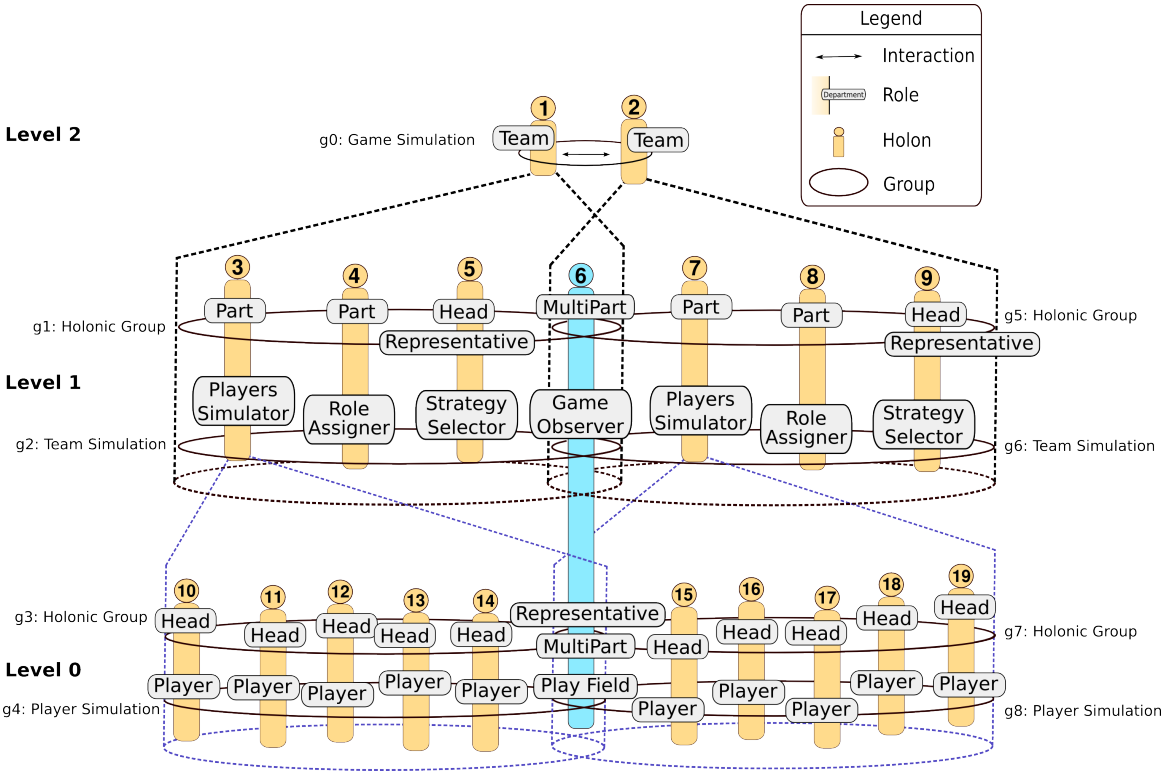


Figure 4.16: La structure holonique du simulateur de robots footballeurs

Un fragment de la structure finale de la solution holonique de l'exemple visant à développer un simulateur de robots footballeurs est présenté dans la figure 4.16.

Au niveau 2 de la holarchie, deux super-holons H_1 et H_2 jouent le rôle *Team* dans le groupe $g0$ *: Game Simulation*. Il est rappelé que cette dénomination signifie que le groupe $g0$ est une instance de l'organisation *Game Simulation*. Chacun de ces deux super-holons contient une instance de l'organisation Team Simulation (groupe $g2$ et $g6$).

L'ensemble des super-holons (H_1, H_2, H_3, H_7) contiennent une instance de l'organisation holonique qui définit la structure de leur gouvernement respectif. Chaque holon *Team* dispose d'un gouvernement des plus simples, inspiré par le mode de gouvernement de type Monarchie (cf section 3.4.4.4, chapitre 3, page 91). La règle est que le holon membre jouant

[12]Le type de gouvernement d'un super-holon est basé sur la manière de répartir les rôles holoniques parmi ses membres, cf. chapitre 3, page 91

le rôle *Strategy Selector* (H_5 et H_9) est automatiquement nommé *Head* et *Representative* des autres membres.

Les holons *Part* H_3 et H_7 jouant le rôle *Players Simulator* sont décomposés et contiennent une instance de l'organisation *Player Simulation*. Leur gouvernement est inspiré de l'apanarchie où tous les membres sont impliqués dans le processus de prise de décision (tous les holons sont *Head*).

Le holon atomique H_6 joue doublement le rôle MultiPart puisqu'il est partagé entre deux couples de super-holons (H_1,H_2) et (H_3, H_7). Ce holon représente la partie environnementale de l'application, il joue donc l'ensemble des *boundary roles* de l'application, en accord avec les contraintes entre rôles définies dans l'activité précédente.

À la fin de cette phase, la structure de la holarchie ainsi que l'architecture des holons qui la composent sont désormais connues. Nous pouvons alors procéder à leur implantation sur la plate-forme choisie.

4.5 Phase d'implantation

La phase d'implantation vise à fournir un modèle implantatoire de la solution multi-agents conçue durant la phase précédente. Cette étape est dépendante de la plate-forme d'implantation choisie. Dans ASPECS, la phase d'implantation est basée sur le modèle de la plate-forme *Janus*[13]. *Janus* est une plate-forme spécifiquement conçue pour implanter et déployer les systèmes multi-agents holoniques. Elle gère de manière native l'intégralité du modèle organisationnel et fournit l'ensemble des moyens nécessaires pour implanter les concepts du métamodèle CRIO : les holons, leurs rôles et les organisations correspondantes.

Sur la base de *Janus*, cette phase détaille l'ensemble des activités nécessaires pour concevoir l'architecture de la solution, produire le code source associé et le tester. Le processus décrit dans cette phase peut être utilisé avec d'autres plates-formes d'implantation, pour peu qu'une transition entre les concepts du modèle de celle-ci et ceux du domaine agent de CRIO ait été établie (ex : Transformation de modèle dans le cadre de l'approche de développement dirigée par les modèles [MDA, 2003]). Les activités qui composent cette phase d'implantation ainsi que les principaux produits fournis sont décrits par la figure 4.17.

Architecture de holon Cette activité vise à définir l'architecture de chaque holon impliqué dans la solution conçue durant la phase précédente, à l'aide des concepts décrits dans le métamodèle de la plate-forme d'implantation choisie. Chaque organisation ainsi que l'ensemble de leurs rôles et les tâches qui les composent doivent être décrits. Chaque holon est ensuite associé à l'ensemble des rôles qu'il doit jouer, ainsi qu'aux capacités et services qu'il possède. Les diagrammes de classe UML sont utilisés pour fournir la description statique de l'ensemble des éléments impliqués dans l'implantation de la solution.

[13]Cette plate-forme sera décrite au chapitre 5

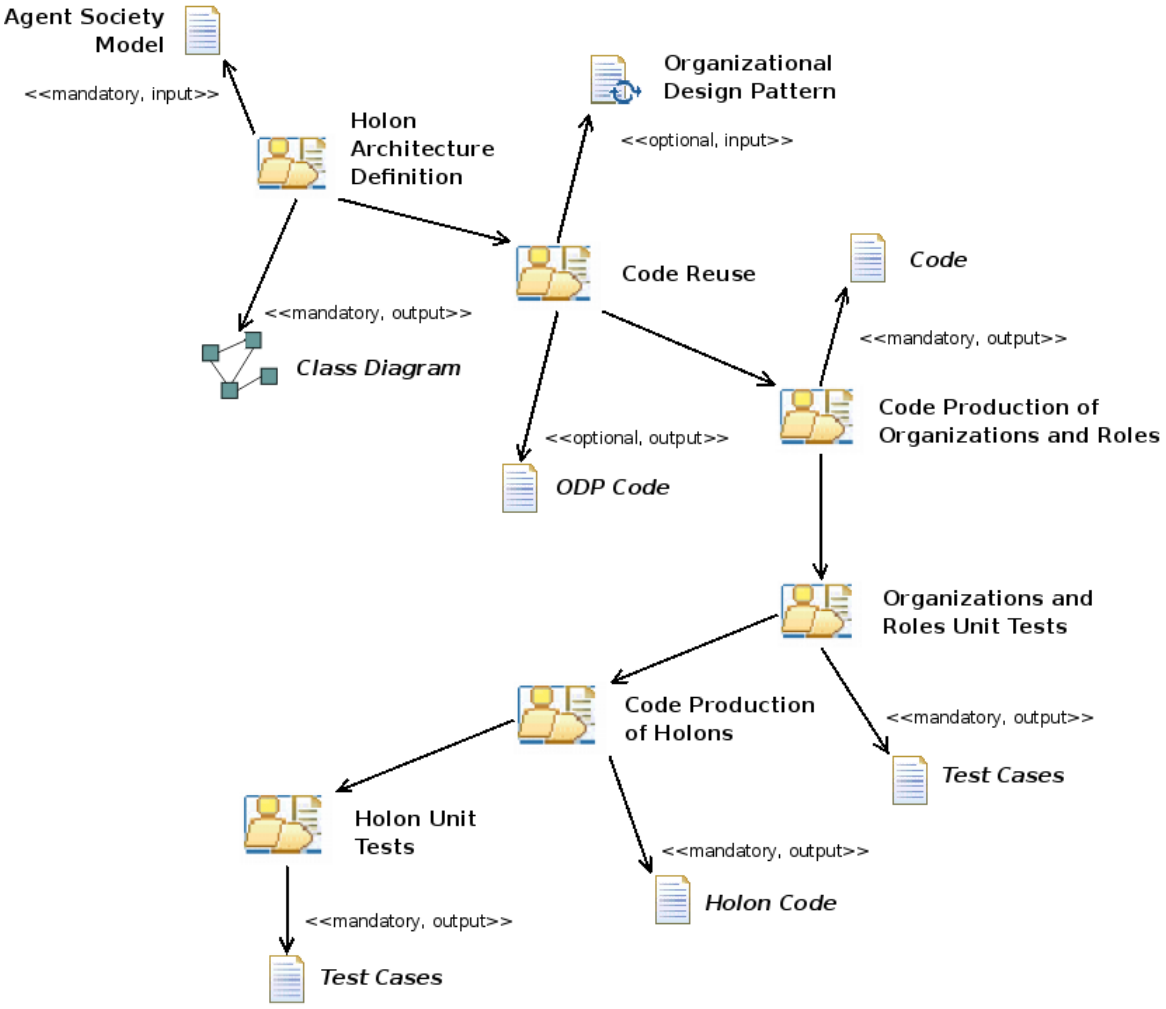


Figure 4.17: Phase d'implantation

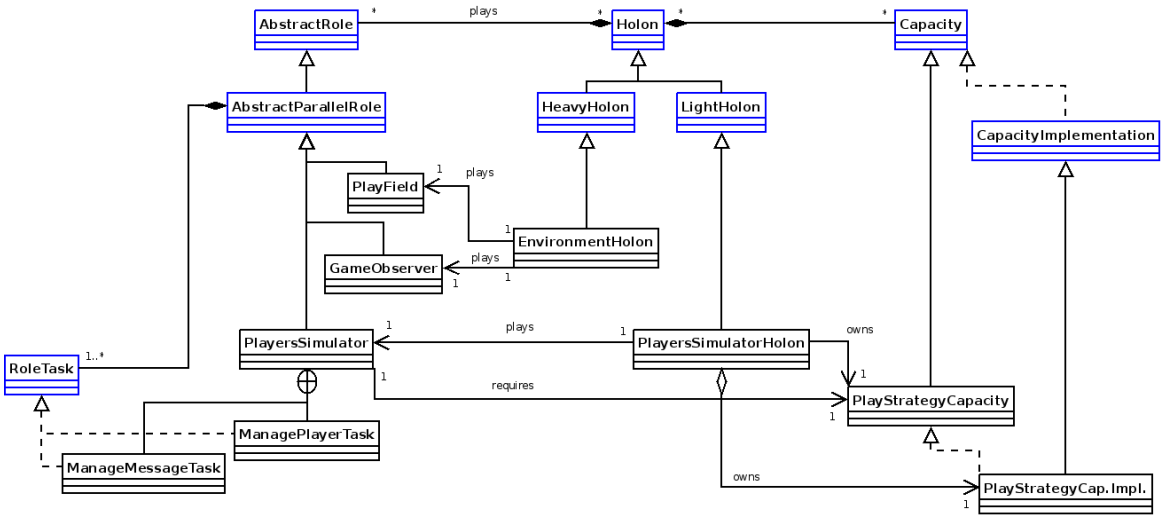


Figure 4.18: Quelques architectures des holons du simulateur de robots footballeurs

La figure 4.18 décrit une partie de l'architecture de la solution implantatoire associée à

l'exemple du simulateur de robots footballeurs. Certaines classes correspondent aux classes fournies par la plate-forme *Janus* et seront présentées au chapitre suivant (AbstractRole, Holon, Capacity, etc). Ces classes sont ensuite spécialisées pour définir les éléments propres à l'application. Dans cet exemple, chaque architecture de holon identifiée durant la *conception des holarchies* correspond à un holon spécifique dans l'implantation. Par exemple, le *PlayerSimulatorHolon* joue le rôle *Player* conformément à la description effectuée dans la *conception des holarchies*.

Les activités suivantes sont consacrées à la production du code (organisations, rôles et holons) et aux tests unitaires correspondants. Les rôles et les organisations deviennent des classes à part entière qui sont regroupées dans un *paquet* spécifique (un par organisation). Les organisations ainsi créées peuvent ensuite être regroupées pour créer un dépôt capitalisant l'ensemble des organisations développées au cours de différents projets. Ce dépôt facilitera ensuite la réutilisation des organisations pour le développement de nouvelles applications.

4.6 Phase de déploiement

La phase de déploiement est la dernière étape du processus ASPECS, elle vise à décrire les éléments nécessaires au déploiement de l'application produite durant la phase précédente. Ceci implique de détailler les étapes de son installation et de son exécution, dans son environnement de production, et éventuellement de sa mise à jour et de sa désinstallation. Dans le contexte des systèmes multi-agents et des systèmes logiciels complexes, la phase de déploiement prend un sens tout particulier. En effet, de tels systèmes se doivent d'être distribués, ouverts, et fortement dynamiques. Cette phase vise également à définir la position initiale (au moins) des agents au sein du système distribué. La figure 4.19 détaille les deux activités de cette phase ainsi que les principaux produits fournis.

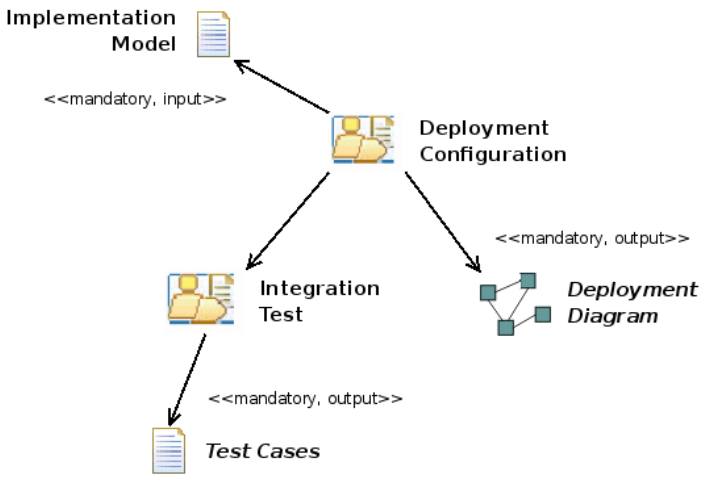


Figure 4.19: Phase de déploiement

L'activité de *configuration du déploiement* vise à détailler comment l'application précédemment développée sera concrètement déployée. La manière de distribuer l'application parmi les différents noyaux *Janus* disponibles devra donc être explicitée. Ces derniers sont connectés entre eux par une technologie *"peer-to-peer"*[14] permettant la création de fédérations de noyaux et la migration d'agents au sein de ces fédérations. La configuration du déploiement doit préciser la position physique de holons et de leurs éventuelles relations avec des ressources ou périphériques externes à l'application (capteur, base de données, etc).

La figure 4.20 décrit une partie du diagramme de déploiement du simulateur de robots footballeurs. Trois noyaux *Janus* sont utilisés pour distribuer cette application. Un noyau est utilisé pour exécuter chaque équipe et donc chaque holon *TeamSimulatorHolon* ainsi que ses membres. Le troisième noyau est dédié à l'exécution de l'interface graphique et du holon *EnvironmentalHolon* en charge de l'environnement de la simulation.

4.7 Conclusion

Ce chapitre a introduit le processus de développement logiciel ASPECS, conçu pour fournir un guide méthodologique pour l'exploitation des abstractions fournies par le métamodèle CRIO. ASPECS couvre l'intégralité du processus de développement, de l'analyse des besoins jusqu'au déploiement. Cette section résume brièvement les principales contributions associées à ce nouveau processus de développement.

Encourager la réutilisation et la modularité des modèles et des connaissances La réutilisation des modèles et l'utilisation des schémas de conception organisationnels est l'un des points clefs d'ASPECS. La définition du comportement des rôles, sur la base des capacités, permet d'augmenter la généricité des organisations et ainsi favoriser leur réutilisation dans de futures applications. La réutilisation est également encouragée dans la modélisation des connaissances. Grâce à l'ontologie, considérée comme la base de connaissances commune et transversale au processus de modélisation, les connaissances du domaine du problème sont regroupées et classifiées.

L'approche organisationnelle : de l'analyse à l'implantation ASPECS, utilisé de concert avec la plate-forme *Janus*, permet d'implanter les organisations et les rôles en tant qu'entités de premier ordre et indépendamment des entités qui les jouent. Ce point de vue permet de conserver les avantages de l'approche organisationnelle depuis l'analyse jusqu'à l'implantation.

[14] basée sur JXTA, Juxtapose : `https://jxta.dev.java.net` pour l'instant, en perspective il est prévu d'intégrer la plate-forme Jade.

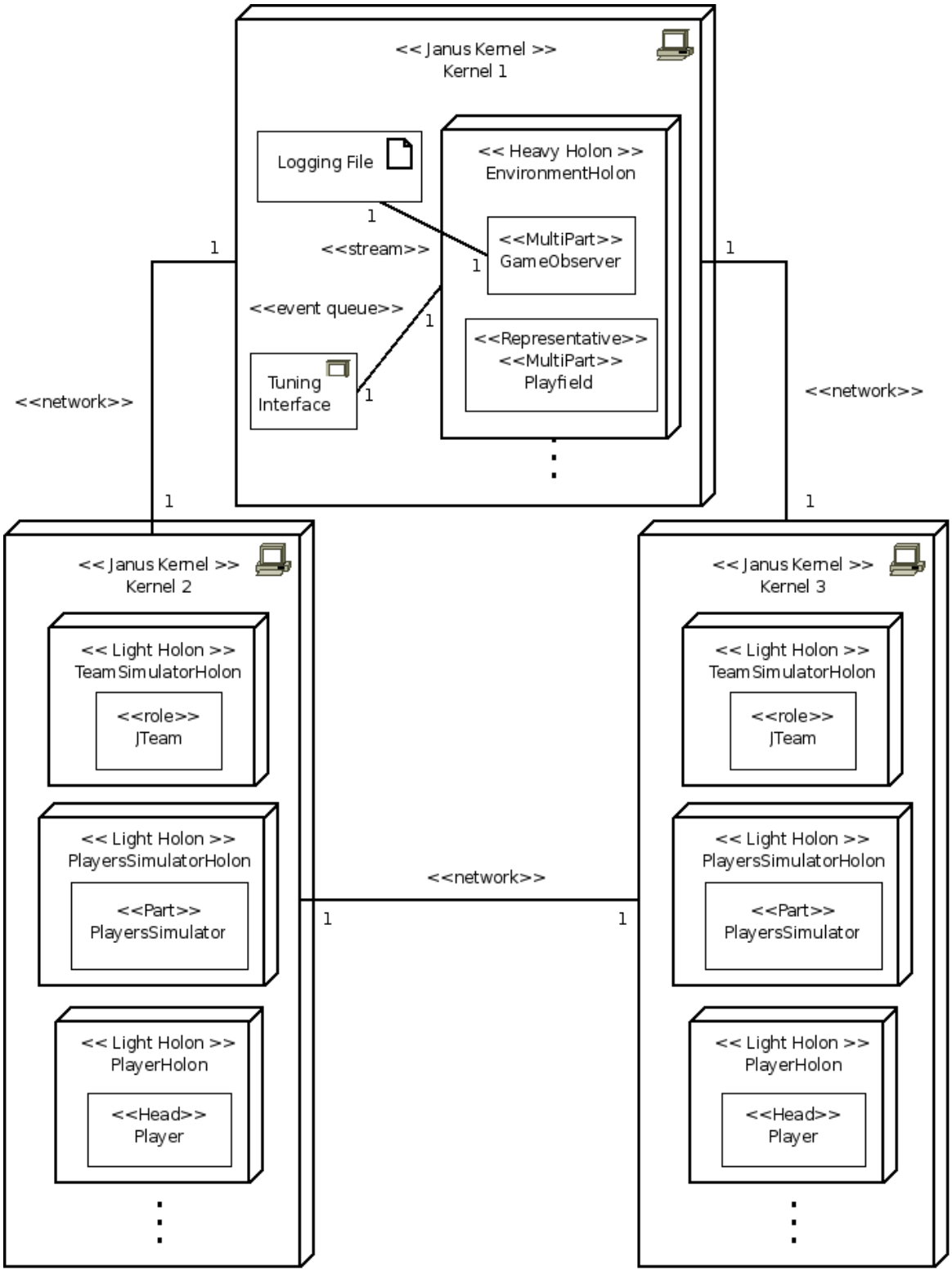


Figure 4.20: Diagramme de déploiement du simulateur de robots footballeurs

Modélisation distribuée de la relation environnement-système L'environnement est modélisé selon le point de vue des organisations qui le manipulent. L'utilisation des *boundary roles* permet de délimiter le périmètre de l'application à développer. Des capacités

spécifiques sont ensuite utilisées pour faire l'interface entre ces rôles et les ressources environnementales nécessaires à l'application. Ces capacités permettent de définir le comportement des *boundary roles* en faisant abstraction de la représentation spécifique des ressources environnementales. L'usage de ces capacités permet de modifier la représentation de l'environnement, sans avoir pour autant à modifier le comportement des rôles. Cette approche facilite la distribution des applications dans des environnements hétérogènes et distribués.

Permettre la modélisation du système à plusieurs niveaux d'abstraction Par définition, l'approche holonique permet de modéliser un système à différents niveaux d'abstraction. La décomposition organisationnelle hiérarchique d'un système peut être affinée jusqu'au niveau où le concepteur considère que la complexité des comportements est suffisamment faible pour une implantation immédiate.

Respecter les standards Le respect des standards actuels est également l'un des points clefs du processus ASPECS. Cette volonté s'intègre dans la perspective, à terme, de pouvoir transférer les technologies multi-agents dans le milieu industriel. Le respect des standards se traduit, d'une part dans la conception même du processus de développement, puisque ce dernier a été créé en conformité avec le modèle de processus [SPEM, 2007] fourni par l'OMG, et d'autre part dans le choix des langages et notations utilisés pour supporter le processus de développement : le standard actuel étant incarné par UML 2.0. Afin de pleinement satisfaire les besoins liés à l'approche orientée-agent, les notations UML ont été étendues en utilisant les mécanismes d'extension fournis par ce langage (et notamment les *"profiles"*).

CHAPITRE 5

JANUS : UNE PLATE-FORME D'IMPLANTATION ET DE DÉPLOIEMENT DE SYSTÈMES MULTI-AGENTS HOLONIQUES

LES SYSTÈMES MULTI-AGENTS HOLONIQUES sont basés sur la définition d'entités auto-similaires récursives. Si de nombreuses plates-formes SMA existent aujourd'hui, la majorité d'entre elles considèrent les agents comme des entités atomiques et ne permettent pas d'implanter correctement la notion de holon. Ce constat nous a conduit à définir une nouvelle plate-forme, nommée *Janus*, spécifiquement dédiée à l'implantation de SMA holoniques conçue avec une approche organisationnelle. Ce chapitre est dédié à la description de cette plate-forme.

Sommaire

5.1 Introduction

Le présent chapitre traite des dernières étapes du processus de développement, consacrées à l'implantation et au déploiement d'applications multi-agents. Il présente une plate-forme particulière, nommée *Janus*, spécifiquement dédiée à l'implantation de systèmes multi-agents holoniques. Le métamodèle de cette plate-forme s'intègre dans le troisième et dernier domaine du métamodèle CRIO : le domaine de la solution. Ce domaine correspond au modèle PSM[1] défini dans l'approche MDA, car il est dépendant d'une plate-forme d'implantation particulière. Afin de faciliter la transition entre la phase de conception et la phase d'implantation, le métamodèle de *Janus* définit un ensemble de concepts relativement proches de ceux introduits dans le domaine agent de CRIO.

Ce chapitre est organisé de la manière suivante : la section 5.2 résume les principaux besoins qui nous ont poussés à développer une nouvelle plate-forme. La section 5.3 présente le métamodèle à la base de *Janus*. Les sections 5.4 et 5.5 décrivent l'architecture et les différentes fonctionnalités de la plate-forme. La section 5.6 décrit une partie de l'implantation de l'exemple du simulateur de robots footballeurs, présenté au chapitre précédent, afin de mettre en valeur les spécificités de la plate-forme *Janus*.

5.2 Motivations

L'objectif poursuivi de la plate-forme *Janus* est, à terme, de faciliter le transfert de la technologie multi-agents et l'utilisation des approches organisationnelle et holonique dans le cadre de projets industriels. [Dastani and Gomez-Sanz, 2005] considèrent que les applications multi-agents ne pourront effectivement convaincre les industriels que si le fossé qui sépare, d'une part, l'analyse et la conception des SMA et, d'autre part, leur implantation est comblé.

Notre approche consiste à réduire l'écart entre le métamodèle de conception et celui de l'implantation, et à faciliter ainsi d'autant la transformation entre ces deux métamodèles. Réduire cet écart impose de disposer d'une plate-forme, dont le métamodèle offre une implantation aussi directe que possible des concepts utilisés lors de la conception de la solution. Pour aisément implanter les modèles conçus sur la base du métamodèle CRIO, les concepts de holon, d'organisation, de rôle et de capacité sont indispensables.

Les systèmes multi-agents holoniques introduisent des entités auto-similaires et récursives. Implanter des modèles holoniques nécessite une plate-forme capable de gérer la notion de hiérarchie de composition entre entités, mais la plupart des plates-formes SMA existantes considèrent les agents comme des entités atomiques. Il est donc difficile d'implanter la notion de holon, et de fournir une représentation opérationnelle de modèles combinant plusieurs niveaux d'abstraction, en utilisant de telles plates-formes.

[1] PSM : *"Platform Specific Model"*

Dans le métamodèle CRIO, les organisations sont considérées comme des modules complètement indépendants des agents et aisément réutilisables dans diverses applications. La clef de cette définition modulaire des organisations repose sur les concepts de rôle et de capacité. Implanter aisément les modèles basés sur CRIO nécessite donc une plate-forme dont le métamodèle considère le rôle comme une entité de premier ordre, indépendante de l'agent. Sur cet aspect, la plate-forme Madkit [Gutknecht and Ferber, 2000, Gutknecht et al., 2000, Gutknecht, 2001] et son extension MOCA [Amiguet, 2003, Amiguet et al., 2002] ont retenu notre attention, car elles gèrent, toutes deux le concept de rôle. Cependant, Madkit ne considère pas le rôle comme une entité de premier ordre. En effet, le comportement associé au rôle est directement implanté dans l'agent qui le joue. Les rôles sont donc fortement liés à l'architecture des agents. Cette approche nuit à la réutilisation et à la modularité des organisations. MOCA considère effectivement les rôles comme des entités de premier ordre, mais fixe des contraintes strictes concernant leur mise en œuvre. Par exemple, un agent ne peut jouer plusieurs fois le même rôle. Enfin, aucune de ces deux plates-formes ne gère la notion de capacité.

En d'autres termes, aucune plate-forme existante ne permet d'implanter aisément des systèmes multi-agents conçus avec une approche organisationnelle, une nouvelle plate-forme d'implantation a donc été créée : *Janus*. Elle se fonde sur la combinaison des approches organisationnelle, multi-agents et holonique. Le cœur de l'implantation de son modèle organisationnel a été inspiré par les approches adoptées dans Madkit et MOCA. *Janus* intègre également l'ensemble des concepts nécessaires à l'implantation des holons.

5.3 Métamodèle de *Janus* : domaine de la solution de CRIO

Cette section vient compléter la présentation du métamodèle CRIO réalisée au chapitre 3 en précisant son troisième et dernier domaine : le domaine de la solution. Les concepts introduits dans ce domaine sont partiellement décrits dans le diagramme UML de la figure 5.1. Le domaine de la solution représente une partie des concepts fournis par la plate-forme logicielle *Janus*. Le modèle UML complet du cœur de *Janus* est présenté dans l'annexe A (cf. page 217).

Le métamodèle de *Janus* est conçu pour faciliter la transition entre la conception et l'implantation logicielle d'un système. *Janus* fournit ainsi une implantation directe des cinq concepts à la base de la conception dans CRIO : organisation, groupe, rôle, holon et capacité.

L'organisation est considérée comme une classe (au sens du métamodèle objet) qui regroupe un ensemble de classes de rôles. Une organisation peut être instanciée sous forme de groupes, chaque groupe contenant un ensemble d'instances des différentes classes de rôles, associées à l'organisation qu'il implémente. Le nombre d'instances autorisées pour chaque rôle est spécifié dans l'organisation. Un rôle est local à un groupe et fournit les moyens de communiquer au sein de ce groupe. L'un des aspects les plus intéressants de *Janus* concerne

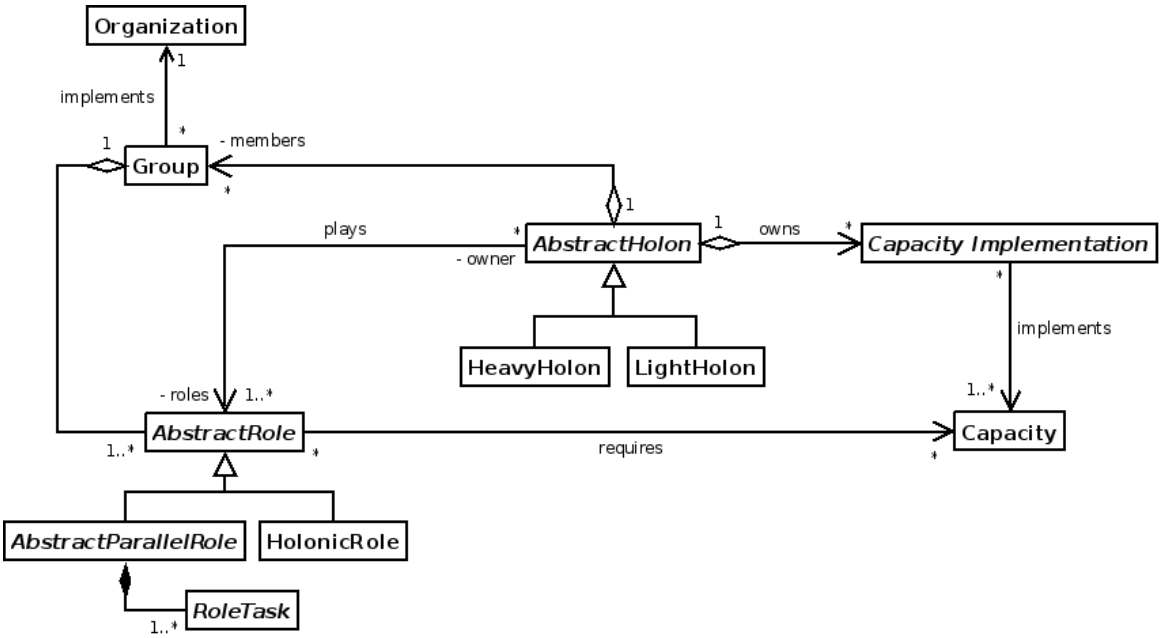


Figure 5.1: Diagramme UML simplifié de la plate-forme *Janus* et du domaine de la solution (PSM) du métamodèle CRIO

l'implantation du concept de rôle en tant qu'entité de première classe. Le rôle est considéré comme une classe à part entière, et les rôles sont implantés indépendamment des entités qui les jouent. Une telle implantation facilite la réutilisation des organisations dans d'autres solutions, mais autorise également une grande dynamique pour les rôles.

Dans *Janus*, un agent est représenté par un holon atomique (ou non composé). *Janus* définit deux principaux types de holon : *HeavyHolon* et *LightHolon*. Un *HeavyHolon* dispose de sa propre ressource d'exécution et peut donc fonctionner de manière indépendante. Le *LightHolon* est associé aux mécanismes d'exécution synchrone. Cette architecture s'intègre dans un modèle de synchronisation qui définit les différentes politiques d'exécution du système et des holons en charge de leur mise en œuvre.

Un holon peut jouer simultanément plusieurs rôles définis dans plusieurs groupes. Il peut accéder dynamiquement à de nouveaux rôles et libérer des rôles dont il n'a plus l'usage. Lorsqu'un holon accède à un rôle, il obtient une instance de la classe de ce rôle qu'il stocke dans son conteneur de rôles ; respectivement, lorsqu'il libère un rôle, l'instance correspondante est supprimée. La taille, en terme de code, d'un holon est toujours minimale, car il ne contient que le code des rôles qu'il joue à un instant donné. Pour accéder ou libérer un rôle, un holon doit satisfaire les conditions d'obtention ou de libération du rôle et du groupe correspondant (*obtainConditions* et *leaveConditions*). Ce mécanisme offre de nombreux avantages en terme de sécurité, puisqu'un holon ne pourra accéder au comportement d'un rôle, et donc obtenir le code exécutable correspondant, que s'il valide les conditions d'accès du rôle et celles de son groupe. Pour une même organisation, chacune de ses instances (ou groupes) peut disposer de droits d'accès et de libération spécifiques.

Les conditions d'accès et de libération des rôles sont en revanche définies au niveau de l'organisation.

La notion de capacité permet de représenter les compétences d'un holon. Tout holon dispose, dès sa création, d'un ensemble de compétences de bases, dont la possibilité de jouer des rôles (et donc de communiquer), d'obtenir des informations sur les organisations et les groupes existants au sein de la plate-forme, de créer des holons, et d'obtenir de nouvelles capacités. Tout comme dans le domaine agent du métamodèle CRIO, la capacité permet de faire l'interface entre le holon et les rôles qu'il joue. Le rôle requiert certaines capacités pour définir son comportement, lesquelles peuvent ensuite être invoquées dans l'une des tâches qui composent le comportement du rôle. L'ensemble des capacités requises par un rôle sont spécifiées dans les conditions d'obtention du rôle. Une capacité peut être réalisée de diverses manières, et chacune de ces réalisations concrètes est modélisée par la notion d'implantation de capacité (ou *"Capacity Implementation"*). Ce concept correspond à la représentation implantatoire de la notion de service au sein du domaine agent. Il ne porte pas le nom de service afin de le distinguer clairement du concept de service web que nous prévoyons d'intégrer dans les futures versions de *Janus*.

En plus de ces différents concepts, *Janus* fournit tout une gamme d'outils pour faciliter le travail du développeur. Les différentes fonctionnalités offertes par *Janus* seront décrites dans la section suivante.

5.4 Architecture et noyau de *Janus*

L'architecture complète de la plate-forme *Janus* est présentée dans la figure 5.2. *Janus* est développée en Java 1.5. Le cœur de la plate-forme est constitué par son noyau qui fournit l'implantation du modèle organisationnel et de la notion de holon. Le noyau a ensuite été étendu pour intégrer le module de simulation et les holons en charge du fonctionnement de la plate-forme et de son intégration avec les applications.

Les diverses fonctionnalités fournies par le noyau *Janus* sont décrites ci-dessous :

Gestion du modèle organisationnel Ce module gère les organisations et leurs instanciations sous forme de groupes. Il fournit également les mécanismes d'acquisition, d'instanciation et de libération des rôles, ainsi que les mécanismes d'acquisition et d'exécution des capacités. Cette gestion de l'information organisationnelle est gérée au plus bas niveau afin que tout holon, y compris ceux de la plate-forme, ait accès à ces fonctionnalités. Ce module est en relation avec le holon noyau en charge du maintien de cette information avec les autres noyaux distants.

Gestion des holons Le noyau fournit également l'ensemble des outils nécessaires à la gestion du cycle de vie des holons : adressage, lancement, arrêt, etc. Chaque classe de holon (*HeavyHolon* et *LightHolon*) propose, en natif, un ensemble de politiques d'exécution pour les rôles qu'il joue, ainsi que différentes politiques pour la gestion des

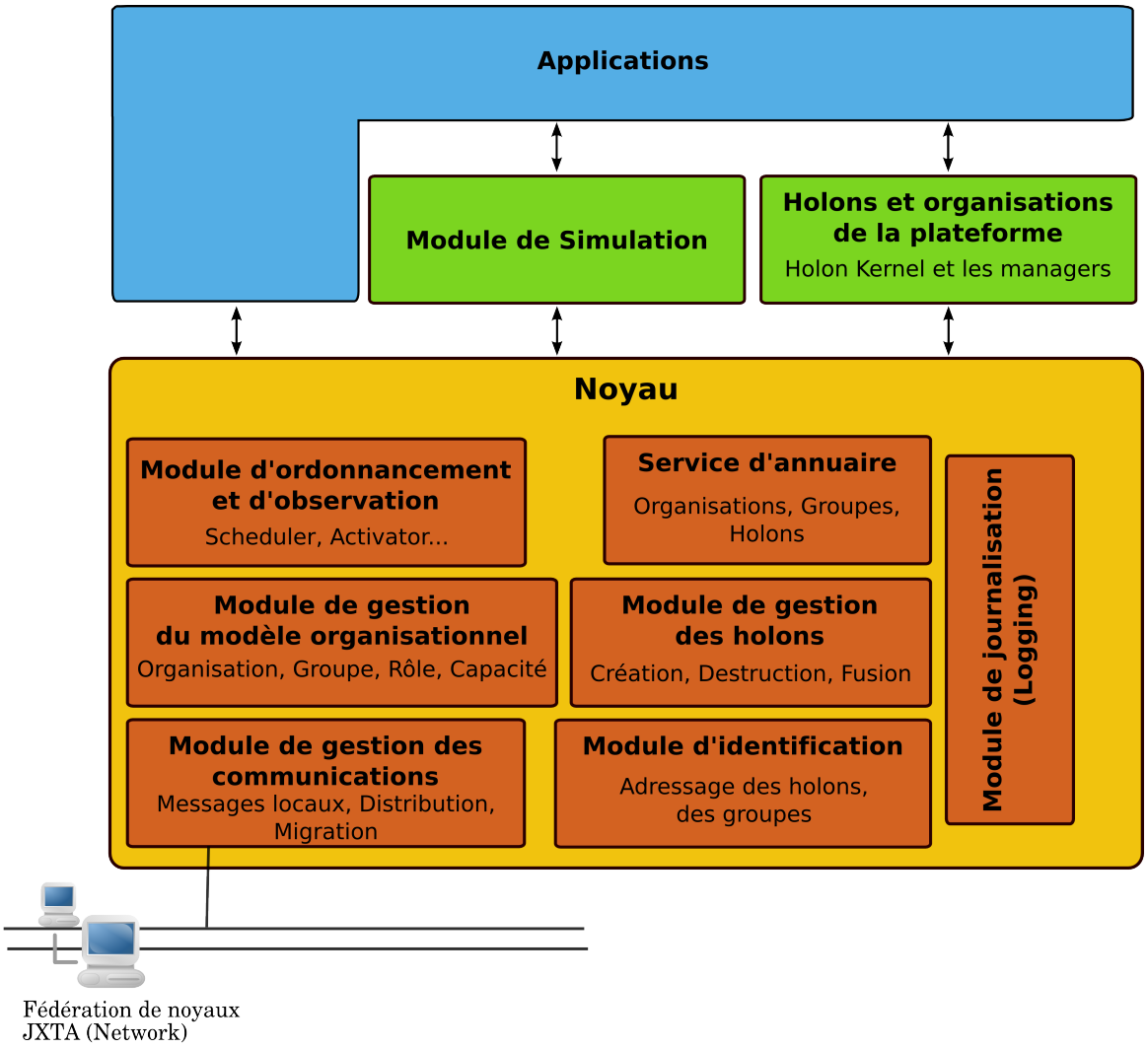


Figure 5.2: Structure générale de la plate-forme *Janus*

messages.

Gestion des communications Le noyau assure l'acheminement des messages en local et au sein de la fédération de noyaux lorsque l'application est distribuée. Le fait de considérer le rôle comme une classe à part entière affecte également cet aspect de la plate-forme. Le mécanisme de gestion des communications au sein de *Janus* sera détaillé dans la section 5.5.1.

Gestion de l'identification Le noyau fournit l'ensemble des mécanismes nécessaires pour l'attribution d'une adresse unique (GUID) à tous les éléments du modèle qui le nécessitent. Ainsi, les holons, les groupes, et les rôles disposent d'une adresse unique au sein d'une fédération de noyaux.

Service d'annuaires : Un service d'annuaire est également disponible au sein de la plate-forme. Il référence notamment l'ensemble des groupes, des holons et des organisations définis dans le noyau.

Gestion de l'ordonnancement des holons *Janus* fournit deux politiques de base pour l'ordonnancement des holons : un modèle pour l'exécution concurrente et un moteur d'exécution synchrone inspiré de celui de Madkit. Ce module fournit également une instrumentation à base de sonde permettant à un rôle d'observer un autre rôle. Contrairement à Madkit qui gère les droits d'observation au niveau de l'agent (agent qui implante ou non l'interface *ReferencableAgent*), *Janus* les gère au niveau du rôle. Elle permet ainsi une gestion plus fine des droits d'observation. Un holon peut donc autoriser l'observation de l'un de ses rôles et l'interdire pour les autres.

Gestion de la journalisation Toutes les applications basées sur *Janus* ont accès à un système de journalisation intégré à la plate-forme, ce qui facilite le processus de débogage (*"debugging"*). Le journal peut être directement affiché ou stocké dans un fichier. Ce système peut être changé et intégré à des systèmes existants. L'implantation actuelle de cette fonctionnalité est basée sur le système *log4j*[2] fourni par l'Apache Software Foundation.

Comme Madkit, *Janus* exploite son propre modèle dans la conception même de la plate-forme. Tous les services, excepté ceux assurés directement par le noyau, sont assurés par des agents ou des holons. Le noyau est ainsi associé à un holon *KernelHolon*, qui contient l'ensemble des organisations locales en charge de gérer la plate-forme, et qui représente son noyau dans la fédération distribuée sur le réseau. Une fédération est, en fait, une organisation en charge de gérer les différents échanges entre noyaux et de propager l'information de gestion du modèle organisationnel : création d'une nouvelle organisation ou d'un nouveau groupe, migration d'un holon, etc.

L'architecture de *Janus* respecte globalement l'architecture abstraite définie par le FIPA pour les plates-formes multi-agents[3]. Seule la partie ACL[4] n'est pas encore complètement implantée. Pour pallier cette lacune, il est prévu, à court terme, d'intégrer la partie correspondante du *"middleware"* JADE afin d'assurer une compatibilité complète de *Janus* avec le standard FIPA. À plus long terme *Janus* devra être associée à un AGL spécifique supportant le métamodèle CRIO, afin d'assurer une transition rapide entre les phases de conception et d'implantation.

5.5 Caractéristiques de *Janus*

Cette section est consacrée à la présentation des principales caractéristiques de la plate-forme *Janus*. Les aspects liés à l'implantation du concept de holon et aux mécanismes de communication entre rôles, seront notamment détaillés.

[2]Plus de détail à l'adresse suivante :`http://logging.apache.org/log4j/docs/index.html`

[3]FIPA Abstract Architecture :`http://fipa.org/specs/fipa00001/SC00001L.html`

[4]Agent-Communication-Language

5.5.1 Communication

Pour que deux holons puissent communiquer, ils doivent appartenir à un groupe commun et jouer un rôle dans ce groupe. Si un groupe est distribué parmi plusieurs noyaux, une instance de ce groupe existe dans chaque noyau ; et toutes les instances possèdent le même nom.

La communication dans *Janus* est basée sur les rôles. Les messages sont délivrés aux agents en ne connaissant que leurs rôles. Ce mode d'interaction permet l'implantation des communications entre un émetteur et plusieurs receveurs (*"one-to-many communication"*, [Gouaich et al., 2004]). L'adresse des agents receveurs est découverte dynamiquement en fonction des rôles qu'ils jouent. Lorsque plusieurs agents jouent le même rôle au sein d'un même groupe, un mode de communication basé sur l'identifiant des agents peut également être utilisé. Bien que l'identifiant explicite de l'agent soit connu, la communication passe toujours par un rôle donné dans un groupe donné. Ce type d'interaction permet l'implantation de la communication entre deux agents (*"one-to-one communication"*).

Chaque holon dispose ainsi d'une boîte à lettres personnelle pour l'émission et la réception de messages. Un holon peut jouer simultanément plusieurs rôles et en acquérir dynamiquement. Le rôle constitue le moyen d'interaction du holon dans le contexte d'interaction particulier représenté par le groupe. Les rôles sont à l'origine de toute interaction. Chaque rôle dispose donc de sa propre boîte à lettres (cf. figure 5.3) et la boîte à lettres d'un holon n'est que le regroupement de l'ensemble des boîtes à lettres des rôles qu'il joue. Un holon ne peut donc recevoir de messages que par l'intermédiaire de ses rôles.

5.5.2 Implantation de la notion de holon

Outre son modèle de rôle, l'une des contributions principales de *Janus* est incarnée par la gestion native du concept de holon. Deux aspects sont à distinguer dans l'implantation du concept de holon au sein de *Janus*. Le premier concerne l'approche d'intégration des concepts de rôle et de capacité pour implanter un holon atomique. Le second aspect concerne la manière d'implanter l'organisation holonique pour assurer la communication entre le super-holon et ses membres. Ces deux aspects seront détaillés dans les sous-sections suivantes.

Architecture d'un holon atomique Elle est décrite dans la figure 5.3. Un holon atomique est avant tout un conteneur de rôles et un conteneur de capacités. Le conteneur de rôles fournit les moyens nécessaires aux rôles d'un holon pour qu'ils puissent interagir dans le contexte d'interaction interne au holon. Ce mécanisme d'interaction, nommé *influence*, est implanté sur la base d'une communication par événement. Chaque rôle peut s'enregistrer pour informer son holon qu'il souhaite recevoir toutes les influences d'un type donné.

Le conteneur de capacités rassemble toutes les capacités possédées par le holon ainsi

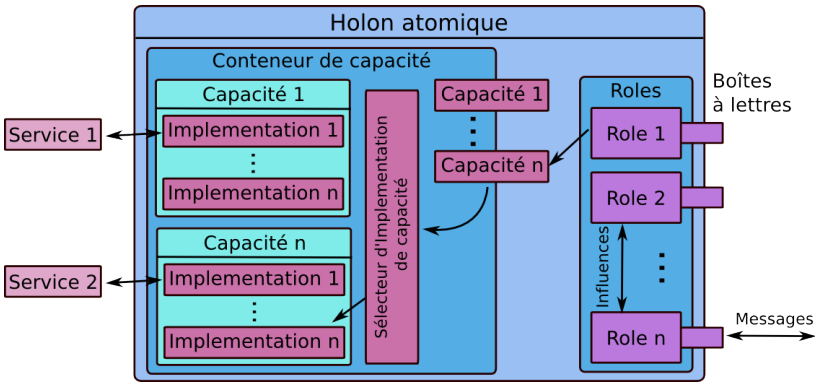


Figure 5.3: Architecture d'un holon atomique

que l'ensemble des implantations disponibles pour chacune d'entre elles. Il assure également l'exécution de celles-ci lorsqu'un rôle en requiert l'usage. Deux types d'exécutions de capacités sont disponibles dans *Janus* : synchrone ou asynchrone. Dans le premier mode, la capacité est directement exécutée lorsqu'elle est invoquée par le rôle. L'exécution de la capacité interrompt alors temporairement celle du rôle. Dans le second mode, l'exécution du rôle n'est pas interrompue, la capacité est exécutée après le rôle ou après l'ensemble des rôles du holon. Le rôle correspondant est ensuite informé du résultat. Ce mode est particulièrement intéressant lorsque la capacité est implantée par un service fourni par les membres d'un super-holon. Il permet notamment d'éviter le blocage de l'exécution du super-holon. Pendant que ses membres effectuent une tâche donnée, il peut ainsi poursuivre l'exécution de ses autres rôles.

Implantation de l'organisation holonique L'un des problèmes pour implanter l'organisation holonique vient du fait que, dans *Janus*, deux holons ne peuvent communiquer que s'ils appartiennent à un groupe commun. Cette règle implique qu'un super-holon doit partager au moins un groupe avec ses membres pour permettre le transfert d'informations entre deux niveaux d'abstraction adjacents. Plusieurs alternatives peuvent être envisagées pour implanter l'organisation holonique et la notion de super-holon. Notre étude se borne aux trois solutions suivantes :

1. La première alternative consiste à désigner l'un des membres pour qu'il représente le reste de la communauté au niveau supérieur. Cette approche est décrite dans la figure 5.4. Le holon H_3 joue le rôle *representative* et représente la communauté au niveau $n + 1$.

   Les membres qui jouent le rôle *representative* apparaissent comme les plus aptes à exercer ce rôle de représentation. Mais au sein d'un super-holon, plusieurs membres peuvent jouer ce rôle, ce qui implique qu'un des *representatives* en particulier soit désigné (ou élu). La communauté ne peut en effet être représentée au niveau supérieur que par un unique membre. Bien que cette approche apparaisse comme la plus

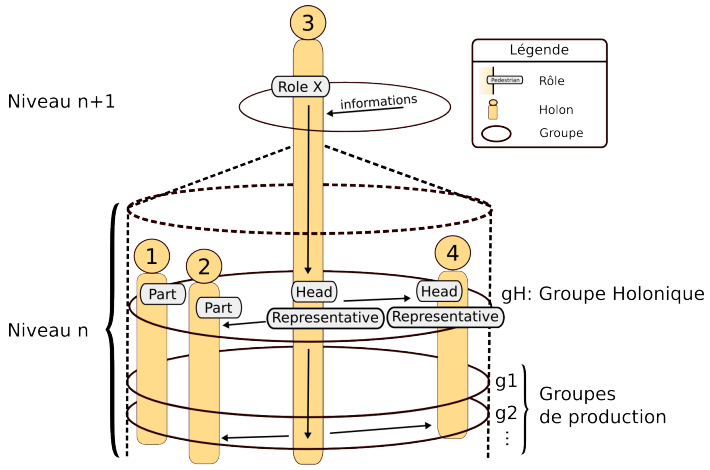


Figure 5.4: Une des alternatives d'implantation de l'organisation holonique

simple, elle introduit une distinction hiérarchique entre les membres pouvant être élus, et n'est donc pas complètement cohérente avec la définition fournie pour le rôle *representative* dans le métamodèle CRIO. De plus, cette première approche soulève deux autres problèmes. Tout d'abord, au niveau modèle, le super-holon est une entité à part entière distincte de ses membres. Le fait qu'un des membres représente la communauté au niveau supérieur signifie qu'il peut jouer des rôles définis par des organisations de niveau d'abstraction supérieur. Ainsi, le holon H_3 joue à la fois le rôle X au niveau $n+1$ et des rôles dans des groupes de production au niveau n.

Pour être cohérent, le super-holon doit être clairement distingué de ses membres. Jouer des rôles situés à des niveaux d'abstraction différents pourrait induire des problèmes d'interférences ou de conflits entre ces rôles.

De plus, le rôle *representative* n'est pas exclusif des autres rôles holoniques. Il peut donc être joué par un holon qui assure également le rôle *Multipart*. Un tel holon serait alors partagé entre deux holons de niveau n, et susceptible de les représenter au niveau $n+1$. Les problèmes de conflits et de confusion entre les niveaux d'abstraction seraient alors très importants. Cette approche est la plus simple, mais elle n'est pas optimale.

2. Pour distinguer clairement les niveaux d'abstraction et éviter les interférences éventuelles entre des rôles situés à des niveaux différents, une autre approche est envisageable. Cette dernière est décrite dans la figure 5.5. Dans cette approche, le super-holon est clairement distinct de ses membres. Une nouvelle entité est créée au niveau $n+1$. Un nouveau groupe $g0$ est introduit au niveau n pour effectuer l'interface entre les différents représentants des membres et le super-holon. En effet, un super-holon doit être clairement séparé de ses membres et ne peut communiquer qu'avec les représentants des membres. Un nouveau rôle est donc introduit : le rôle *Super*, pour permettre au super-holon de communiquer avec les représentants de ses membres

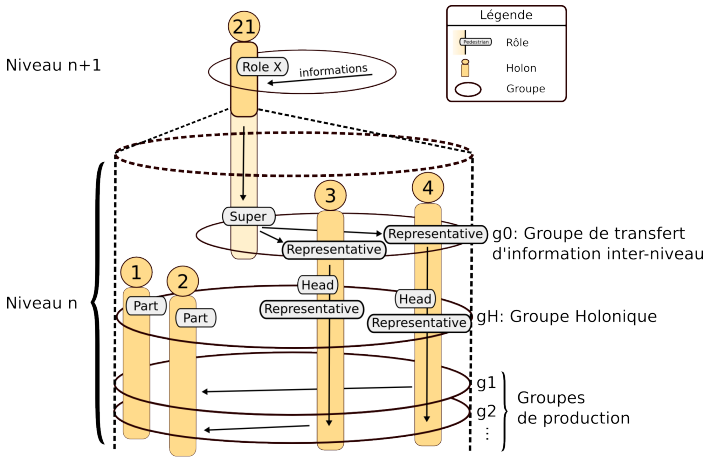


Figure 5.5: Modèle théorique de la structure d'un super-holon dans *Janus*

dans le groupe $g0$. Cette approche est la plus cohérente avec le domaine agent du métamodèle CRIO, mais exige la création d'un groupe supplémentaire dans tous les holons composés, et alourdit de manière notoire l'implantation des holons composés.

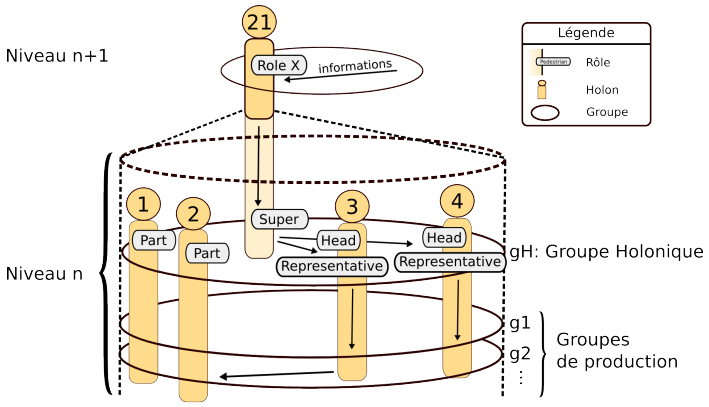


Figure 5.6: Structure d'un super-holon dans *Janus*

3. Afin de conserver les avantages de l'alternative précédente, tout en évitant le surcoût dû à la création d'un nouveau groupe, une troisième approche fondée sur un compromis entre les deux précédentes, a été adoptée. L'implantation de l'organisation holonique adoptée dans *Janus* est basée sur cette alternative. La figure 5.6 illustre cette implantation. Dans cette approche, le groupe $g0$ décrit précédemment est fusionné avec le groupe holonique. Le groupe holonique est en effet présent dans tous les holons composés. Un nouveau rôle est donc introduit dans l'organisation holonique, le rôle *Super*, pour représenter le niveau supérieur et ainsi permettre le transfert d'informations entre le super-holon et les représentants de ses membres. Cette dernière approche offre le meilleur compromis entre compatibilité avec le métamodèle CRIO et performance implantatoire.

5.6 Exemple : simulateur de robots footballeurs

La présente section termine la description de l'exemple d'un simulateur de robots footballeurs, débuté au chapitre précédent. Elle décrit une partie de son implantation sur la plate-forme *Janus*. Le programmeur dispose à ce stade, de la description complète de l'architecture des différents holons impliqués dans la modélisation du simulateur. La suite de cette section est consacrée à la description de l'implantation du rôle *PlayersSimulator* défini par l'organisation *TeamSimulation*, et de celle du holon *PlayersSimulatorHolon* qui doit jouer ce rôle. L'organisation *TeamSimulation*, comme les autres organisations de l'application, dispose de son propre paquet Java contenant les classes de ses rôles et sa propre classe. Chaque architecture de holon est également implantée dans une classe indépendante.

Dans *Janus*, l'accès et la libération d'un groupe ou d'un rôle sont soumis à certaines conditions. Les classes *Group* et *AbstractRole* héritent de la classe *ConditionnedObject* qui définit leurs conditions d'accès *ObtainConditions* et de libération *LeaveConditions*. Lorsqu'un holon demande l'accès à un rôle donné dans un groupe, via la méthode *requestRole(classe du rôle, Adresse du groupe)*, les conditions du groupe et du rôle correspondant sont automatiquement testées. Si le holon les valide toutes, il pourra effectivement jouer le rôle ; sinon, le statut de retour de la fonction *requestRole* lui permettra de déterminer les causes de cet échec.

D'après le modèle de conception du simulateur, le rôle *PlayersSimulator* requiert la capacité *PlayStrategy*. Elle est utilisée pour transférer l'information, liée à la nouvelle stratégie à adopter, vers l'organisation en charge de simuler le comportement des robots footballeurs. Le rôle *PlayersSimulator* hérite de la classe *AbstractParallelRole*, définie dans *Janus*, qui fournit une définition abstraite d'un rôle (spécialisation de la classe *AbstractRole*), dont le comportement est composé de plusieurs tâches aini que l'ensemble des outils nécessaires à la gestion de leur politique d'exécution. L'interface *RoleTask* fournit la description des méthodes génériques nécessaires à l'implantation d'une tâche composant le comportement d'un rôle. Le comportement du rôle *PlayersSimulator* est décomposé en deux *RoleTasks* : *ManageMessageTask* et *ManagePlayerTask*, conformément au statechart élaboré pendant l'activité de description comportementale des rôles (cf. figure 4.14, page 124). Le comportement associé à chaque tâche est ensuite défini dans sa méthode *execute()*.

Un fragment du code du rôle *PlayersSimulator* est fourni ci-après.

```
1 package janus.demos.robotsimulator.teamsimulation;
2 /* ... */
3 public class PlayersSimulator extends AbstractParallelRole {
4 // ... Détail des attributs du rôle ...
5    public PlayersSimulator() {
6       super();
7       //Définition des capacités requises par le rôle
8       List<Class<? extends Capacity>> requiredCapacities = new
```

```
            LinkedList<Class<? extends Capacity>>();
        requiredCapacities.add(PlayStrategyCapacity.class);
        HasAllRequiredCapacitiesCondition capCondi = new
            HasAllRequiredCapacitiesCondition(requiredCapacities);
        //Ajout des conditions d'obtention
        addObtainCondition(capCondi);
    }
/* ... */
    public void initTasks() {
        //Définition des tâches qui composent le comportement
        //du rôle
        this.addTask(new ManagePlayersTask());
        this.addTask(new ManageMessageTask());
    }
//Le coeur du comportement du rôle.
    public void behavior() {
        super.scheduleTasks();
    }
//Définition de la première tâche.
    private class ManagePlayersTask implements RoleTask {
        private int current = 1; // L'état courant de la tâche
        public ManagePlayersTask() { }
        //Comportement de la tâche
        public int execute() {
            switch (current){
            /*...*/
            // Cette méthode correspond à la traduction en java
            //du statechart correspondant au comportement de la tâche
            //et issu de la phase de conception
            default : return 0;
            }
        }
    }
/* ... */
}
```

La structure du code source de chaque holon est fournie par l'activité de description des architectures de holons et de conception des holarchies. Ces dernières définissent également l'ensemble des règles qui gèrent la dynamique des holons et qui permettent de déterminer la politique d'exécution la plus adaptée pour chaque holon, mais également pour les rôles qu'il joue.

À partir des résultats de ces activités, le programmeur choisit le type de holon le plus adapté (*HeavyHolon* ou *LightHolon*) et produit le code source correspondant, sur la base des primitives fournies par la plate-forme *Janus*. Chaque type de holon fournit l'ensemble des outils nécessaires à la gestion de la synchronisation et de l'ordonnancement des rôles

qu'il joue. Le cycle de vie de tout *LightHolon* est partitionné en trois phases principales : activation, exécution et terminaison. Traditionnellement, la phase d'activation est conçue pour permettre à chaque holon d'accéder aux rôles qu'il doit jouer et aux capacités dont il souhaite disposer. La phase d'exécution correspond au comportement proprement dit du holon et à l'exécution des rôles qu'il joue. Enfin, la phase de terminaison est consacrée à la libération des rôles assurés par le holon.

Le holon *PlayersSimulatorHolon* est exécuté avec une politique synchrone. Il hérite donc de la classe *LightHolon*. Les méthodes *activate()*, *live()* et *end()* correspondent aux trois phases de son cycle de vie. Un fragment du code source de la classe correspondante est fourni ci-dessous.

```
1  package janus.demos.robotsimulator.holon;
2  /* ... */
3  public class PlayersSimulatorHolon extends LightHolon {
4  // ... Attributs du holon ...
5     private GroupAddress TeamGA;
6
7  //1ere Étape du cycle de vie du holon : l'activation
8     public void activate() {
9        //Attribution des capacités au holon
10       addCapacity(PlayStrategyCapacity.class, new
             PlayStrategyCapacityImpl(this));
11
12       //Création d'un groupe ou
13       //Accès à l'adresse d'un groupe existant
14       //qui implémente l'organisation TeamSimulation
15       TeamGA = getOrCreateGroup(TeamSimulationOrganization.getInstance()
             );
16       println("About to request role "+TeamGA);
17
18       //Demande d'accès au rôle PlayersSimulator
19       // dans le groupe précédent
20       if(requestRole(PlayersSimulator.class,TeamGA)){
21          //Journalisation d'informations
22          getLogger().info("role PlayersSimulator assigned");
23       }
24    }
25 //2ieme Étape du cycle de vie du holon : comportement
26    public void live() {
27       //Exécution des rôles
28       while(true) {
29          for (Role role : getRoles())
30             role.behavior();
31       }
32    }
33 //3ieme Étape du cycle de vie du holon : Terminaison
```

```
34      public void end() {
35          //Demande de libération du rôle PlayersSimulator
36          if(leaveRole(PlayersSimulator.class,TeamGA)){
37              getLogger().info("role TeamSimulationOrganization:
                    PlayersSimulator disassigned");
38          } else {
39              getLogger().error("Pb during disassignation of
                    TeamSimulationOrganization:PlayersSimulator role");
40          }
41      }
42  }
```

5.7 Conclusion

Dans ce chapitre, nous avons brièvement introduit la plate-forme *Janus* destinée à l'implantation et au déploiement de systèmes multi-agents holoniques. *Janus* permet véritablement de considérer l'organisation comme un module Java à part entière. La gestion native du concept de capacité permet d'implanter un rôle, sans faire de pré-supposition sur l'architecture des holons qui le jouent, et favorise ainsi la réutilisation des organisations dans diverses applications. Il faut relativiser cette approche qui nécessite tout de même la définition d'un nombre assez important de classes, et cela même pour de petites applications (une classe par organisation, par rôle, par type de holon ou d'agent). *Janus* se destine donc avant tout à des applications de grande taille où la modularité est un élément essentiel. Ce problème du nombre de classes dans l'implantation confirme la nécessité d'intégrer *Janus* à un AGL tel que Janeiro, pour générer automatiquement des portions importantes de code, et ainsi simplifier l'intervention du programmeur. En outre *Janus*, fournissant une implantation directe des quatre concepts à la base de CRIO et d'ASPECS (capacité, rôle, organisation et holon), contribue à réduire l'écart entre la conception et l'implantation. Cette plate-forme s'intègre ainsi dans la perspective consistant à fournir une suite complète d'outils logiciels pour le développement d'applications complexes à vocation industrielle. L'annexe A (page 217) fournit un exemple plus complet que celui proposé dans ce chapitre, pour illustrer le fonctionnement de la plate-forme *Janus* sur l'exemple du problème des enchères.

TROISIÈME PARTIE

Des systèmes multi-agents holoniques à la simulation multi-niveaux

CHAPITRE 6

VERS UN MODÈLE DE SIMULATION MULTI-NIVEAUX

EXPLOITANT LE MÉTAMODÈLE CRIO et sa capacité à modéliser un système à différents niveaux d'abstraction, ce chapitre montre l'élaboration de modèles multi-niveaux pour des systèmes complexes. Les outils et modèles, présentés dans ce chapitre, sont le fruit d'une expérience menée dans le cadre du développement de la simulation multi-niveaux de piétons en environnement virtuel qui sera présentée au chapitre suivant. Sans prétendre être exhaustif, ce chapitre présente les conclusions de notre expérience que nous pensons pouvoir étendre et généraliser en vue d'enrichir la plate-forme *Janus* par un outil de simulation multi-niveaux.

Sommaire

6.1 Introduction

La simulation des systèmes complexes constitue actuellement un enjeu majeur dans de nombreux domaines tels que les neuro-sciences, la biologie, la physiologie, l'écologie, l'économie ou encore la sociologie. La simulation est une approche appropriée pour étudier des systèmes qui ne peuvent pas être directement observés ou mesurés [Conte and Gilbert, 1995]. Elle est devenue un outil scientifique à part entière permettant d'étudier l'évolution des phénomènes climatiques et météorologiques, l'évolution des marchés financiers ou encore évaluer l'impact d'une explosion nucléaire ou de la propagation d'une maladie contagieuse.

Un système complexe peut être défini comme un système contenant un grand nombre de composants en interaction dont le comportement global n'est pas dérivable de la somme des comportements de ses composants individuels (interactions non-linéaires). Typiquement, de tels systèmes exhibent une structure hiérarchique et des processus d'auto-organisation sous des contraintes sélectives. Ces phénomènes collectifs donnent souvent naissance à des propriétés émergentes [Drogoul and Ferber, 1994].

Pour pleinement comprendre la dynamique d'un système complexe, il est souvent nécessaire de combiner plusieurs points de vue sur le système à des niveaux d'abstraction différents. La simulation multi-niveaux peut apporter une solution à ce type de problème en permettant de moduler la complexité d'une simulation en fonction de contraintes spécifiques et notamment des ressources de calcul disponibles. L'une des approches pour moduler la complexité d'une simulation consiste notamment à adapter de manière dynamique le niveau de comportement des entités simulées (microscopique, macroscopique, etc.) tout en tentant de rester le plus précis et le plus proche possible du modèle originel.

Sur la base de la littérature et de notre propre expérience dans la simulation multi-niveaux (notamment à travers l'application présentée au chapitre suivant), nous tentons, dans ce chapitre, de présenter les éléments d'une approche de modélisation et d'implantation pouvant être enrichie pour aboutir au développement d'un simulateur multi-niveaux.

Concevoir une simulation nécessite la création d'au moins deux modèles : un premier pour le système étudié et un second pour le simulateur. La mise en place des mécanismes multi-niveaux impacte l'ensemble de ces modèles. Le modèle du système doit ainsi être enrichi afin d'intégrer les différents niveaux d'abstraction considérés sur le système. Le modèle du simulateur, quant à lui, doit être adapté afin d'incorporer les outils nécessaires à la synchronisation et à la transition entre les différents niveaux d'abstraction. La simulation multi-niveaux exige en outre que le système soit clairement distingué de son environnement afin d'autoriser une gestion indépendante et spécifique de leurs niveaux d'abstraction respectifs. Concevoir une simulation multi-niveaux impose donc de distinguer clairement des aspects qui sont d'ordinaire fusionnés au sein d'un même modèle de simulation traditionnelle. Une simulation multi-niveaux nécessite ainsi la création de quatre modèles multi-niveaux : un pour le système, un second pour son environnement et enfin un modèle d'exé-

cution pour chacun des modèles précédents. Ce chapitre détaille une manière de construire ces différents modèles et une méthode pour les fusionner afin d'obtenir un modèle global de la simulation.

Dans ce contexte, la section 6.2 présente un court état de l'art sur la simulation, et plus spécifiquement sur la simulation multi-niveaux et multi-agents. La section 6.3 fournit une description générale de l'approche proposée pour la conception de modèles multi-niveaux. La section 6.4 introduit un modèle de gestion de simulation multi-niveaux. Enfin, la section 6.6 présente un ensemble d'indicateurs qui visent à aider le concepteur de simulation pour déterminer la compatibilité de modèles de comportements qu'il fera cohabiter au sein d'une même simulation.

6.2 Contexte

Cette section s'attache à fournir un récapitulatif des définitions fondamentales au cœur de la simulation. Une présentation ainsi qu'un bref état de l'art sur la simulation multi-niveaux sont également proposés.

6.2.1 La simulation

[Shannon, 1977] définit la simulation comme *« le processus permettant de concevoir un modèle d'un système réel et de mener des expérimentations sur la base de ce modèle pour comprendre le comportement du système ou évaluer différentes stratégies pour son fonctionnement (dans les limites imposées par un critère ou un ensemble de critères) »*[1].

L'objectif de la simulation est de faciliter la compréhension de la dynamique d'un système et tenter d'en prédire l'évolution. Satisfaire cet objectif nécessite l'élaboration d'un modèle du système à étudier, son exécution sur un calculateur, et l'analyse des résultats de cette exécution [Fishwick, 1997].

Le modèle de simulation désigne généralement l'ensemble des mécanismes qui gèrent les changements d'état du système. Il correspond à l'ensemble des lois, conditions ou contraintes qui définissent le comportement du système, ainsi que la manière dont ses composantes sont agrégées. L'exécution, quant à elle, doit faire évoluer dans le temps le modèle du système [Coquillard and Hill, 1997]. Pour y parvenir, elle est généralement associée à un ensemble d'outils qui constituent le simulateur.

Dès lors que l'on considère un système et son modèle, la question de la cohérence de ce modèle se pose. Dans la même optique, dès lors que l'on considère un modèle et le simulateur en charge de l'exécuter, la cohérence de ce simulateur se doit d'être étudiée.

[1] la citation originelle est : *"the process of designing a model of a real system and conducting experiments with this model for the purpose either of understanding the behaviour of the system or of evaluating various strategies (within the limits imposed by a criterion or a set of criteria) for the operation of the system."*

Dans [Zeigler et al., 2000], on distingue ainsi deux relations fondamentales qui doivent être prises en compte lors de la conception d'une simulation :

Relation Système-Modèle : La **relation de modélisation** définit la notion de validité de l'expérience en soulevant la question de la correspondance entre le modèle (et le comportement qu'il génère) et le système source dans le contexte du cadre expérimental. Il s'agit donc de déterminer si le modèle du système est une simplification acceptable de celui-ci en fonction des critères qualitatifs choisis et des objectifs de l'expérimentation. Cette relation se rapporte directement à la cohérence du modèle de simulation.

Relation Modèle-Simulateur : La **relation de simulation** s'intéresse à l'aptitude du simulateur à reproduire fidèlement la dynamique du modèle initial, et à gérer correctement les phases de changement d'états du modèle (cohérence du simulateur).

6.2.2 La simulation multi-niveaux : définition et état de l'art

Dans le cadre de la simulation des systèmes complexes et plus spécifiquement celle de la dynamique de populations composées d'entités individuelles en interaction, deux grands types de simulation peuvent être distingués en fonction du niveau d'abstraction de leurs modèles [Davidsson, 2000, Hoogendoorn and Bovy, 2001] :

Les simulations macroscopiques (ou macro-simulation) : Elles sont généralement basées sur des modèles mathématiques (équations, ...), lesquels sont établis à partir des caractéristiques moyennées d'une population d'individus. Dans ce type de simulation, la population est considérée comme une structure qui peut être caractérisée par un certain nombre de variables. Le modèle de la simulation tente de simuler les variations et les changements de ces variables. La perspective adoptée par les simulations macroscopiques se situe au niveau du système. Elles cherchent à recréer la dynamique globale du système.

Les simulations microscopiques (ou micro-simulation) : Elles tentent explicitement de modéliser les comportements spécifiques des individus composant une population. La perspective adoptée se situe au niveau de l'individu ; la dynamique du système est issue des interactions entre les individus. La structure du système est considérée comme émergente de ces interactions. Les modèles de simulations microscopiques sont généralement basés sur les automates cellulaires ou les systèmes multi-agents.

Les forces et les faiblesses de ces deux approches ont été soulignées par [Van Dyke Parunak et al., 1998] qui considèrent que les approches microscopiques et plus particulièrement multi-agents sont plus appropriées aux systèmes caractérisés par un haut degré de localisation et de distribution. Van Dyke Parunak et al. considèrent également que les approches microscopiques sont régies par un processus de décision discret, alors que les approches

macroscopiques sont naturellement appliquées à des systèmes qui peuvent être modélisés de manière centralisée et dans lesquels la dynamique est régie par des lois physiques.

De manière plus générale, le volume et la précision des données requises pour l'initialisation des approches macroscopiques sont beaucoup plus réduits que pour les approches microscopiques. De plus les approches macroscopiques nécessitent moins de ressources de calcul. Dès lors, les approches macroscopiques apparaissent comme les plus aptes à simuler des systèmes de grande échelle. En revanche, elles s'avèrent inadéquates, voire incapables de modéliser l'impact d'événements microscopiques.

Les approches microscopiques quant à elles peuvent aisément modéliser l'impact de tels événements. Elles permettent plus facilement de comprendre le fonctionnement d'un système puisque la structure du système émerge des interactions entre les individus qui le composent. En revanche, elles nécessitent des données précises et nombreuses pour leur initialisation, ce qui les rend de fait difficilement applicables à des systèmes de grande échelle.

Ce constat, partagé également par [Magne et al., 2000] dans la simulation des réseaux de transport, amène naturellement vers des modèles qui tentent d'allier les avantages de ces deux approches. Deux perspectives peuvent alors être envisagées. (i) Tout d'abord, il est possible de disposer de simulations dont le niveau de détails se situe à une échelle intermédiaire entre ces deux extrémités, tel que le niveau mésoscopique où l'individu peut être modélisé, mais pas ses interactions. (ii) Il est également possible de combiner différents niveaux au sein de la même simulation. Les approches hybrides réunissent en général deux ou trois niveaux.

La simulation multi-niveaux (cf. définition 6.1), quant à elle, peut être considérée comme une généralisation de ces deux perspectives où toute une palette de niveaux de détails cohabitent au sein de la même simulation et où les transitions entre ces niveaux peuvent être assurées de manière dynamique.

Définition 6.1 Simulation multi-niveaux

La simulation multi-niveaux est un type particulier de simulation[a] où le modèle proposé du système intègre différents niveaux d'abstraction (au moins deux) et où les outils nécessaires à son exécution permettent de faire cohabiter ces différents niveaux d'abstraction au sein d'une même exécution et d'assurer la transition dynamique entre eux en fonction de contraintes définies (dépendantes du modèle ou du cadre expérimental).

[a]Dans cette définition, le terme *« simulation »* est entendu au sens de la définition fournie par [Fishwick, 1997].

La suite de cette section se propose de faire un bref tour d'horizon des approches existantes dans le domaine de la simulation multi-niveaux. La plupart des modèles multi-niveaux actuels disposent d'un nombre limité et fixe de niveaux de simulation. Deux niveaux sont très souvent considérés : microscopique et macroscopique, microscopique et mésoscopique, mésoscopique et macroscopique. Ces modèles sont généralement dépendants de l'application cible.

Dans de nombreuses approches, l'environnement est découpé sous forme de zones et le niveau de simulation de chacune d'elles est fixé a priori et cela pour l'ensemble de la simulation. Les transitions sont donc effectuées à des points de connexion déterminés. Dans d'autres approches, c'est le niveau de simulation des entités qui est fixé pour la simulation tout entière, déterminé a priori par le concepteur en fonction de son expérience et des résultats expérimentaux de précédentes simulations. Ce point de vue est d'ailleurs partagé par [Ghosh, 1986] qui propose l'un des premiers modèles de simulation multi-niveaux dynamique. Son domaine d'application est la simulation de composants électroniques. Il utilise un modèle basé sur la décomposition hiérarchique des composants, dans lequel le niveau de décomposition peut être dynamiquement changé. Mais le niveau n'est pas automatiquement déterminé en fonction des contraintes de la simulation ou des conditions d'applicabilité du niveau de simulation. C'est l'utilisateur du modèle qui choisit le niveau de décomposition.

Le domaine de la simulation en environnement virtuel fournit des modèles avec davantage de dynamique. Dans [Musse and Thalmann, 2001], la notion de niveau d'autonomie pour la simulation de personnages et de foules virtuels est proposée. Ils distinguent trois niveaux d'autonomie où le comportement de l'entité est soit fixe, soit *« autonome »* (proche des agents réactifs simples dans les SMA), soit directement contrôlé par l'utilisateur. Une autre contribution de ces travaux concerne la modélisation de la structure d'une foule d'individus qui est décomposée hiérarchiquement sous forme de groupes. L'objectif de ces travaux consiste à assurer le meilleur niveau de réalisme visuel pour la simulation, tout en maintenant des performances temps-réel[2]. En revanche, le niveau de précision des com-

[2]Ici *« temps réel »* se réfère à la simulation d'événements dont la vitesse de déroulement dans l'espace simulé est équivalente à celle perçue dans le monde réel.

portements des entités est relativement faible face à celui atteignable dans les simulations multi-agents. Cependant, le principe de l'approche constitue l'une de nos inspirations.

Dans la même optique, [Brogan and Hodgins, 2002] ont adapté la notion de niveau de détails (*"level of detail"*), initialement utilisée pour moduler la complexité de la représentation géométrique d'un environnement virtuel, aux comportements des entités évoluant dans de cet environnement. Toujours dans le domaine de la simulation en environnement virtuel, les travaux de [Chenney and Forsyth, 1997, Chenney et al., 2001] visent également à maintenir un niveau de réalisme maximal dans une simulation tout en conservant des performances optimales. Ils introduisent notamment la notion de *"proxy simulation"* où les entités hors du champ de vision de l'utilisateur sont simulées à un niveau faible de détails, sous forme événementielle. Les transitions dynamiques sont assurées de sorte à regénérer dans un état cohérent des entités qui vont apparaître dans le champ de vision de l'utilisateur.

Dans le domaine de la simulation de réseaux de transports, les travaux de [Magne et al., 2000] sur les approches hybrides *« micro-macro »* et ceux de [Burghout, 2004, Burghout et al., 2004, 2005] ou [Ziliaskopoulos et al., 2006] pour les hybrides *« méso-micro »* peuvent être soulignés. Dans la même optique, dans [Gloor et al., 2004], on propose un modèle hybride pour la simulation de piétons dans des environnements de grande taille.

6.2.3 Problématique : transitions entre niveaux d'abstraction

Dès lors que différents niveaux de détails sont considérés au sein d'une même simulation, la question de la transition entre ces niveaux devient cruciale. En effet, assurer le passage entre deux niveaux d'abstraction implique que les modèles utilisés par chacun d'entre eux soient compatibles. Cette compatibilité entre niveaux d'abstraction est l'un des problèmes majeurs des approches hybrides, en particulier pour les approches combinant les niveaux microscopique et macroscopique (approche *« micro-macro »*). En effet, dans de telles approches, les deux niveaux d'abstraction considérés sont très éloignés. Il devient alors difficile d'assurer la compatibilité entre leurs modèles. Cette dernière ne peut généralement être assurée que sous certaines conditions strictes : conditions stationnaires, équilibre, etc.

L'approche multi-niveaux tente d'intégrer des niveaux d'abstraction intermédiaires afin de réaliser une transition par palliers entre des niveaux d'échelle très différents, réduisant ainsi les risques d'incompatibilité entre les modèles. L'étude de ces transitions est généralement qualifiée dans la littérature par la notion de *« Pont »* (*"Bridge"*, *"Bridging the gap between"* micro–macro, micro–meso, etc). Ce problème de transition est directement lié à la théorie des systèmes complexes, et touche à des domaines très variés tels que la sociologie [Sawyer, 2003, Schillo et al., 2001, Schwenk, 2004, Troitzsch, 1996], la physique des matériaux [Ghosh, 1986, Li et al., 2005, Lundqvist et al., 2002], l'écologie [Balser et al., 2006, Wu, 1999], l'économie [Dopfer et al., 2004, Kemp and Van den Bergh, 2006, Sato and Namatame, 2001], la robotique [Pettinaro et al., 2003], la biologie [Uhrmacher and Priami, 2005, Uhrmacher et al., 2005], ou encore les systèmes de transport [Hoogendoorn

and Bovy, 2001].

Le problème du changement de niveaux introduit de nouvelles contraintes pour la conception d'une simulation. Ces contraintes viennent s'adjoindre aux relations de modélisation et de simulation précédemment citées.

Le niveau le plus précis auquel il est possible de simuler un système donné est le niveau microscopique[3] où le comportement des individus et leurs interactions sont modélisés et simulés. Le modèle d'un système dans une simulation ne constitue qu'une approximation du comportement du système selon un certain point de vue. Mais le point de vue microscopique peut être considéré comme celui qui rend le mieux compte de la dynamique du système. Il peut en ce sens être considéré comme le plus précis, le plus proche du comportement réel du système. Dès lors que sont considérés des niveaux d'abstraction plus élevés que le niveau microscopique (ex : mésoscopique ou macroscopique), certains aspects du système ne sont plus simulés, la précision du modèle diminue. Cette notion de précision de la simulation réfère à l'écart entre le modèle simulé et le fonctionnement réel du système. Dans la plupart des cas, les techniques macroscopiques exigent un nombre important d'individus pour être considérées comme représentatives et fournir une bonne approximation du comportement du système. Par exemple dans le cas de la simulation de piétons ou de trafic de véhicules, l'une des conditions pour utiliser les modèles macroscopiques est de disposer d'une densité forte d'individus (piétons ou véhicules) dans l'espace simulé. Cette contrainte est directement liée à la nature même des approches macroscopiques qui se basent sur des valeurs moyennées des caractéristiques d'une population. Si le nombre d'individus diminue, le modèle s'éloigne du comportement réel du système.

La complexité associée à un niveau d'abstraction dépend des modèles utilisés pour chacun d'eux et de l'application concernée. En revanche, quelque soit l'application cible, le problème de transition entre des niveaux d'abstraction différents demeure. La notion d'interaction est au cœur des problématiques de transition entre niveaux d'abstraction et soulève plus particulièrement deux questions ouvertes et cruciales : Comment évaluer l'impact de l'action d'un individu sur le système et sur les autres individus ? Comment évaluer l'impact du système, de la société ou du groupe sur les actions ou le comportement d'un individu ?

Ces différentes problématiques nous ont amenés à suivre une voie légèrement différente de celles classiquement proposées dans la littérature. En effet, l'approche proposée se positionne sur l'aspect commun à toute simulation : le simulateur. Elle consiste à fournir, au concepteur de simulations, les outils nécessaires pour réaliser des modèles multi-niveaux adaptés au système cible, et pour assurer une transition dynamique entre les niveaux du modèle. À chaque concepteur revient ensuite la charge d'étudier la compatibilité de modèles de comportements qu'il fera cohabiter au sein d'une même simulation. La section 6.6 présente un ensemble d'indicateurs qui visent à l'aider dans cette tâche.

[3]Certains auteurs traitent de la simulation sub-microscopique où les interactions entre les constituants d'une entité sont étudiés [Hoogendoorn and Bovy, 2001].

6.2.4 La simulation multi-agents

La simulation multi-agents se réfère aux modèles individu-centrés (cf. [Amblard, 2003]) et fournit un outil permettant de modéliser et simuler la dynamique de populations composées d'individus en interaction. Ce type de simulation assimile l'individu à un agent. La simulation multi-agents a été appliquée à un grand nombre de domaines tels que la robotique [Drogoul, 1993, Kitano et al., 1997], l'éthologie [Drogoul and Picault, 1999], l'écologie et la biologie (cf. annexe B.3 et B.5, sur CORMAS, GAEMAS et Mobidyc), ou les sciences sociales [Conte et al., 1998, Gilbert and Troitzsch, 2005].

Le modèle proposé par [Michel, 2004] pour la modélisation et la simulation de systèmes multi-agents constitue une des principales inspirations des travaux décrits dans ce chapitre. Il adopte une approche multi-vues et distingue quatre aspects fondamentaux dans un modèle de simulation multi-agents :

Comportements : cet aspect traite de la modélisation des processus de délibération des agents (leurs *« esprits »*).

Environnement : ce point de vue vise à définir les différents objets physiques du monde simulé (l'environnement situé et le *« corps »* des agents) ainsi que la dynamique endogène de l'environnement.

Ordonnancement : cet aspect concerne la modélisation de l'écoulement du temps et la définition de l'ordonnancement utilisé pour exécuter le comportement des agents.

Interaction : cette vue s'intéresse plus particulièrement à la modélisation du résultat des actions et des interactions à un instant donné.

La figure 6.1 illustre les relations entre ces quatre aspects fondamentaux d'une simulation multi-agents.

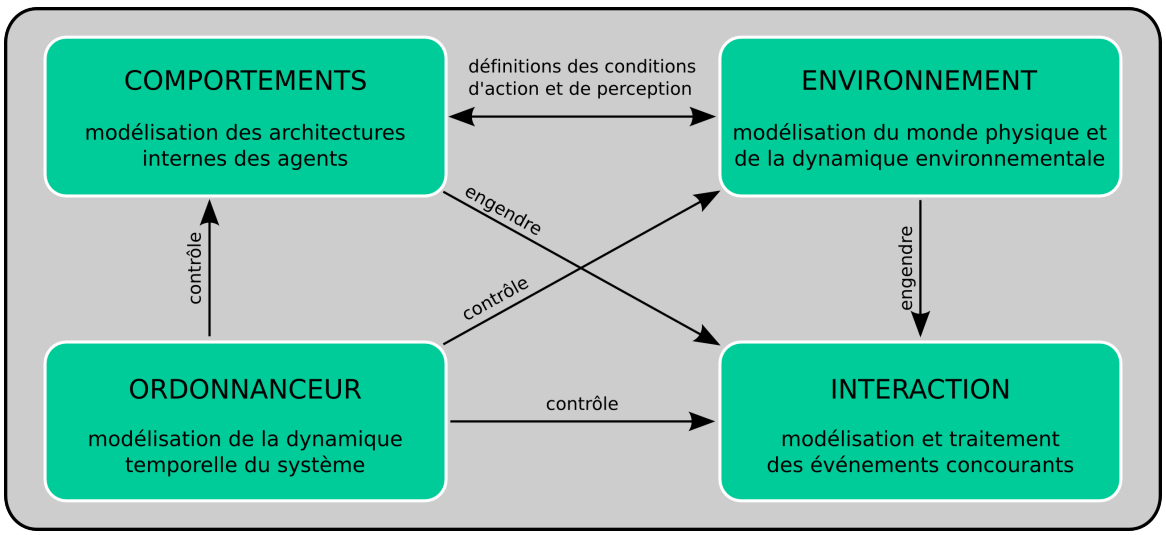


Figure 6.1: Les quatre aspects d'un modèle de simulation multi-agents selon [Michel, 2004]

La modélisation et l'implantation de chacun de ces aspects et de leurs relations sont autant de points délicats qui soulèvent les problématiques suivantes :

Respecter la contrainte de localité : un agent est une entité dont les perceptions et les actions n'ont qu'une portée locale. Deux approches principales existent pour respecter cette contrainte :

- L'approche discrète (centrée environnement) où la discrétisation de l'environnement sous forme de zones définit la granularité des perceptions et des actions des agents.
- L'approche continue (centrée agent) où la portée de chaque perception et de chaque action fait l'objet d'un traitement particulier qui est fonction de la nature et des caractéristiques de l'agent concerné [Weyns et al., 2004].

Ces deux approches peuvent également être combinées.

Respecter la contrainte d'intégrité environnementale : un agent ne doit pas être en mesure de modifier directement les variables d'état de l'environnement.

Biais de simulation et simultanéité des actions : Pour éviter d'introduire des biais dans la simulation, il est nécessaire de disposer d'un modèle de gestion de l'action des agents et du temps, qui permettent de modéliser la simultanéité de deux événements. Un modèle de simulation ne doit pas être lié à une implantation particulière [Zeigler et al., 2000]. L'ordre dans lequel les agents sont activés influe sur la dynamique du système et peut entraîner des biais de simulation.

De manière générale, pour prendre en considération ces différentes contraintes et concevoir des simulations multi-agents, nous adoptons le modèle Influence-Réaction [Ferber and Müller, 1996, Michel, 2001, 2006, Weyns and Holvoet, 2004].

L'action d'un agent au sein d'une simulation est généralement modélisée comme la transformation d'un état global [Ferber, 1995], c'est-à-dire la modification directe d'une des variables de l'environnement. Le cœur du modèle Influence-Réaction consiste à séparer l'action d'un agent de l'effet qu'elle produit. Dans ce modèle, un agent produit des influences sur son environnement et non des actions au sens vu précédemment. L'influence représente le désir d'un agent de voir modifier l'environnement d'une certaine façon. Le résultat effectif de cette tentative de modification de l'environnement par un agent ne peut être calculé sans connaître l'ensemble des influences produites au même instant. Ce modèle se base en fait sur la distinction claire entre deux dynamiques qui sont combinées dans un système multi-agents [Michel, 2006] : (i) La dynamique au niveau de l'agent qui produit des influences. (ii) La dynamique au niveau du système qui calcule la réaction de l'environnement compte tenu de l'ensemble des influences émises simultanément. Pour calculer cette réaction, les influences sont considérées en fonction des lois de l'univers [Ferber and Müller, 1996]. Dans la simulation de piétons présentée au chapitre 7, le modèle Influence-Réaction a été intégré conjointement avec l'approche organisationnelle pour la modélisation multi-niveaux d'un environnement urbain virtuel.

Adopter le modèle Influence-Réaction implique également de distinguer deux composantes de l'agent : son corps et son esprit (cf. [Michel, 2004, p. 117-119]). Cette distinction est nécessaire afin de clairement séparer le modèle d'environnement du modèle du système. En effet, l'esprit de l'agent est dépendant du système cible alors que son corps est intégré au modèle d'environnement. Cette distinction esprit/corps correspond à la partition entre les variables utilisées par le système décisionnel de l'agent sur lesquelles l'agent dispose d'un contrôle total, et les variables liées à la modélisation de sa partie physique contrôlées par l'environnement et sur lesquelles l'agent n'a aucun pouvoir. La figure 6.2 explicite cette distinction entre l'esprit et le corps d'un agent.

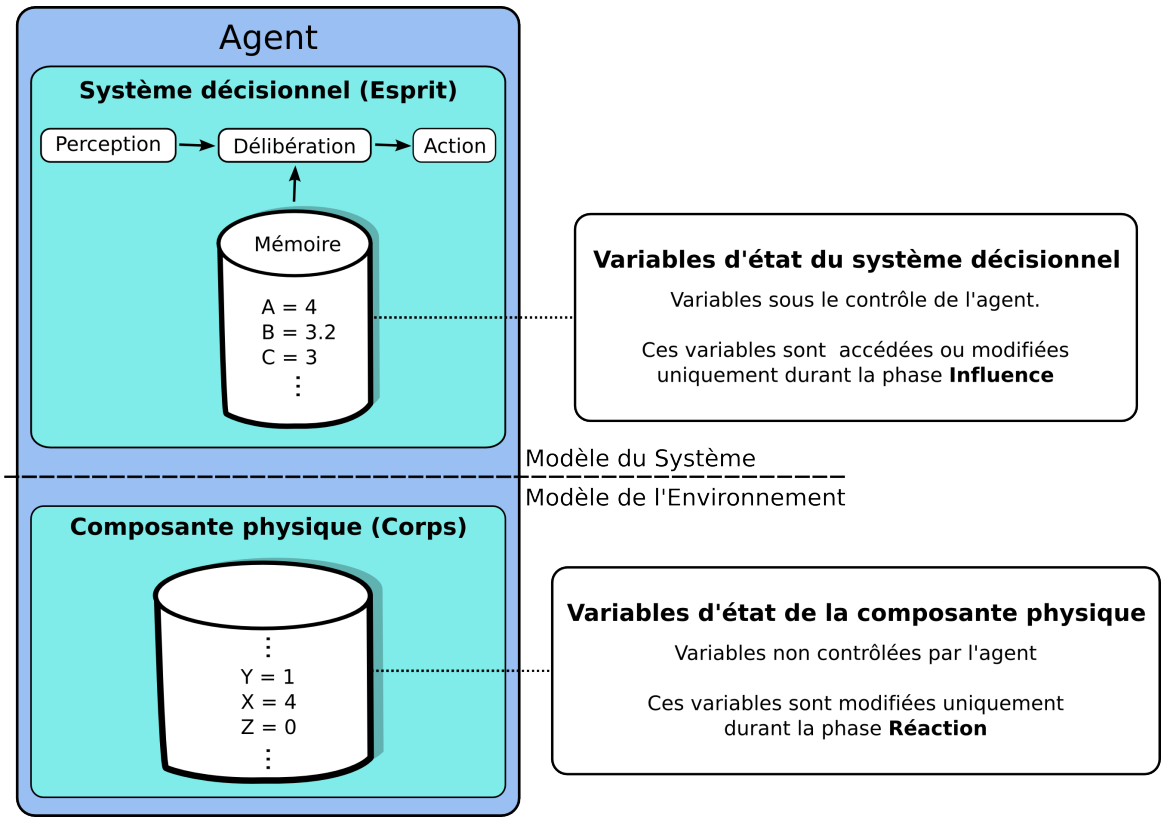


Figure 6.2: Distinction entre l'esprit et le corps d'un agent dans le contexte du principe Influence/Réaction [Michel, 2004]

Les simulations multi-agents conduisent souvent à l'émergence de groupes locaux d'entités [Servat et al., 1998], mais fournissent rarement les moyens de les manipuler. Exploiter pleinement ces simulations implique certes la création dynamique de tels groupes d'agents, mais également leur agentification de sorte à pouvoir gérer des comportements spécifiques à chaque niveau d'abstraction. Dès lors, les systèmes multi-agents hiérarchiques ou holoniques apparaissent comment une solution intéressante. [Van Aeken, 1999] introduit la notion de *systèmes multi-agents minimaux* pour étudier la dynamique organisationnelle des SMA. Cette approche est basée sur la création d'un agent composé pour chaque couple d'agents atomiques qui pourraient être fusionnés. Il construit ainsi une arborescence binaire

par regroupements successifs d'agents. Il fournit ensuite un ensemble de mesures pour caractériser la structure et les propriétés d'un *système multi-agents minimal* telles que la taille, l'équilibre, l'entropie et la notion de distance entre systèmes. L'inconvénient principal de cette approche est d'abord lié à sa structure uniquement binaire qui conduit à la création d'arbre de grande taille et, par conséquent, à de nombreux agents intermédiaires lorsque le système est composé de nombreux individus. Ce phénomène est généralement présent pour les systèmes complexes et induit de fait un surcoût notoire dans l'exécution des modèles. Toutefois, si cette approche demeure relativement abstraite et difficile à appliquer dans le cadre d'applications réelles, elle peut être considérée comme similaire à l'approche holonique adoptée dans ce manuscrit. L'un des avantages de la structure holonique est qu'elle n'impose pas cette partition binaire et permet de créer des groupes d'agents de tailles variables en fonction du problème et de sa dynamique. De manière générale, une holarchie est une structure beaucoup plus souple qu'une hiérarchie.

Après ce bref état de l'art sur la simulation, la section suivante sera consacrée à la description générale de l'approche proposée pour gérer des simulations multi-agents multi-niveaux.

6.3 Description générale de l'approche de simulation multi-niveaux

Pour satisfaire les différentes contraintes décrites précédemment (relation de modélisation, relation de simulation, intégrité environnementale, localité, etc), il est important de fournir aux développeurs les outils logiciels leur permettant d'implanter aisément différents types de simulateurs. Bien que la conception de simulateurs se situe à un niveau relativement technique, de tels outils doivent être conçus de manière réutilisable, et de sorte à faciliter le travail des développeurs et concepteurs de simulations : plus l'implantation d'un simulateur sera rendue explicite et intelligible moins les modèles comporteront de spécifications incomplètes ou ambiguës [Michel, 2004].

Cette section décrit tout d'abord les fondements de notre approche ainsi que les principaux modèles qui l'ont inspirée. Après une courte analyse des avantages et inconvénients de ces modèles, la section 6.3.2 présente l'approche que nous proposons pour la conception de modèles de simulation multi-niveaux.

6.3.1 Fondements de l'approche proposée

Madkit et Swarm constituent les principales inspirations des propositions effectuées dans ce chapitre. Dans la plupart des simulateurs, le fonctionnement d'une simulation est généralement basé sur une politique unique de gestion des agents : tous les agents sont soumis au même principe de synchronisation. Ce choix contribue à rendre l'analyse du système

difficile et limite les possibilités d'extension et de modification du système. En effet, ce type d'approche est extrêmement limitative surtout lorsqu'une simulation intègre plusieurs types d'agents nécessitant chacun une politique d'ordonnancement et de synchronisation particulière. Afin de surmonter cette limitation, l'approche adoptée par Madkit et Swarm consiste à diviser le problème de l'ordonnancement global d'une simulation en un ensemble de sous problèmes spécifiques. Une partition est ainsi effectuée au sein des agents à exécuter en fonction de la politique d'exécution ou de la méthode de synchronisation qu'ils requièrent. Chaque groupe d'agents est ensuite traité indépendamment.

Simulation multi-agents avec Madkit Les outils de conception de simulation intégrés à la plate-forme MadKit sont basés sur deux principales notions : l'ordonnancement et l'observation. Dans Madkit ces deux aspects sont gérés par un SMA. Afin de faciliter la réutilisation des modèles liés à l'ordonnancement et l'observation, Madkit propose un schéma d'organisation pour la gestion d'une simulation multi-agents [Michel, 2004]. Le but est de fournir aux développeurs une organisation prête à l'emploi préalablement peuplée par un ensemble d'agents. La définition de ces agents peut ensuite être étendue ou modifiée en fonction des besoins spécifiques d'une simulation donnée.

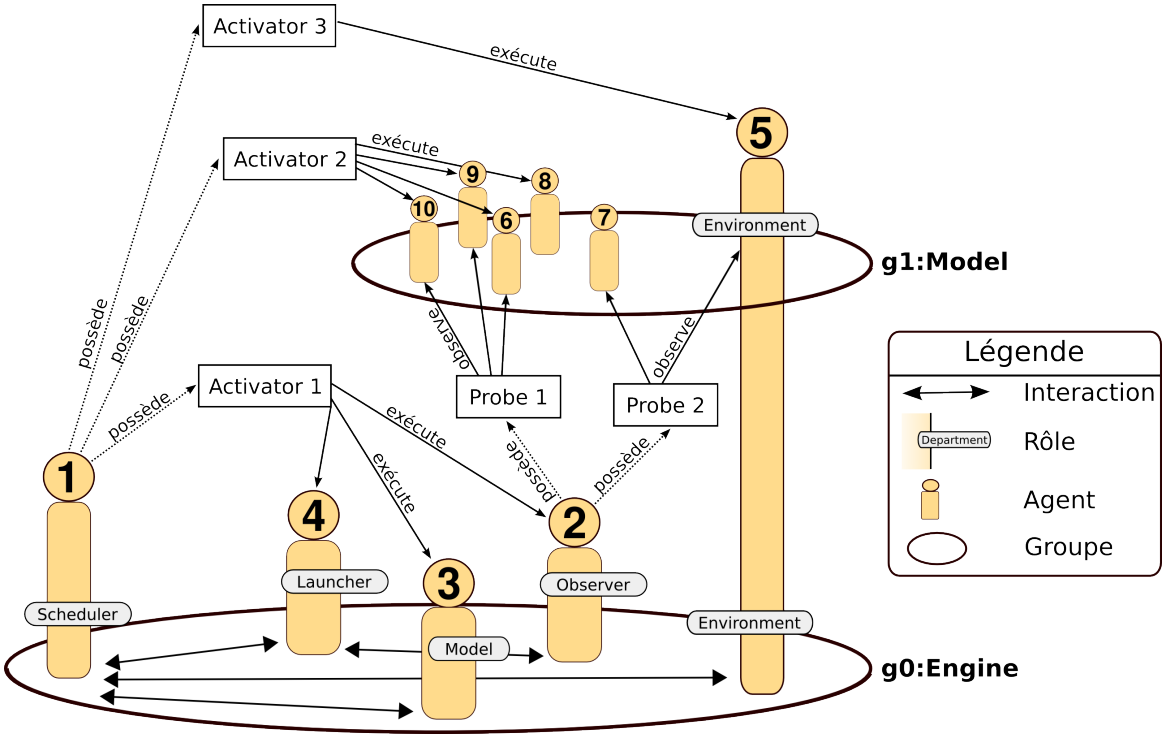


Figure 6.3: Le modèle du simulateur de Madkit (inspiré de [Michel, 2004, p. 100])

Le fonctionnement du simulateur de Madkit est basé sur une instance de ce schéma d'organisation nommée groupe *Engine*. La figure 6.3 illustre le fonctionnement de ce simulateur. Le groupe *Model* correspond à l'un des groupes du modèle du système cible. Ce dernier groupe est composé des différents agents du modèle et de l'agent environnement. Le

groupe *Engine* contient l'ensemble des agents en charge du fonctionnement du simulateur et définit les cinq rôles suivants :

- l'ordonnanceur (*"Scheduler"*) contrôle l'exécution des agents du simulateur et du modèle du système. Il gère des politiques d'exécutions et, pour ce faire, il est associé à un ensemble d'activateurs. Chaque *activateur* fournit à l'ordonnanceur le moyen d'identifier un ensemble d'agents étant donné un groupe et un rôle donné, et permet d'exécuter le comportement associé au rôle. Pour concevoir un simulateur particulier, le principe réside dans la spécialisation et l'ajout de nouveaux types d'activateurs.
- l'observateur (*"Observer"* ou *"Watcher"*) désigne les agents en charge d'observer ou de surveiller les agents du modèle (du système cible). Les observateurs n'interviennent pas dans la dynamique du système, mais permettent de collecter certains résultats de la simulation. Ils fonctionnent de manière analogue à l'ordonnanceur et gèrent un ensemble de sondes (*"Probe"*) leur permettant d'observer d'autres agents.
- l'environnement est chargé de récupérer, d'intégrer et d'initialiser les agents qui participent au processus de simulation.
- le lanceur (*"Launcher"*) est responsable du chargement de la simulation et du lancement des différents agents de la simulation, ex : scheduler et model.
- le modèle (*"Model"*) stocke les paramètres initiaux de la simulation et les caractéristiques du modèle.

Avantages et inconvénients Le schéma d'organisation nommée groupe *Engine* ne constitue que l'approche de modélisation adoptée dans Madkit, l'implantation correspondante en revanche est essentiellement basée sur les architectures de deux agents : *Scheduler* et *Watcher*. Par le fait que Madkit ne distingue pas clairement l'agent de ses rôles, et que la notion d'organisation n'existe pas en tant que telle, l'implantation correspondante ne permet pas de conserver l'ensemble des avantages du *"pattern"* organisationnel précédent.

L'approche, que nous proposons, repose globalement sur les mêmes principes que ceux proposés dans Madkit. Les principes d'ordonnancement hiérarchique et organisationnel sont conservés, mais ils sont évidemment étendus afin d'intégrer les mécanismes nécessaires à la gestion de simulations multi-niveaux et à la création de modèles multi-niveaux. Le schéma d'organisation précédent sera notamment étendu, et de nouveaux rôles seront introduits (cf section 6.4).

Si l'approche de conception de simulation, décrite dans ce chapitre, est proche de celle utilisée dans Madkit, l'implantation, quant à elle, diffère totalement, car elle est basée sur les principes de la plate-forme *Janus* et sur la distinction claire entre un agent et ses rôles. Cette distinction permet notamment aux rôles impliqués dans la gestion de la simulation d'être joués soit par des agents dédiés à cet effet soit par des agents intervenants dans d'autres aspects de la simulation tels que le modèle de l'environnement ou le modèle du système cible.

La section suivante introduit brièvement l'approche que nous proposons pour la conception de simulations multi-niveaux.

6.3.2 Description du modèle de simulation multi-niveaux

Pour pleinement comprendre la dynamique d'un système complexe, il est nécessaire de combiner plusieurs points de vue sur le système à des niveaux d'abstraction différents. La simulation multi-niveaux est une solution possible à ce type de problème en permettant de moduler la complexité d'une simulation en fonction de contraintes spécifiques et notamment des ressources de calcul disponibles. L'approche proposée pour moduler la complexité d'une simulation consiste à adapter de manière dynamique le niveau du comportement des entités simulées et le niveau de précision de la réaction environnementale.

Afin de clairement séparer les modèles de leur exécution, et le système de son environnement, une simulation multi-niveaux nécessite la création de quatre modèles multi-niveaux : le premier pour l'environnement du système, le second pour le système cible et enfin les modèles d'exécution associés aux deux précédents. Le fait que l'environnement soit clairement distingué du système permet de moduler sa complexité indépendamment de celle du système. Du point de vue de la gestion du multi-niveaux, les modèles du système et de son environnement sont donc indépendants. Les mécanismes d'exécution multi-niveaux peuvent être activés ou désactivés pour l'un des deux, de manière transparente pour l'autre.

Chacun de ces modèles (système et environnement) est représenté par une hiérarchie organisationnelle. Ces hiérarchies seront ensuite instanciées et associés à leur modèle d'exécution respectif de sorte à créer au moins deux holarchies : les holarchies d'exécution du système et de l'environnement. L'exploitation des propriétés de ces holarchies permet de moduler dynamiquement la complexité d'une simulation donnée. La holarchie est utilisée pour organiser hiérarchiquement les composantes du système et de l'environnement ainsi que les différents niveaux d'abstraction qui les composent. Chaque niveau de la holarchie correspond à un niveau d'abstraction du modèle. Le niveau exécuté dans la holarchie pour une composante donnée du système détermine son niveau de simulation (microscopique, mésoscopique, macroscopique, etc).

La figure 6.4 décrit les différents modèles impliqués dans l'approche proposée et les relations qui les lient. Cette figure peut être considérée comme une version étendue et adaptée au multi-niveaux de la vision proposée par [Michel, 2004] décrite précédemment dans la figure 6.1.

Les sections suivantes présentent brièvement l'approche proposée pour la conception de chacun de ces types modèles (système, environnement, exécution).

Conception de modèles multi-niveaux pour un système donné La conception de modèles multi-niveaux est basée sur la combinaison des approches holonique et organisationnelle (cf. chapitre 3 et 4). L'approche holonique permet en effet de modéliser aisément

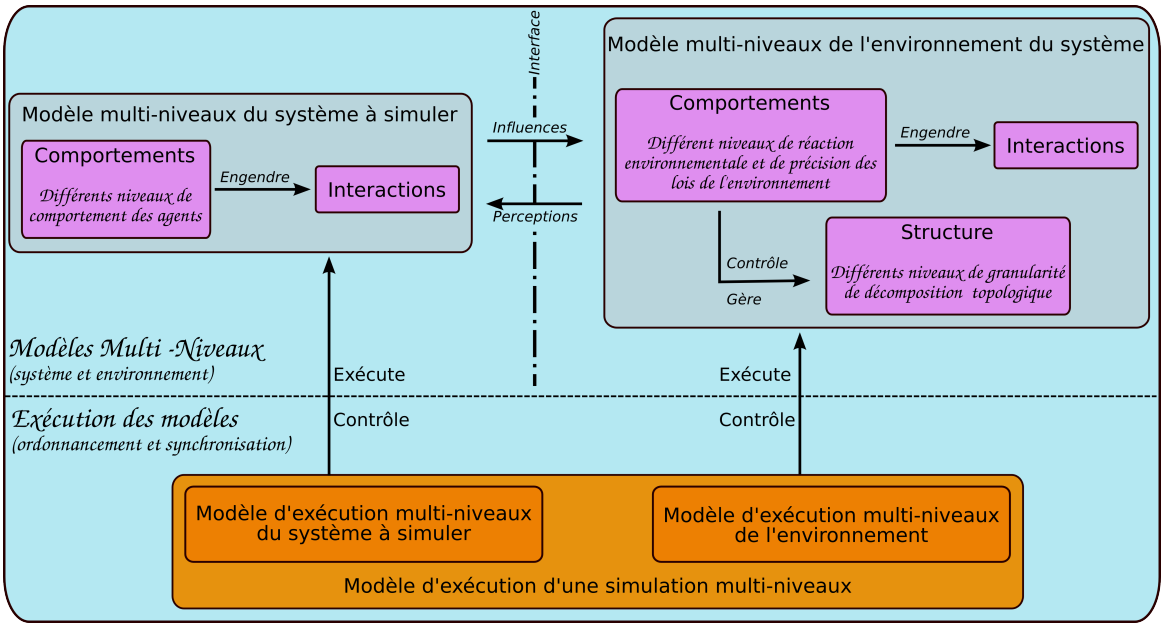


Figure 6.4: Les différents aspects d'une simulation multi-niveaux

un système à différents niveaux d'abstraction et favorise une intégration aisée des aspects structurel et comportemental. La notion de niveau d'abstraction dans les modèles est liée à la manière de décomposer structurellement et fonctionnellement un système en sous-systèmes. Un modèle multi-niveaux doit fournir une description de la décomposition de la structure du système et des comportements associés à chaque niveau. Il doit relier le point de vue holistique où le comportement global du système est étudié et le point de vue individualiste où le système est considéré comme une population d'individus en interaction.

La première étape de conception d'un modèle multi-niveaux consiste à identifier le ou les types de comportements qui doivent être simulés à plusieurs niveaux d'abstraction. Pour chacun de ces types de comportements, une hiérarchie comportementale est créée. Le niveau le plus bas de cette hiérarchie correspond aux comportements les plus précis (ex : ceux des individus) et le niveau le plus haut au comportement global du système. Gravir cette hiérarchie signifie que les caractéristiques propres à chaque composant du système sont progressivement agrégées, et que la diversité et la complexité de leurs comportements diminuent. En terme structurel, les composants sont graduellement agrégés en groupes, qui sont à leur tour regroupés en groupes de plus grandes tailles, et ainsi de suite jusqu'à obtenir un seul groupe ou composant correspondant au système dans sa globalité. Chacun de ces groupes est ensuite associé à un holon en charge de simuler son comportement.

Un modèle multi-niveaux est constitué par un ensemble de hiérarchies de comportements. Chacune de ces hiérarchies est ensuite associée à un modèle d'exécution particulier pour donner naissance à une holarchie d'exécution. Cette holarchie devra assurer la synchronisation entre les agents d'un niveau d'abstraction donné, mais également assurer la transition entre les différents niveaux d'abstraction en fonction des contraintes définies (ex :

volume de ressources de calcul disponible).

La figure 6.5 présente un exemple d'une holarchie d'exécution dans le cadre d'une simulation de piétons. Le niveau le plus bas de cette holarchie correspond au niveau microscopique où chaque piéton est associé à un holon. Les piétons sont ensuite regroupés, et chaque groupe est associé à un super-holon dont les membres sont des piétons individuels. Ce processus de composition se poursuit jusqu'à l'obtention d'une holarchie complète avec à son sommet un seul et unique super-holon. Ce dernier représente la population de piétons dans sa globalité. À chaque niveau de cette holarchie, les holons disposent d'un comportement d'un niveau d'abstraction différent.

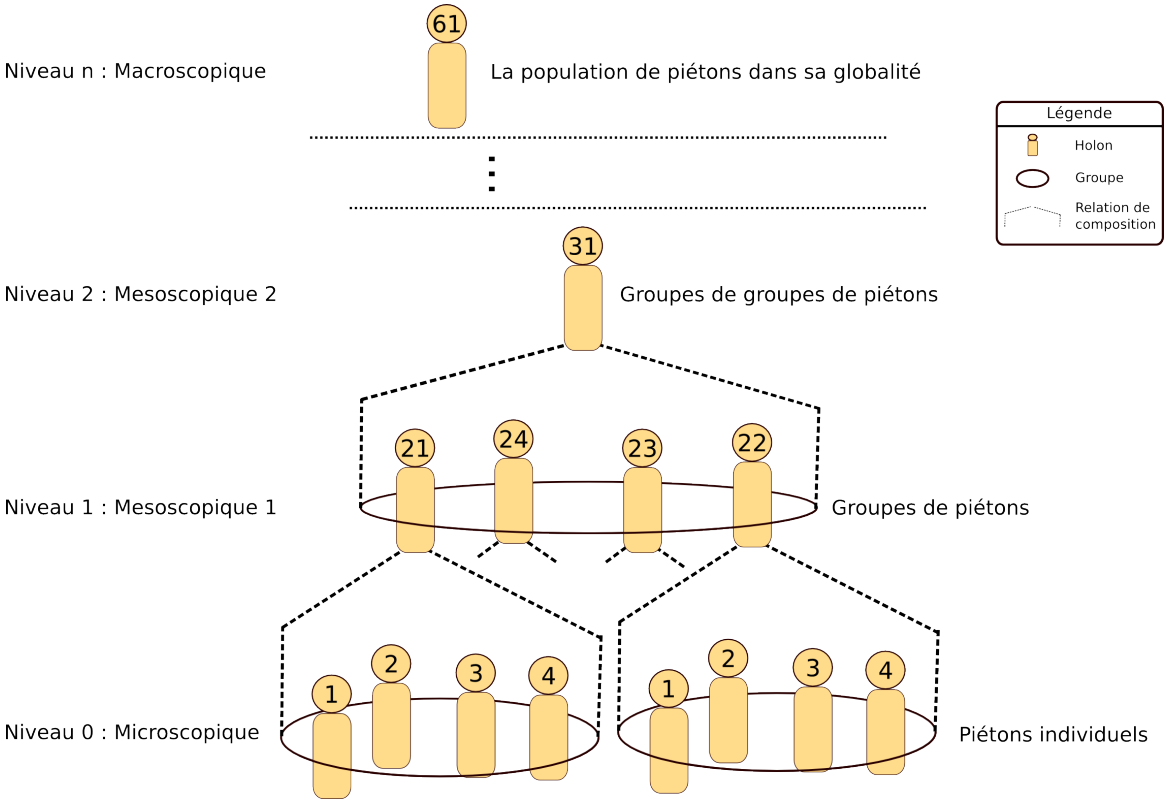


Figure 6.5: Exemple d'une holarchie d'exécution dans une simulation de piétons

Conception de modèle multi-niveaux pour l'environnement Dans le cadre de la simulation multi-agents, un modèle multi-agents multi-niveaux doit certes intégrer différents niveaux d'abstraction pour les comportements et les interactions des agents, mais également pour leur environnement. L'environnement est généralement considéré comme une entité monolithique dans les simulations multi-agents [Weyns et al., 2005, 2006]. Or cette vision ne se prête guère à la conception de modèles multi-niveaux. L'approche proposée dans ce manuscrit considère l'environnement d'une simulation comme un système multi-agents disposant de missions spécifiques. L'environnement est décomposé structurellement sous la forme d'une hiérarchie, et les comportements associés à chaque niveau sont ensuite

spécifiés.

La section 7.3 du chapitre suivant illustre la production d'un modèle multi-niveaux pour un environnement urbain vituel.

Conception d'un modèle d'exécution multi-niveaux Un simulateur doit être indépendant des modèles (agent, environnement, interaction et gestion du temps) qu'il exécute. À l'instar de Madkit, *Janus* considère que l'exécution d'une simulation multi-agents doit, au minimum, intégrer les outils nécessaires à la gestion de différentes politiques d'exécution et à l'observation du système en cours d'exécution. La base de ces outils est incarnée par une version étendue de l'organisation *Engine* de Madkit décrite dans la section précédente. Cette organisation permet d'assurer la gestion d'une simulation multi-niveaux (avec autant de niveaux que nécessaire) et d'assurer un changement dynamique entre ces niveaux.

La dynamique dans un modèle multi-niveaux réfère à la dynamique temporelle, mais également spatiale. Le simulateur doit être en mesure de changer le niveau d'abstraction d'une entité ou d'un groupe d'entités au cours du temps (dynamique temporelle). Mais à un instant donné, il doit être également capable de faire cohabiter au sein de la simulation des composantes du système simulées à des niveaux d'abstraction différents (dynamique spatiale). Un système dans sa globalité est en effet rarement simulé à un seul et unique niveau à un instant donné. Dans l'exemple présenté dans la figure 6.5, cette dynamique spatiale réfère au fait qu'au même instant dans la simulation, peuvent cohabiter un piéton individuel simulé microscopiquement par un holon atomique, et un groupe de piétons simulé mésoscopiquement par un super-holon. Le simulateur multi-niveaux en fonction des contraintes de la simulation (ex : ressource de calcul disponible) doit déterminer le niveau d'abstraction optimal pour chaque composante de la simulation.

La section suivante est consacrée à la présentation d'un modèle pour la mise en œuvre et l'exécution de modèles multi-niveaux.

6.4 Mise en œuvre et exécution d'un modèle multi-niveaux

Les outils proposés pour la simulation sont basés sur les notions d'ordonnancement et d'observation. Chacun de ces deux aspects est géré par une organisation spécifique. Cette approche permet de distinguer clairement la manière d'exécuter une simulation, de la manière de collecter ses résultats. De plus la problématique du multi-niveaux n'altère que très peu les mécanismes d'observation alors qu'elle impacte largement les aspects liés à l'ordonnancement et à la synchronisation. Seul l'aspect lié à l'ordonnancement sera décrit dans ce document. L'observation dans *Janus* hérite globalement des mêmes principes que ceux proposés dans Madkit. La seule différence majeure tient à leur implantation intégrant pleinement l'approche organisationnelle et distinguant clairement l'agent de ses rôles.

L'organisation en charge de la gestion de l'ordonnancement multi-niveaux est décrite

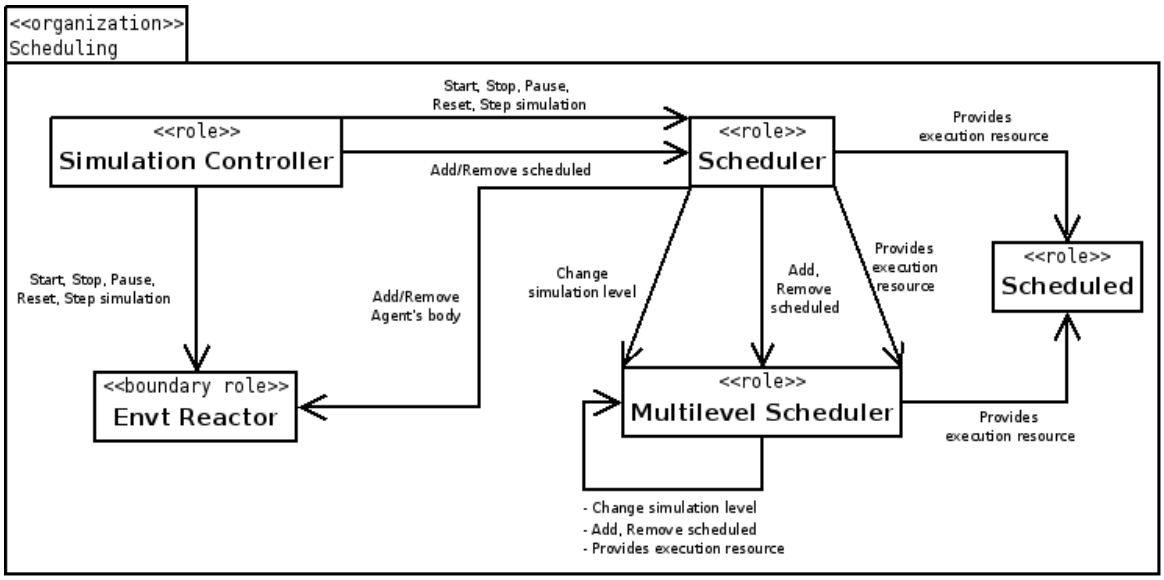


Figure 6.6: Le diagramme UML de l'organisation d'ordonnancement multi-niveaux

dans la figure 6.6. Elle définit les cinq rôles suivants :

- l'ordonnanceur (*Scheduler*) fournit l'ensemble des droits et des moyens nécessaires à l'exécution et à l'ordonnancement des holons qui jouent le rôle *Multilevel Scheduler* ou *Scheduled*. Pour ce faire, il gère un ensemble de politiques d'exécution représentées par des activateurs, selon un principe équivalent à celui utilisé dans Madkit. Ce rôle d'ordonnanceur doit être obligatoirement joué par un holon qui dispose de sa propre ressource d'exécution.
- Le rôle *Scheduled* permet aux holons qui le joue de pouvoir disposer des ressources nécessaires pour exécuter leurs rôles et leurs capacités au moment où l'ordonnanceur le juge opportun.
- Le rôle *Multilevel Scheduler* ne peut être joué que par un super-holon (holon composé) et représente la fusion des deux précédents rôles. Il permet ainsi au holon qui le joue, de pouvoir exécuter ses rôles et ses holons membres. Ce rôle fournit l'ensemble des moyens nécessaires pour intégrer les contraintes spécifiques liées au changement de niveau et ainsi déterminer s'il doit exécuter ses propres rôles (ou un sous-ensemble de ceux-ci), ses membres ou éventuellement les deux (ex : durant une phase de transition entre niveaux d'abstraction).
- Le rôle *Environmental Reactor* représente l'environnement de la simulation dans cette organisation et permet aux autres rôles de pouvoir interagir avec lui si nécessaire.
- Le *Simulation's Controller* est dédié au contrôle de la simulation et aux interactions avec l'extérieur de la simulation (interface graphique, paramètres initiaux, ...).

Dans le cadre de l'approche organisationnelle, le fait d'exécuter un holon est modélisé comme une interaction entre les rôles *Scheduler* et *Scheduled* où le premier fournit la

ressource d'exécution au second.

À la différence de Madkit, l'ordonnancement dans *Janus* touche à la fois l'exécution des holons, mais également celui des rôles qu'il joue (puisqu'un rôle est une entité à part entière). Cette organisation ne permet de gérer que la politique d'exécution des holons. Chaque holon peut ensuite disposer de ses propres politiques d'exécution pour gérer l'ordonnancement des rôles qu'il joue.

Disposant désormais des moyens nécessaires à l'ordonnancement des holons d'un système, il est temps de considérer la combinaison de cette approche organisationnelle de l'ordonnancement avec le modèle multi-niveaux du système à simuler.

6.5 Intégration du modèle multi-niveaux d'un système avec le modèle d'exécution

Le modèle multi-niveaux d'un système correspond globalement à une hiérarchie comportementale. Chaque niveau de cette hiérarchie correspond à un niveau d'abstraction (microscopique, mésoscopique, macroscopique, etc) pour le comportement considéré. Un comportement d'un niveau d'abstraction donné est représenté par un rôle. Ce rôle est évidemment dépendant du système à simuler et donc de l'application ou du domaine de l'application. Il est référencé dans la suite de ce document par le terme : *rôle applicatif*.

Au plus bas d'une hiérarchie comportementale, on trouve l'ensemble des rôles dont le niveau d'abstraction est considéré comme le plus précis de la simulation, généralement le niveau microscopique. Au niveau directement supérieur, un rôle correspond au comportement agrégé d'un ensemble de rôles du niveau inférieur. Ce schéma d'agrégation est ensuite reproduit jusqu'à obtenir au sommet de la hiérarchie un unique comportement, généralement qualifié de macroscopique, capable de simuler la dynamique du système dans sa globalité (pour le comportement considéré).

La holarchie d'exécution est généralement construire en suivant une approche ascendante, *"bottom-up"*. Les comportements du niveau le plus bas de la hiérarchie comportementale du système sont associés à des holons atomiques en charge de les exécuter. Ces holons atomiques sont regroupés et associés à un super-holon. Le mécanisme de regroupement successif des holons est ainsi reproduit jusqu'à obtenir la holarchie d'exécution du système. Chaque super-holon de la holarchie d'exécution joue au moins deux rôles : (i) un rôle applicatif dont le niveau d'abstraction est directement supérieur aux rôles joués par ses membres. Le comportement de ce rôle correspond à une approximation du comportement des rôles des membres. (ii) et le rôle *Multilevel Scheduler* défini par un groupe d'ordonnancement. En fonction des contraintes de la simulation, un super-holon déterminera s'il doit exécuter son rôle applicatif ou celui de ses membres. En fonction de cette décision, la simulation sera localement plus ou moins précise. Si tous les super-holons de la simulation exécutent leurs membres respectifs, l'ensemble des holons du niveau le plus bas de

la holarchie sera exécuté. La simulation est alors globalement à son niveau le plus précis. Inversement si seul le rôle applicatif associé au holon situé au sommet de la holarchie est exécuté, la simulation est à son niveau de précision le plus faible. La holarchie d'exécution permet ainsi de moduler dynamiquement le niveau de précision d'une simulation.

Les moyens nécessaires à un super-holon pour déterminer s'il doit ou non exécuter ses membres sont dépendants de l'application. La section 6.6 présente une manière possible d'effectuer ce choix, en exploitant un ensemble d'indicateurs, lesquels permettent d'évaluer la qualité de la simulation.

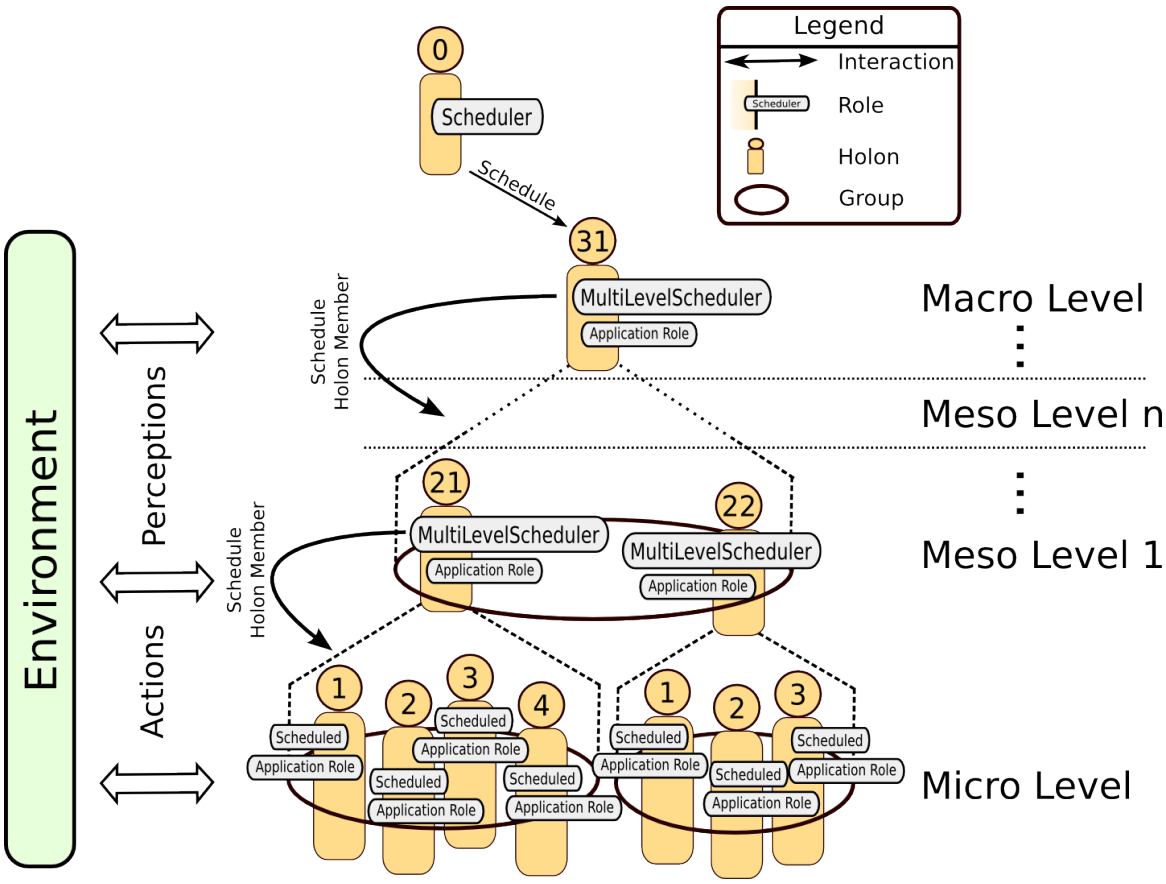


Figure 6.7: Un exemple de structure concrète de la holarchie d'ordonnancement multi-niveaux

Un exemple de la structure possible d'une holarchie d'exécution multi-niveaux est présenté sur la figure 6.7. Dans cet exemple, tout holon joue à la fois un rôle applicatif et un rôle d'ordonnancement. Le niveau le plus bas de la holarchie correspond au niveau microscopique, le niveau intermédiaire à une première approximation mésoscopique. Enfin, le sommet de la hiérarchie correspond au niveau macroscopique. Simuler un groupe d'entités à un niveau d'abstraction donné signifie que le super-holon du niveau considéré n'exécutera pas ses membres, mais son rôle applicatif. Le système est simulé au niveau microscopique, si tous les holons, composant le plus bas niveau de la hiérarchie, sont exécutés.

Si le holon H_{21} par exemple décide de ne pas exécuter ses membres, le groupe des

holons H_1, H_2, H_3 et H_4 sera simulé au niveau mésoscopique 1. Chaque super-holon peut ainsi en fonction de contraintes définies et intégrées dans son rôle *Multilevel Scheduler* décider du niveau de simulation qu'il considère comme le plus approprié en fonction des contraintes de la simulation.

6.6 Vers l'évaluation de la cohérence d'une simulation multi-niveaux

Dès lors que l'on considère différents niveaux d'abstraction au sein d'une même simulation, la question de la transition entre ces niveaux devient cruciale (cf. section 6.2.3, page 159). Effectuer une transition entre deux niveaux d'abstraction implique que les modèles utilisés pour chacun d'eux soient compatibles. Cette section introduit quelques outils destinés à faciliter le travail du concepteur d'une simulation et l'élaboration de modèles compatibles. Ces outils permettent également d'évaluer le moment opportun pour effectuer un changement de niveau en fonction des contraintes imposées par le contexte de la simulation.

Le niveau le plus précis auquel il est possible de simuler un système donné est le niveau microscopique. Dès lors que l'on considère des niveaux d'abstraction plus élevés que le niveau microscopique, la précision de la simulation diminue. Les niveaux mésoscopiques ou macroscopiques ne constituent qu'une approximation du comportement du système selon un certain point de vue. L'objectif des outils fournis dans cette section vise à estimer le niveau de qualité de cette approximation.

Ce problème peut être apparenté à un problème plus large au sein des SMA qui consiste à évaluer la précision, l'exactitude ou l'efficacité d'un système relativement à la tâche qu'il doit effectuer et aux mécanismes locaux impliqués dans l'accomplissement de cette tâche. Diverses approches ont déjà été proposées dans la littérature, certaines s'inspirant de la biologie (fonction de *"fitness"*), d'autres de la sociologie (fonction d'utilité, satisfaction, ...), ou encore de la physique (entropie). Les solutions inspirées de la physique sont probablement les plus répandues. Parmi elles, l'entropie a été couramment employée, en particulier dans le domaine des SMA réactifs, pour représenter ou tenter de mesurer le désordre (respectivement l'ordre ou l'organisation) au sein d'un système. Diverses méthodes ont été proposées pour calculer l'entropie d'un SMA depuis la notion d'entropie sociale hiérarchique de [Balch, 2000], à l'entropie dynamique et statique de [Parunak and Brueckner, 2001] (et [Parunak et al., 2002]). [Van Aeken, 1999] utilise également l'entropie pour tenter de déterminer le meilleur compromis entre la taille et l'équilibre de sa structure hiérarchique dans les *systèmes multi-agents minimaux*.

Même si cette mesure s'avère utile dans de nombreux cas, l'entropie possède deux inconvénients majeurs. Tout d'abord, elle dépend des transformations passées du système : l'entropie ne peut pas être considérée comme une fonction d'état. En cela deux systèmes

identiques peuvent être dans le même état, mais avoir des entropies différentes. En second lieu, l'entropie est principalement une mesure globale qui ne tient pas compte des phénomènes locaux du système.

Une solution relativement généraliste réside dans le calcul de l'énergie comme fonction d'état au niveau de l'agent et du système [Contet et al., 2007]. Cette solution est bien adaptée aux systèmes holoniques grâce aux propriétés compositionnelles de l'énergie. Cependant, l'énergie n'est pas toujours facilement calculable. Elle est essentiellement réservée aux modèles de comportements basés sur la notion de force (ou ceux permettant une analogie entre la sémantique des énergies et celles des variables comportementales).

D'autres méthodes intéressantes basées sur la physique statistique ou la thermodynamique sont de plus en plus utilisées [Baras and Tan, 2004, Martinas, 2004]. Ces méthodes sont basées sur la fonction de partition Z[4] de laquelle découle toutes les fonctions d'état dans les systèmes thermodynamiques telles que la fonction de Gibbs, l'Énergie libre, l'Enthalpie, l'Enthalpie libre, etc. Ces méthodes basées sur la fonction Z sont capables de tenir compte statistiquement de tous les éléments comportementaux des composants d'un système afin de calculer une valeur globale représentative. La principale difficulté de ces méthodes concerne les conditions restrictives de son application. En effet, la physique statistique a pour but d'expliquer le comportement et l'évolution de systèmes physiques comportant un grand nombre de particules (systèmes macroscopiques), à partir des caractéristiques de leurs constituants microscopiques (les particules). Pour être en mesure de calculer une valeur significative, le nombre des individus (ici des agents) au sein du système doit être suffisamment important pour être statistiquement significatif.

La formulation de ces indicateurs est bien souvent dépendante de l'application étudiée et donc du modèle du système. Chacun doit déterminer la meilleure analogie face au problème en cours d'étude. Dans le cadre de l'approche proposée pour la simulation de piétons, une version basée sur le calcul des énergies a été choisie. En effet, dès lors qu'il y a mouvement (déplacement des piétons), la vitesse et l'accélération peuvent être calculées.

Le principe de base de l'approche proposée réside dans le calcul de l'énergie des holons composant la holarchie d'exécution. Le niveau de qualité de l'approximation réalisée à un niveau donné de la holarchie est obtenu par comparaison successive des énergies des holons des niveaux inférieurs. Trois types d'énergie sont pris en compte :

- L'énergie cinétique E_{c_i}, mesure liée à la dynamique, ou plus exactement à la vitesse du holon i considéré.
- L'énergie potentielle objectif E_{pg_i}, dépendante de l'objectif du holon i
- L'énergie potentielle de contrainte E_{pc_i}, liée aux éléments qui entravent la progression du holon i vers son objectif : conflits avec les autres holons, obstacles éventuels de l'environnement, etc.

Ces trois mesures peuvent être considérées comme des fonctions d'état, car elles ne dé-

[4]Pour plus de précision voir : `http://fr.wikipedia.org/wiki/Fonction_de_partition`

pendent que des paramètres et caractéristiques courants d'un holon tels que la vitesse, la position relative à l'objectif à atteindre, les positions des obstacles et des autres holons (dans un périmètre d'influence donné). De plus ces indicateurs peuvent être calculés quelque soit le niveau d'abstraction (le niveau dans la holarchie) pris en compte. Que ce soit pour un holon représentant un composant atomique du modèle ou un groupe de composants, la formulation de ces indicateurs énergétiques ne change pas.

Sur la base de ces trois énergies, il est possible de calculer l'énergie globale E_k d'un holon k donné dans la holarchie selon la définition fournie dans l'équation 6.1. Cette énergie globale caractérise l'état courant du holon k.

$$E_k = E_{c_k} + E_{pg_k} + E_{pc_k} \tag{6.1}$$

L'évaluation de la qualité de l'approximation fournie à un niveau n donné dans la holarchie est alors basée sur la comparaison des énergies entre niveaux adjacents. On définit ainsi la notion de similarité s entre niveaux de simulation de la manière suivante :

$$s_{n+1} = (\Delta E)_{n+1} = E_j^{n+1} - E_i^n \tag{6.2}$$

avec E_i^n l'énergie d'un holon i de niveau n et E_j^{n+1} l'énergie de son super-holon j (niveau $n+1$).

Si un super-holon décide de ne pas exécuter ses membres, l'énergie de ceux-ci sera tout de même calculée. La similarité entre l'énergie d'un membre et celle de son super-holon, permet de déterminer si le comportement agrégé associé au super-holon constitue une approximation acceptable du comportement de ce membre. La moyenne de ces similarités permet de déterminer si l'approximation est acceptable pour tous les membres. Si cette moyenne dépasse un seuil donné, le super-holon doit alors dans la mesure du possible (en respectant les autres contraintes) tenter d'exécuter ses membres et donc d'affiner la simulation. Cette approche permet également de détecter lorsque le comportement d'un membre en particulier n'est plus correctement approximé, et si celui-ci doit éventuellement changer de groupe ou être simulé à un niveau plus précis. Même lorsque les autres contraintes de la simulation, en l'occurrence le volume de ressources disponibles, interdisent d'affiner la simulation, ces indicateurs permettent à l'utilisateur de la simulation de disposer d'un outil lui permettant d'estimer le décalage entre le niveau de précision de sa simulation et le niveau le plus précis susceptible d'être atteint (le niveau situé au plus bas de la holarchie d'exécution). Par extrapolation, l'utilisateur peut estimer le décalage entre la simulation et le fonctionnement réel de son système. Un exemple d'instanciation de ces indicateurs énergétiques est fourni au chapitre suivant dans le cadre d'une simulation de piétons en environnement virtuel (cf. section 7.4, page 198).

Si la simulation dispose de toutes les ressources de calcul dont elle a besoin, la simulation sera réalisée au niveau le plus précis. En somme les holons du niveau 0 de la holarchie

seront toujours exécutés. En revanche, si ces ressources viennent à manquer, le simulateur est capable de déterminer les éléments qui nécessitent une allocation prioritaire des ressources disponibles. En effet, les indicateurs permettent d'identifier quels sont les holons qui ne sont pas exécutés et dont le comportement est approximé de manière insatisfaisante par leur super-holon. Ces indicateurs permettent ainsi d'identifier quels sont les holons qui doivent en priorité bénéficier des ressources disponibles et où la simulation doit être affinée. Ces indicateurs énergétiques sont intégrés au comportement du rôle *Multilevel Scheduler* afin de faciliter le choix entre l'exécution de ses rôles applicatifs ou celles de ses membres. Ce choix est équivalent à déterminer le niveau de simulation le plus adapté en accord avec les ressources de calcul disponible. Enfin, les indicateurs énergétiques contribuent également au calcul de l'affinité entre deux holons d'un même niveau d'abstraction de sorte à optimiser la structure de la holarchie d'exécution en fonction des contraintes de la simulation. Ce dernier point sera détaillé au chapitre suivant (cf. section 7.2, page 189).

6.7 Conclusion

Nous avons présenté un ensemble d'outils destinés à la simulation multi-agents multi-niveaux. Cette conception est basée sur trois types de modèles multi-niveaux : le modèle du système cible, le modèle d'environnement, et le modèle d'exécution. Les modèles du système et de l'environnement sont ensuite projetés sur deux holarchies distinctes en charge de leur exécution. Ces holarchies offrent le support nécessaire à la mise en place des mécanismes de changement de niveau et à la modulation dynamique de la complexité des comportements d'une simulation. Elles sont indépendantes l'une de l'autre.

Les outils de gestion d'une simulation, décrits dans ce chapitre, reposent sur un modèle d'organisation en charge de la gestion de l'ordonnancement et de la synchronisation des holons au sein de la simulation. Cette organisation constitue la base de la conception d'un modèle d'exécution. Elle permet de modéliser la structure organisationnelle d'un simulateur sous forme de groupes et de rôles en accord avec la structure du modèle à exécuter et les différents niveaux d'abstraction qui le composent.

Ce chapitre introduit également un ensemble d'indicateurs, définis en fonction de l'application, et qui permet d'évaluer l'écart entre deux niveaux de simulation adjacents en prenant comme référence le niveau le plus précis disponible. Ces indicateurs tentent ainsi d'apporter une réponse au problème complexe de l'évaluation de la cohérence d'une simulation multi-niveaux. Ils permettent également d'identifier les éléments de la simulation qui nécessitent en priorité l'attribution des ressources de calcul disponibles. Ces outils participent à garantir le meilleur compromis entre les contraintes d'exécution de la simulation et son niveau de précision. Ils permettent également d'évaluer le moment opportun pour effectuer un changement de niveau d'abstraction.

Les travaux présentés dans ce chapitre sont à considérer comme un guide pour la

conception de simulations multi-niveaux, et nécessitent d'être approfondis. Néanmoins, ils représentent une première étape vers la conception d'un simulateur multi-niveaux, et de futures recherches seront menées pour les compléter. Cette approche a été appliquée avec succès pour développer une simulation de piétons en environnement urbain virtuel qui sera décrite au chapitre suivant. Cette application montre comment calculer et utiliser concrètement les indicateurs physiques proposés.

« The mystery deepens when we observe the kaleidoscopic nature of large cities. Buyers, sellers, administrations, streets, bridges, and buildings are always changing, so that a city's coherence is somehow imposed on a perpetual flux of people and structures. [...], a city is a pattern in time. No single constituent remains in place, but the city persists. [...] Agent formed by aggregation are a central feature, [...] At one level of abstraction the complex system that is New York city is well described by evolving interactions of these agents. [...] That New York retains both a short-term and a long-term coherence, despite the diversity, change, and lack of central direction, is typical of the enigmas posed by complex adaptive systems. As is usual, nonlinerarities lie near the center of the enigma. »

John H. Holland

Hidden order : how adaptation builds complexity,

1995, Addison Wesley Longman Publishing

CHAPITRE 7

SIMULATION MULTI-NIVEAUX DE PIÉTONS EN ENVIRONNEMENT URBAIN VIRTUEL

LA SIMULATION DE PIÉTONS en environnement urbain virtuel présentée dans ce chapitre, est utilisée comme une application des systèmes multi-agents holoniques. Le modèle proposé est basé sur le métamodèle CRIO et exploite sa capacité à modéliser un système à différents niveaux d'abstraction. L'approche décrite au chapitre précédent, est utilisée pour gérer la mise en œuvre des mécanismes de simulation multi-niveaux. Un modèle multi-niveaux est également proposé pour l'environnement urbain de la simulation de piétons.

Sommaire

7.1 Introduction

Les systèmes urbains sont des exemples de systèmes complexes. Structurellement, un système urbain peut être décomposé hiérarchiquement sous forme de rues, bâtiments, quartiers, arrondissements, etc. Économiquement et socialement, il peut être perçu comme une structure composée d'individus, de familles, d'entreprises, ou de commerces. La complexité des environnements urbains peut être abordée selon de nombreux points de vue [Allen, 1997]. Toutefois, deux principaux points de vue peuvent être adoptés. Le premier concerne l'évolution de la structure urbaine (formation des villes, morphologie, expansion, etc.) et le second traite davantage des activités sociales, et notamment des modèles de trafic et de déplacement de foules [Jiang, 1999]. La simulation multi-agents est un outil particulièrement bien adapté à la simulation des dynamiques urbaines. Les SMA s'avèrent plus flexibles que les modèles macroscopiques à base d'équations différentielles pour simuler des phénomènes spatiaux et évolutifs [Bretagnolle et al., 2003].

Ce chapitre présente un modèle de simulation multi-niveaux de piétons en environnement urbain virtuel. Pour concevoir cette simulation, la modélisation de la dynamique d'une population de piétons doit être clairement distinguée de celle de la structure de l'environnement urbain. En effet, la modélisation structurelle de la ville se rapporte à l'aspect environnemental de la simulation, alors que le modèle de la dynamique des piétons concerne davantage le modèle du système. Dans ce chapitre, un modèle multi-niveaux pour l'environnement urbain est proposé. Les environnements urbains exhibent naturellement une structure spatiale hiérarchique, voire holonique [Pun-Cheng, 2000], qui facilite amplement l'adaptation du modèle proposé. Le modèle multi-niveaux du système reflète les différents niveaux d'abstraction d'une population de piétons (individu, groupe, foule, etc) et doit intégrer les données disponibles au niveau individuel (issues des sondages et d'enquêtes ménages par exemple) avec les observations statistiques agrégées d'une population dans son ensemble. Ces différents modèles (environnement et système) sont ensuite combinés avec les outils de gestion de la simulation multi-niveaux. Ces outils assurent dynamiquement les transitions entre les différents niveaux d'abstraction du comportement des piétons, et adaptent automatiquement le niveau de décomposition de l'environnement urbain.

L'application décrite dans ce chapitre s'intègre dans un projet de plus grande envergure du laboratoire SeT[1], visant à fournir un environnement logiciel permettant de simuler la dynamique d'une ville et les différents modes de déplacement : piétons, automobiles, bus, vélos, etc. Ce chapitre n'aborde que le cas de la simulation de piétons, mais les modèles proposés peuvent être étendus à d'autres types d'entités. L'objectif global de cette application est de disposer d'un outil complet d'aide à la décision pour tester et valider différents scénarios de planification, d'aménagement du territoire et d'optimisation des réseaux transports au sein d'une ville. Cette application se situe ainsi au confluent de nombreux domaines

[1] Systèmes et Transport : `set.utbm.fr`

d'expertise : la simulation multi-agents, la géomatique, et la réalité virtuelle. En effet, les systèmes multi-agents sont utilisés pour simuler la dynamique urbaine. La construction des modèles d'environnements urbains et l'initialisation des simulations sont directement basées sur les connaissances capitalisées au sein des systèmes d'information géographique (SIG). Enfin, la réalité virtuelle permet aux utilisateurs de la simulation de pouvoir évoluer dans l'environnement urbain virtuel et éventuellement interagir avec les entités qui le peuplent. Le simulateur est ainsi couplé à une visualisation 2D ou 3D de l'environnement urbain virtuel. Il doit également être en mesure de fonctionner sur une plate-forme de réalité virtuelle de sorte à pouvoir immerger un ou plusieurs utilisateurs dans le système simulé.

La combinaison de ces différents outils contraint sensiblement la conception de la simulation. La connexion éventuelle avec une plate-forme de réalité virtuelle constitue l'une des principales contraintes. La simulation doit ainsi garantir des performances temps réel (ou très proche du temps réel) afin d'assurer une immersion optimale de l'utilisateur. Cette contrainte justifie l'utilisation des mécanismes de gestion multi-niveaux, et impacte l'intégralité du modèle de la simulation : depuis le comportement des agents peuplant l'environnement virtuel, jusqu'à la complexité géométrique et graphique de cet environnement. La simulation multi-niveaux est utilisée pour garantir un réalisme maximal dans la simulation tout en conservant des performances proches du temps réel. Elle doit maintenir le meilleur compromis entre la complexité de la simulation et les ressources de calcul disponibles. La complexité de la simulation est directement liée à la complexité des comportements des piétons ainsi qu'à la finesse de la décomposition environnementale et de sa représentation graphique. Afin de garantir un niveau de réalisme optimal pour l'utilisateur immergé dans le monde virtuel, toutes les entités qui se situent dans un rayon donné autour de l'utilisateur doivent être simulées le plus précisément possible. De plus, dans une simulation en environnement urbain, tous les lieux ne disposent pas de la même importance aux yeux de l'utilisateur, certains points définis tels que les principaux nœuds d'un réseau de transport, les gares ou les centres industriels nécessitent d'être simulés avec le maximum de précision possible compte tenu des ressources fournies. Cette catégorisation des espaces simulés impose de nouvelles contraintes pour l'allocation de ressources de calcul disponibles. Ces diverses contraintes déterminent les espaces fixes ou mobiles considérés comme cruciaux dans la simulation.

Dans ce contexte, la section 7.2 est consacrée à la description du modèle multi-niveaux utilisé pour la simulation des piétons. La section 7.3, quant à elle, détaille les aspects relatifs à la modélisation et l'implantation de l'environnement urbain virtuel, et à sa connexion avec le système multi-agents en charge de la simulation des piétons. Les mécanismes de perception et d'action mis à la disposition des agents sont également décrits. L'instanciation des indicateurs inspirés de la physique visant à déterminer le décalage de précision entre deux niveaux de simulation adjacents est l'objet de la section 7.4. Enfin, la section 7.5 est consacrée à la présentation des résultats expérimentaux illustrant le fonctionnement de

l'approche proposée, et montrant expérimentalement la cohérence des indicateurs énergétiques proposés.

7.2 Simulation multi-niveaux de piétons

Simuler des foules de piétons en environnement virtuel 3D implique d'être en mesure de simuler le mouvement d'un grand nombre d'entités tout en maintenant un taux de rafraîchissement de l'interface graphique (2D et 3D) proche du temps réel. Les SMA sont particulièrement adaptés pour simuler de tels types de comportement au niveau le plus précis où chaque piéton peut disposer d'objectifs et de caractéristiques propres. Toutefois, les SMA requièrent rapidement des ressources de calcul importantes. Il devient dès lors, rapidement difficile de maintenir les performances requises par l'interface graphique et l'environnement virtuel. Cette problématique est un exemple typique dans lequel le niveau de complexité des comportements, et de la simulation en général, doit être adapté en fonction des ressources de calcul disponibles.

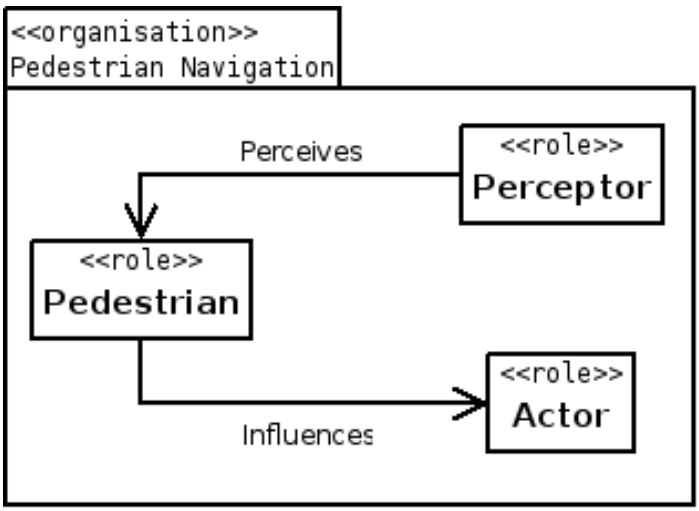


Figure 7.1: L'organisation de gestion des comportements de piétons

En accord avec l'approche organisationnelle, le comportement des piétons est modélisé en termes de rôles et d'interactions, puis intégré à l'organisation décrite dans la figure 7.1. Trois rôles sont identifiés : *Actor*, *Perceptor* et *Pedestrian*. Les deux premiers fournissent à un piéton les moyens de percevoir et de se déplacer dans le monde virtuel, ces points seront davantage détaillés dans la section 7.3.2.3. Le comportement proprement dit d'un piéton est modélisé par le rôle *Pedestrian* qui est en charge de simuler le comportement d'un piéton unique ou d'un groupe de piétons en fonction du niveau d'abstraction considéré. Cette organisation est ensuite combinée au modèle multi-niveaux pour créer la holarchie d'exécution du système, selon l'approche décrite à la section 6.5 du chapitre 6.

Différents niveaux d'abstraction sont considérés pour le rôle *Pedestrian*. Le niveau le

plus précis correspond au niveau microscopique : un piéton est associé à un holon. Chaque piéton dispose d'un point de départ dans le monde virtuel (ex : son domicile), et d'un ou plusieurs objectifs à atteindre (ex : son lieu de travail, supermarché, etc). Au niveau supérieur, qualifié de *macroscopique*, chaque super-holon simule le comportement d'un groupe de piétons. Le groupe est considéré comme un piéton à part entière dont la zone de perception est élargie, et dont l'objectif correspond à la moyenne des objectifs de ses membres. Le rôle *Pedestrian* joué par le super-holon reste le même qu'au niveau inférieur, mais les perceptions et les actions sont agrégées. Le corps ou la représentation du super-holon dans le monde virtuel correspond à l'agrégation des corps de ses membres. Lorsqu'un mouvement est calculé au niveau macroscopique, le corps des membres du super-holon considéré est déplacé selon le mouvement considéré. L'auto-similarité des holons est directement exploitée pour réutiliser le même comportement à différents niveaux d'abstraction. Le large panel de niveaux de détails possibles pour les piétons est illustré par la figure 7.2.

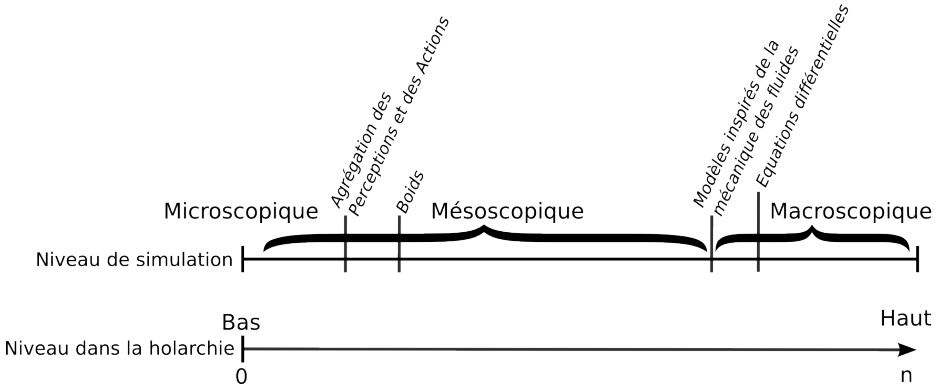


Figure 7.2: Les différents niveaux de simulation possibles

Le comportement proprement dit d'un piéton est inspiré d'un modèle classique basé sur un ensemble de forces d'attraction et de répulsion (*"social force model"*, *"magnetic force model"*[2]). Les modèles de comportements proposés par [Bierlaire et al., 2003, Gloor et al., 2003, Helbing et al., 2000, Teknomo et al., 2001] peuvent également être utilisés. Le comportement d'un piéton (respectivement d'un groupe de piétons) est basé sur trois forces : (i) la force d'attraction de l'objectif, tractant le piéton vers son prochain objectif local (équation 7.1) ; (ii) la force résultante des forces de répulsion des obstacles de l'environnement (équation 7.3) ; (iii) la force résultante des forces de répulsion issues des autres piétons (équation 7.2).

Force $\overrightarrow{F}_{obj}$ qui tracte l'agent i vers son prochain objectif local :

$$\overrightarrow{F}_{obj} = \beta_{obj}.\overrightarrow{A_i G} \tag{7.1}$$

[2]Pour davantage de détails sur ces modèles de simulation de piétons : cf. [Teknomo et al., 2000]

Force $\overrightarrow{F}_{rep_{ij}}$ de répulsion de l'agent j sur l'agent i :

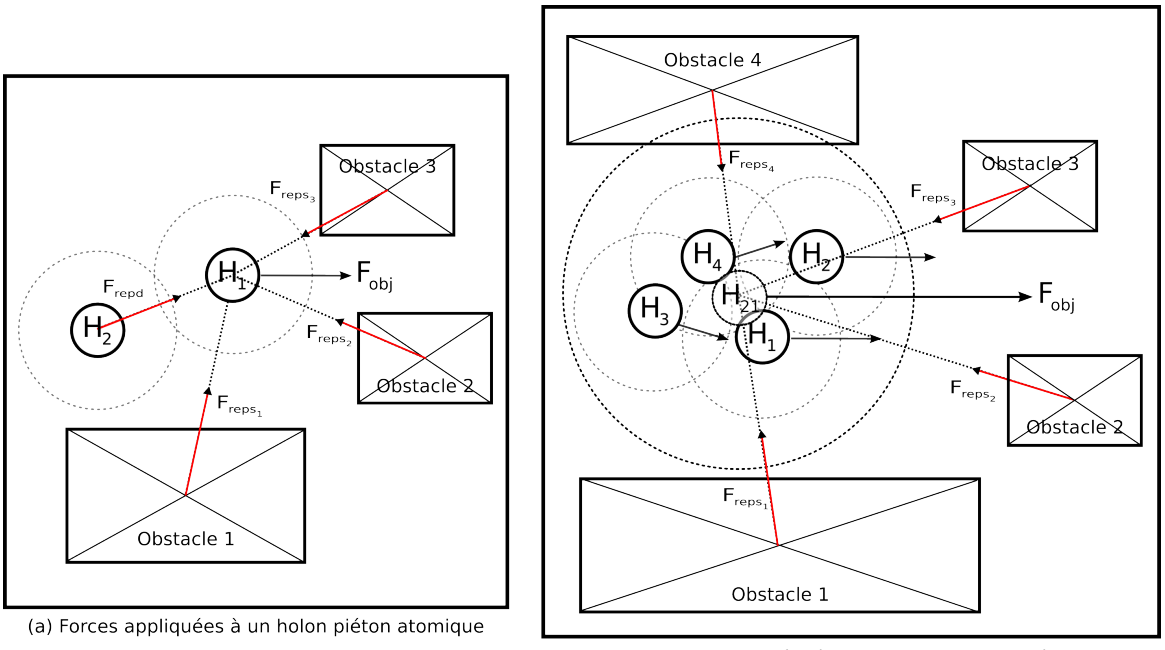
$$\overrightarrow{F}_{rep_{ij}} = \beta_{ij}.\frac{m_i.m_j}{\|\overrightarrow{d_{ij}}\|^2}.\overrightarrow{d_{ij}} \tag{7.2}$$

Force $\overrightarrow{F}_{rep_{ik}}$ de répulsion de l'obstacle k sur l'agent i :

$$\overrightarrow{F}_{rep_{ik}} = \beta_{ik}.\frac{m_i}{(d_{ik}.\sin(\alpha_{ik}))^4}.\overrightarrow{n_{ik}} \tag{7.3}$$

avec
- A_i la position du piéton i.
- G la position du prochain objectif local du piéton i.
- O_k la position d'un obstacle k donné, situé dans l'espace de perçu du piéton i.
- $\overrightarrow{d_{ij}}$ le vecteur entre les piétons i et j, et $\|\overrightarrow{d_{ij}}\|$ la norme de ce vecteur.
- m_i masse du piéton i.
- $d_{ik} = \|\overrightarrow{A_i O_k}\|$, distance entre le piéton i et l'obstacle k.
- $\alpha_{ik} = \angle(\overrightarrow{A_i G}, \overrightarrow{A_i O_k})$
- $\overrightarrow{n_{ik}} = \begin{pmatrix} 0 & -sign(\alpha_{ik}) \\ sign(\alpha_{ik}) & 0 \end{pmatrix} . \frac{\overrightarrow{A_i O_k}}{\|\overrightarrow{A_i O_k}\|}$
- $\beta_{obj}, \beta_{ij}, \beta_{ik}$ constantes de calibrage.

La figure 7.3(a), respectivement 7.3(b), décrit les forces appliquées à un holon piéton H_1, respectivement à un groupe de 4 piétons associés au super-holon H_{21}.



(a) Forces appliquées à un holon piéton atomique

(b) Forces appliquées à un holon groupe de piétons

Figure 7.3: Forces appliquées aux holons piétons

La holarchie en charge de l'exécution multi-niveaux du comportement des piétons est décrite dans la figure 7.4. Si les ressources de calcul nécessaires sont disponibles, le super-holon H_{21} exécute ses membres sinon son rôle *Pedestrian* est exécuté.

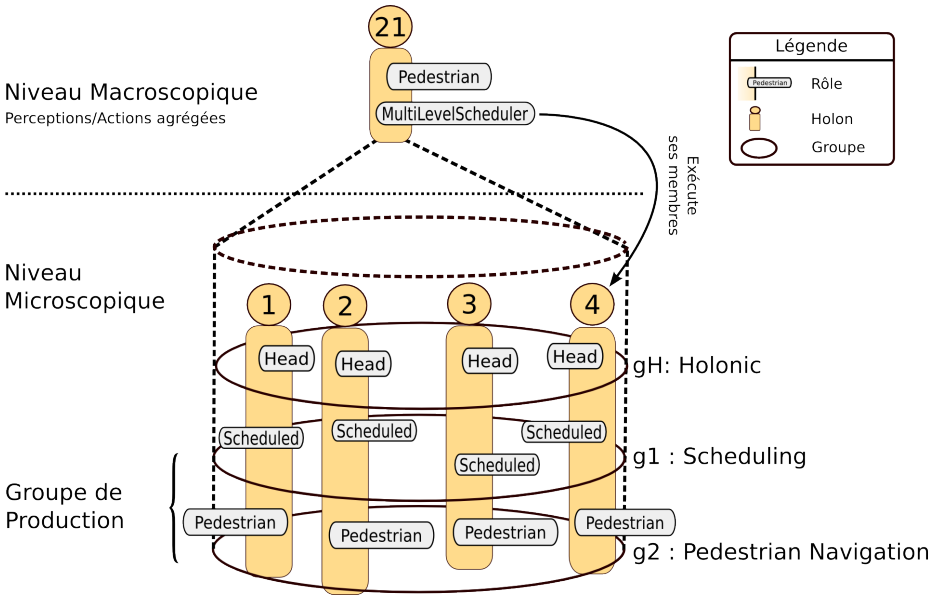


Figure 7.4: Un fragment de la holarchie d'exécution des piétons

Cette holarchie d'exécution est construire en suivant une approche ascendante *"bottom-up"* basée sur le regroupement successif des piétons sous forme de groupes. Chaque groupe est ensuite associé à un super-holon. Ces super-holons sont à leur tour regroupés jusqu'à obtenir la holarchie d'exécution du système. Le regroupement des piétons est effectué en accord avec leur affinité respective. Celle-ci est fonction de la distance entre les objectifs locaux des piétons, leurs positions courantes et leur affinité énergétique. Cette dernière est calculée de la manière suivante :

$$\text{Soit deux holons i et j, } \mathrm{Aff}_{\mathrm{ener}}(i,j) = \frac{1}{E_i^n - E_j^n} \tag{7.4}$$

La manière de calculer l'énergie E_i^n d'un holon i de niveau n dans la holarchie, en accord avec les forces auxquelles il est soumis, est détaillée à la section 7.4 de ce chapitre.

7.3 Simulation multi-niveaux d'environnements urbains virtuels

L'environnement est communément considéré comme l'une des parties essentielles d'une simulation multi-agents [Demazeau, 2001, Michel, 2004, Odell et al., 2002, Weyns et al., 2005, 2006]. Cependant, différentes perspectives existent sur le rôle qu'il joue dans un SMA et sur sa définition. Cette section se focalise sur la modélisation d'un environnement

urbain dédié à la simulation multi-agents. Ce type d'environnement peut être considéré comme un cas particulier d'*« environnement situé »*. La notion d'*« environnement situé »* se réfère à la classe de systèmes dans lesquels les agents, ainsi que les objets, disposent d'une position explicite et produisent des actions elles-aussi situées [Ferber and Müller, 1996].

Cette section présente un ensemble d'abstractions pour la modélisation d'un environnement urbain. Ces éléments viennent étendre le métamodèle CRIO, présenté au chapitre 3, et sont basés sur les approches organisationnelle et holonique. L'une des particularités de l'approche proposée est que l'environnement de la simulation est géré par un système multi-agents holonique, distinct de celui destiné à simuler le comportement des piétons.

Cette section est organisée de la manière suivante : La section 7.3.1 décrit l'architecture globale de l'environnement utilisé pour la simulation des piétons. La section 7.3.2 présente ensuite les extensions apportées au métamodèle CRIO pour concevoir un modèle multi-niveaux pour l'environnement d'une simulation multi-agents dans le cadre de l'application proposée. Les éléments permettant d'immerger un SMA dans un environnement virtuel sont également décrits.

7.3.1 Architecture globale de l'environnement urbain

Le modèle d'environnement urbain virtuel que nous proposons a été conçu pour être associé à une grande variété de modèles de systèmes pour simuler les diverses dynamiques d'une ville virtuelle : simulation de piétons, simulation des réseaux de transport (bus, automobiles, poids louds ou vélo). Cette section présente l'architecture générale de cet environnement et ses relations avec les divers outils nécessaires à la mise en œuvre de l'application : moteur graphique, plate-forme de réalité virtuelle, systèmes multi-agents, etc. Pour assurer la connexion entre cet environnement et ces différents outils, un ensemble d'*interfaces environnementales* ont été développé. Chacune d'elles correspond à un point de vue particulier sur l'environnement global de la simulation. Une interface environnementale agit tel un filtre permettant de se focaliser sur un aspect spécifique de l'environnement, regroupant les éléments nécessaires à l'application correspondante. Ces interfaces regroupent notamment les outils permettant de gérer la connexion entre un SMA et l'environnement urbain virtuel, et les mécanismes de perception et d'action disponibles pour les agents qui le peuplent. La figure 7.5 décrit l'architecture globale de l'environnement urbain proposé.

Le cœur de l'environnement de la simulation est représenté par la structure et la décomposition hiérarchique de l'environnement virtuel (*3D World*). Cette partie de l'environnement est stockée dans une base de données spécialisée, directement construite à partir des données extraites des SIG et des maquettes numériques des espaces virtuels considérés. Elle doit intégrer l'information spatiale graphique (images, textures, etc) et statistique ainsi que l'information géométrique, topologique et sémantique de l'environnement urbain. Ce type de base de données est généralement qualifié de Système d'Information Urbain

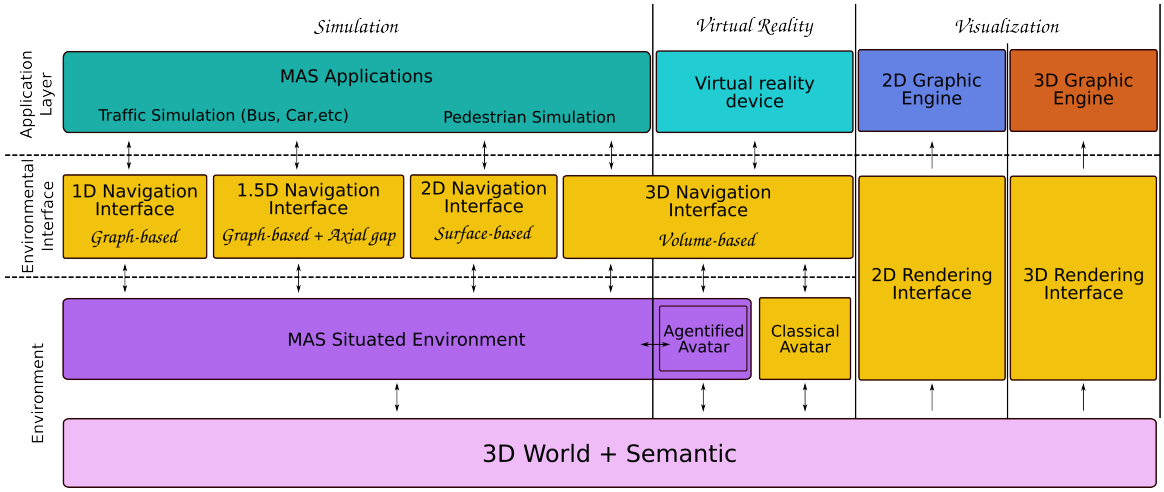


Figure 7.5: Architecture d'un système immergeant un SMA dans un environnement virtuel

(SIU) [Ben Mahbous, 1995].

Ce cœur de l'environnement peut ensuite être manipulé par les agents environnementaux, mais également par d'autres applications, notamment celles dédiées à l'affichage et à l'interface graphique telles que les moteurs graphiques (2D et 3D), et les modules de réalité virtuelle permettant à un utilisateur d'évoluer dans le monde virtuel et d'interagir avec lui. La manière de modéliser et de décomposer l'environnement urbain dans le cadre d'une simulation multi-agents est décrite dans la section suivante.

7.3.2 Extension de CRIO pour la modélisation d'environnements urbains

Dans l'application étudiée, l'environnement urbain est géré par un SMAH en charge d'assurer les différentes missions classiquement assignées à l'environnement dans une simulation multi-agents. Quatre missions principales sont distinguées [Weyns et al., 2005, 2006, 2007] :

- Partager les informations : L'environnement est avant tout considéré comme une structure partagée entre les agents, dans laquelle ils perçoivent et agissent. Cette structure peut parfois être le support d'une communication indirecte entre les agents (stigmergie).
- Gérer les actions des agents : La gestion des actions est basée sur le modèle influence-réaction (cf. section 6.2.4, chapitre 6, page 161). Les agents ne modifient pas directement l'état de l'environnement. Un agent ne gère pas directement son corps. Ce modèle permet de faciliter la gestion des actions simultanées des agents et celle des éventuels conflits. Il permet également de vérifier que les *lois de l'environnement* ne sont pas bafouées par les agents.
- Gérer la perception et l'observation : L'environnement doit être localement et partiel-

lement observable. Les agents de l'environnement gèrent l'accès aux informations environnementales et garantissent ainsi les contraintes d'intégrité environnementale et de localité des perceptions (cf. section 6.2.4, chapitre 6).

– Maintenir et gérer la dynamique interne ou endogène de l'environnement : L'environnement est une entité active. Il dispose de ses propres processus indépendants de ceux des agents du système étudié. Il est également capable de produire ses propres influences [Michel, 2006]. Un exemple typique de cette dynamique interne est incarné par la gestion de l'évaporation des phéromones artificielles dans les simulations de colonies de fourmis.

Pour assurer ces missions spécifiques attribuées à l'environnement d'un SMA situé, le métamodèle CRIO a été étendu pour intégrer les objets et organisations nécessaires à la modélisation de ces différents aspects. Cette extension est illustrée par la figure 7.6. Ce modèle introduit notamment le concept d'environnement ainsi que ces différentes composantes.

7.3.2.1 Structure d'un environnement urbain

L'environnement est composé d'une structure hiérarchique d'objet appelée *Monde* (*World*) et d'un ensemble d'organisations spécifiques en charge des différentes missions environnementales, nommées *organisations environnementales* (*environmental organizations*). Le *Monde* se réfère à l'environnement en tant que structure partagée entre les agents. Il contient la structure de l'environnement qui peut être hiérarchiquement décomposée en différentes zones, sous-zones, etc, connectées entre elles. La structure résultante s'apparente à un *"clustered graph"* [Balzer and Deussen, 2007, Eades and Feng, 1996, P. Eades, 1996], qui correspond à un arbre où chaque niveau est un graphe connectant les nœuds du niveau correspondant. Chaque zone est ensuite spécialisée en accord avec son rôle dans l'environnement urbain : quartier, route, piste cyclable, etc. La partition utilisée pour décomposer un environnement urbain est inspirée des travaux de [Pun-Cheng, 2000], [Farenc et al., 1999], et [Thomas, 1999]. Davantage de détails sur la modélisation d'environnements urbains peuvent également être trouvés dans les travaux suivants : [Benenson et al., 2001, 2005, Donikian, 1997, Perret, 2006].

Chaque *Zone* contient un ensemble d'objets environnementaux (*environmental objects*). Les corps des agents ou des holons applicatifs (du système simulé et non de l'environnement) sont considérés comme des objets environnementaux. En effet, le corps d'un agent n'est pas directement géré par celui-ci. Comme les objets, ils sont soumis aux influences de nombreux agents et pas uniquement à celles de leurs propriétaires. L'ensemble des influences produites simultanément est collecté par l'environnement. Ainsi, un agent qui souhaite se déplacer, émettra une influence de mouvement. Le corps de l'agent correspondant ne sera effectivement déplacé que si l'influence qu'il a émise ne viole aucune loi environnementale et qu'aucune autre influence ne vient contrarier son action. Un exemple simple de ce type de loi peut être trouvé dans la simulation des piétons, ex : Un holon piéton face

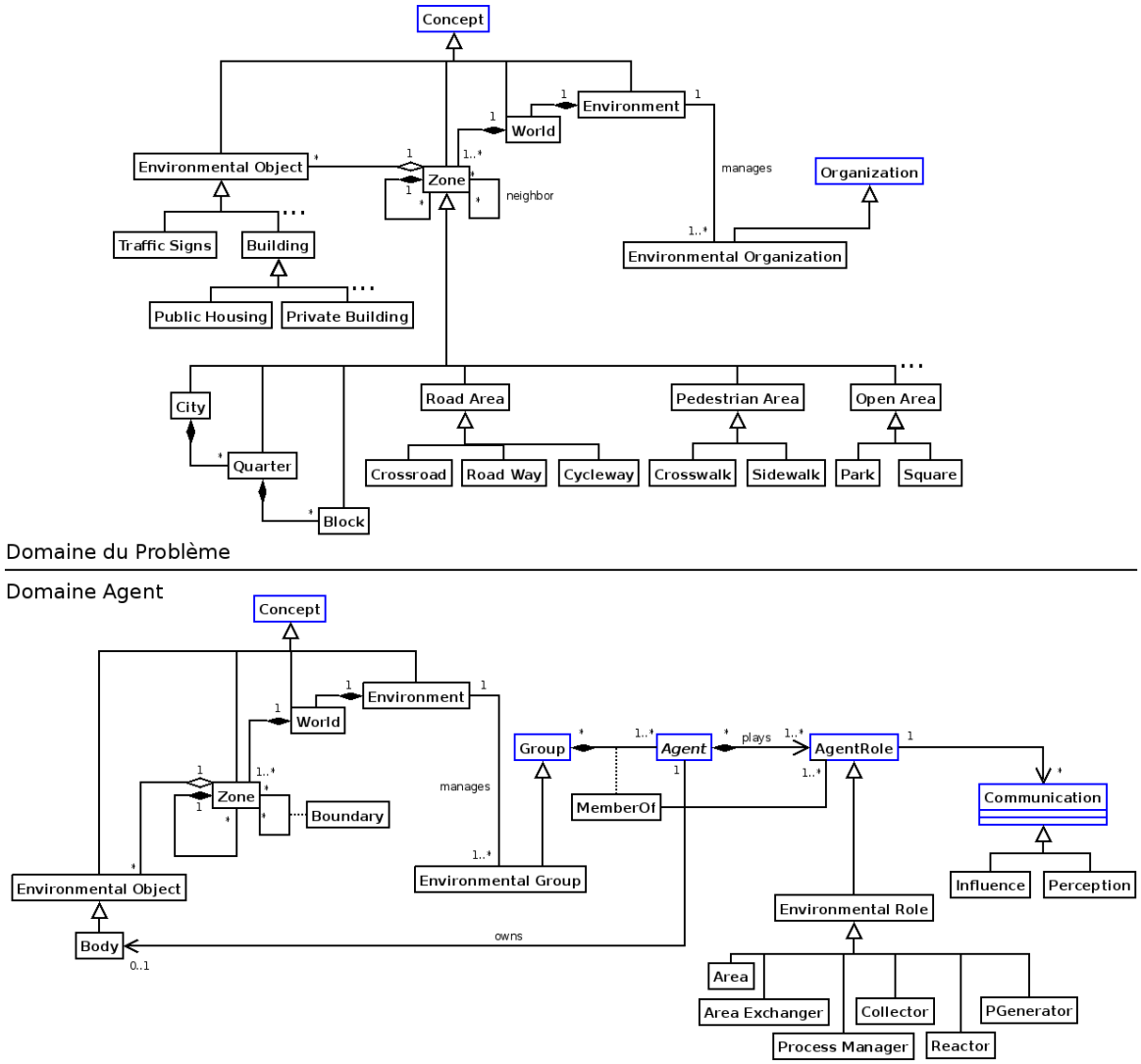


Figure 7.6: Le diagramme UML des extensions du métamodèle CRIO pour la modélisation d'environnements urbains

à un mur qui souhaite avancer, demeurera cependant immobile. En effet, les lois du monde virtuel (environnement) interdisent à tous agents de traverser les murs.

Cette section a détaillé les abstractions nécessaires pour décrire la structure d'un environnement urbain. Un modèle multi-niveaux doit certes fournir une description de la décomposition structurelle de l'environnement, mais également des comportements associés à chaque niveau. La section suivante s'attache donc à décrire le modèle des comportements de l'environnement.

7.3.2.2 Les organisations de l'environnement

Les organisations environnementales doivent accomplir les différentes missions assignées à l'environnement d'une simulation. Pour distinguer de manière claire les comportements qui composent et participent à la dynamique interne de l'environnement, des com-

portements qui composent le système cible, un type particulier de rôles a été introduit : les rôles environnementaux (*environmental role*). Les rôles environnementaux sont à distinguer des rôles frontières (*boundary role*). Ces derniers sont en effet chargés de gérer les interactions survenant à la frontière entre une application et son environnement, mais font tout de même partie du modèle du système. Alors que les rôles environnementaux décrivent les rôles qui participent à la satisfaction de besoins purement liés à l'environnement et sont par conséquent considérés comme extérieur au modèle du système cible.

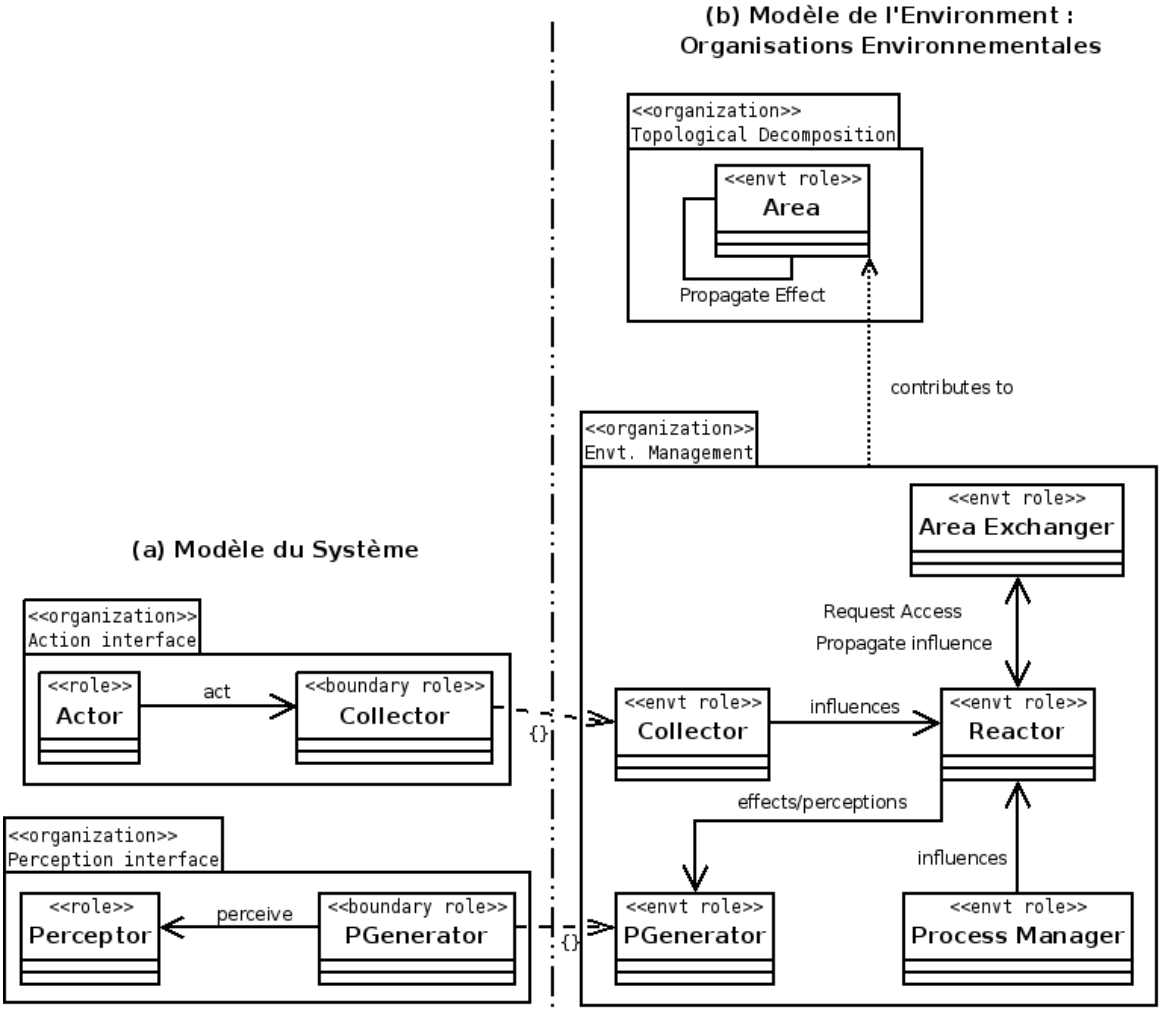


Figure 7.7: Le diagramme UML des organisations environnementales

Deux organisations environnementales principales ont été identifiées et sont décrites dans la figure 7.7(b). La première, l'organisation *Topoligical Decomposition*, est en charge d'assurer et de définir le niveau et la granularité de la décomposition hiérarchique de l'environnement. Elle définit le rôle *Area* joué par un super-holon en charge de gérer une *zone* particulière de l'environnement. L'organisation de gestion environnementale (inspirée de [Weyns and Holvoet, 2004]), *Environmental Management*, définie l'ensemble des rôles nécessaires à la satisfaction de l'ensemble des missions environnementales pour une zone donnée de l'environnement. Une instance de cette organisation est intégrée sous forme

d'un groupe de production à tous les super-holons jouant le rôle *Area* associée à la zone concernée. Elle définit cinq rôles :

- Le rôle *Collector* collecte l'ensemble des influences émises simultanément par tous les agents et les holons applicatifs situés sur la zone considérée, et les classe ensuite par type d'influences avant de les envoyer au *Reactor*.
- Le rôle *Area Exchanger*, quant à lui, collecte les influences provenant des zones voisines et dont l'impact ou la portée excèdent leurs seules étendues géométriques. De même, il assure le transfert d'influences émises dans la zone courante. C'est ce rôle qui assure la connexion entre les différentes zones à un niveau de décomposition donné de l'environnement.
- Le rôle *Process Manager* doit assurer la dynamique endogène de l'environnement sur la zone courante. Il est donc capable d'émettre des influences au même titre qu'un agent applicatif. Ce rôle assure par exemple l'évaporation des phéromones artificielles ou entretient le mouvement d'une balle (non agentifiée) qui a été poussée par un agent et qui continue encore à rouler. Il assure également le calcul des variations de variables *« globales »* (à la zone) de l'environnement telles que la température, la pression ou la gravité, etc.
- Le rôle *Reactor* : Toutes les influences émises simultanément dans un pas de temps donné sont ensuite transférées au rôle *Reactor* qui calcule la réaction environnementale en accord avec les lois de l'environnement. Il résout les éventuels conflits entre les influences qui pourraient s'interférer, et modifie l'état de l'environnement. Il informe ensuite le rôle *Percept Generator*.
- Le rôle *Percept Generator* est en charge de calculer les perceptions des agents pour le prochain pas de simulation.

La figure 7.8(b) décrit un exemple d'une holarchie environnementale et la figure 7.8(a) détaille la décomposition hiérarchique correspondante de l'environnement sous forme de zones. Deux super-holons environnementaux H_{21} et H_{22} gèrent ainsi deux zones voisines. Ils sont connectés entre eux par le holon *MultiPart* H_{23} qui joue le rôle *Area Exchanger* dans chacune de leurs organisations de gestion et permet ainsi le transfert des influences entre ces deux zones.

Il est important de retenir que les comportements sont modélisés par des rôles et qu'ils peuvent être aussi bien tous joués par le même agent ou distribués parmi un groupe d'agents. En outre, ce modèle ne s'adresse pas à des environnements de petite taille, mais à la modélisation d'environnements de grande échelle comportant un grand nombre d'objets, et où le coût du calcul de l'ensemble de la dynamique environnementale n'est pas négligeable face au coût d'exécution du comportement des agents applicatifs.

L'intérêt principal du modèle est révélé lorsqu'il est combiné avec les mécanismes d'ordonnancement multi-niveaux. Il permet alors de gérer dynamiquement le niveau de décomposition topologique ou géométrique de l'environnement et la complexité de la réaction

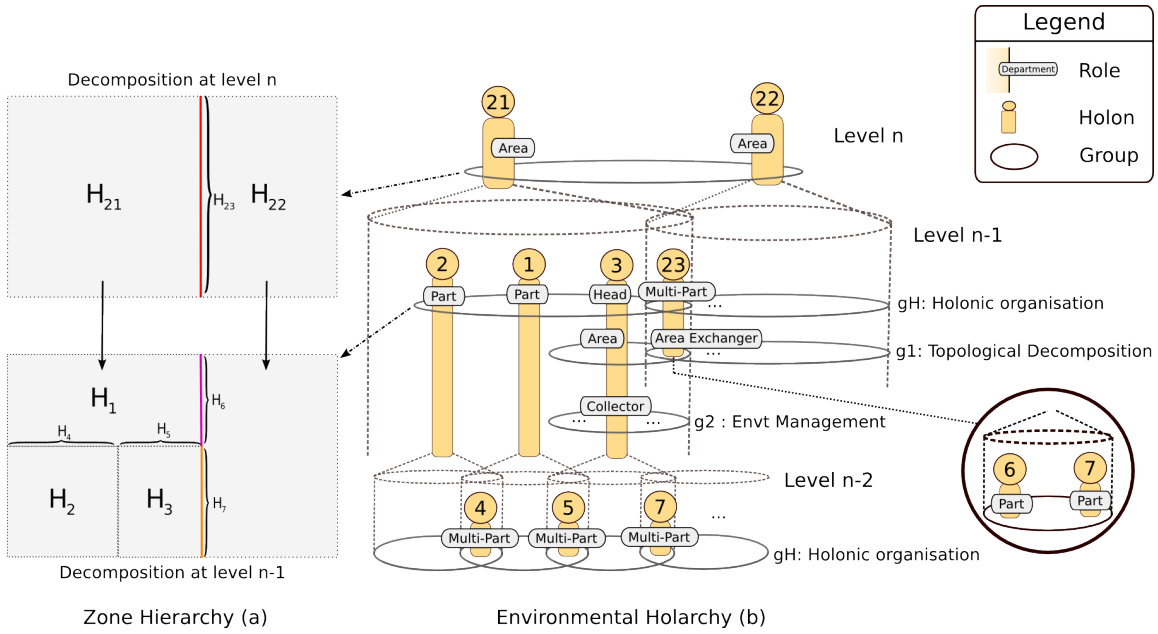


Figure 7.8: Un exemple d'une possible holarchie environnementale

environnementale associée. Par exemple, il est possible d'adapter cette décomposition en fonction de la densité de piétons présents dans la zone à gérer. Si de nombreux piétons sont regroupés sur une zone donnée, la décomposition sera affinée de sorte à disposer de davantage d'agents environnementaux pour accélérer le traitement des influences émises. En revanche, si très peu de piétons y évoluent une décomposition moins fine sera utilisée.

Après cette description des comportements en charge d'assurer la dynamique d'un environnement urbain, la section suivante détaille les relations entre le modèle d'environnement et le modèle du système cible. Etudier cette relation, implique également d'étudier les relations entre le SMAH en charge de l'exécution du comportement des piétons et celui en charge de l'exécution de l'environnement de la simulation.

7.3.2.3 Interface entre l'environnement et le système

Deux organisations spécifiques assurent l'interface entre le modèle des piétons et le modèle de l'environnement. Elles gèrent les actions et les perceptions des agents applicatifs (les piétons) au sein de la simulation. La figure 7.7(a) détaille ces deux organisations situées à la frontière entre l'environnement et le système. L'organisation *Action Interface* définit le rôle *Actor* joué par un agent applicatif du modèle du système et lui fournit les moyens nécessaires pour émettre des influences dans l'environnement. Le rôle *Collector* correspond à la représentation de l'environnement dans le modèle du système, il est en charge de collecter les influences émises par les agents applicatifs.

Respectivement l'interface de perception définit le rôle *Perceptor* qui fournit à l'agent qui le joue, les moyens nécessaires à la perception et lui permet d'utiliser des filtres spécifiques sur ces percepts ou d'effectuer des traitements particuliers en fonction du type de

percepts (approche continue ou perception active cf. [Weyns et al., 2004]). Le rôle *Percept Generator*, quant à lui, assure le transfert des perceptions de l'environnement vers les agents concernés.

Les rôles *Collector* et *Percept Generator* sont joués par des agents environnementaux qui jouent également les rôles correspondants du côté environnemental. Ce sont ces agents qui assurent le transfert d'informations (influences et perceptions) entre l'environnement et le système cible. Chacun de ces modèles peut être spécialisé en fonction de l'application concernée.

Ces modèles de gestion des perceptions et des actions précédents ont été instanciés dans le cadre de la simulation multi-agents en environnement urbain virtuel. Différents types d'interfaces environnementales ont ainsi été créés en fonction du type des entités considérées et de la complexité de leurs comportements. La figure 7.5 décrit les quatre interfaces proposées (cf. page 191). Ces quatre interfaces sont brièvement décrites ci-dessous :

Navigation 1D : Les entités se déplacent le long de courbes. Leur position est ainsi représentée par l'abscisse curviligne sur la courbe correspondante. Une influence de mouvement correspond à un pas sur la courbe ou à un pourcentage d'avancement. L'ensemble des courbes est intégré à la structure de l'environnement et pré-calculé avant la simulation. Une courbe correspond à un chemin particulier pour éviter les obstacles. Les autres entités sont également perçues en fonction de leur abscisse curviligne. Ce type d'interface est généralement associé au modèle de comportement de véhicules. Le modèle de perception associé à ce type d'interface de navigation est illustré par la figure 7.9.

Navigation 1.5D : C'est une extension de l'interface 1D, où les entités sont localisées par une abscisse curviligne x sur une courbe et par un décalage y perpendiculaire à la tangente à la courbe en x.

Navigation 2D : Les entités perçoivent le monde virtuel à travers sa projection sur un plan (en l'occurrence le sol) et l'utilisation d'un champ de perception 2D. Le lecteur intéressé pourra se référer à [Gaud et al., 2004] pour davantage de détails. Dans cette vue, une influence de mouvement correspond à la combinaison entre un vecteur et une rotation dans le plan. Le modèle de perception correspondant est illustré par la figure 7.10. Chaque piéton est doté d'une zone de perception composée de deux espaces : un triangle pour la perception lointaine et un cercle centré sur la position du piéton pour la vision périphérique.

Navigation 3D : Les entités perçoivent le monde en 3D via un champ de perception 3D. Une influence de mouvement correspond à la combinaison entre un vecteur et une rotation dans l'espace. Le modèle de perception 3D d'un piéton est illustré par la figure 7.11. Cette version correspond à une généralisation en 3D du cas précédent, les piétons sont dotés d'un frustum pour la perception lointaine et d'une sphère pour la vision périphérique.

Dans ces deux derniers cas (2D et 3D), la structure de l'environnement, utilisée pour calculer les perceptions des agents, est inspirée des structures utilisées en calcul graphique et correspond à un type particulier d'arborescence : une version adaptée des *Icospetree* [Shagam, 2003]. Davantage de détails sur les mécanismes de perception visuelle d'un agent, évoluant dans un environnement virtuel, peuvent être trouvés dans [Kuffner et al., 1999, Noser et al., 1995, Renault et al., 1990, Terzopoulos and Rabie, 1995, Wen and Mehdi, 2002].

Figure 7.9: Modèle de perception 1D d'un piéton

Ce mécanisme d'interface permet d'assurer une séparation claire entre les comportements du système simulé et de l'application, et les comportements liés à l'environnement. Ces différents types d'interface correspondent à des niveaux de réalisme de plus en plus importants, mais également de plus en plus coûteux en termes de gestion des perceptions et des actions. Les interfaces environnementales permettent de disposer de toute une palette de comportements applicatifs de complexité diverse et de les faire cohabiter au sein d'une même simulation. Elles introduisent une autre manière de moduler la complexité d'une simulation en changeant simplement la manière d'agir et de percevoir pour les entités du modèle.

7.4 Évaluer la cohérence de la simulation de piétons

Cette section est dédiée à la description des indicateurs d'aide au changement de niveaux utilisés dans l'application de simulation des piétons. Ils permettent d'évaluer la qua-

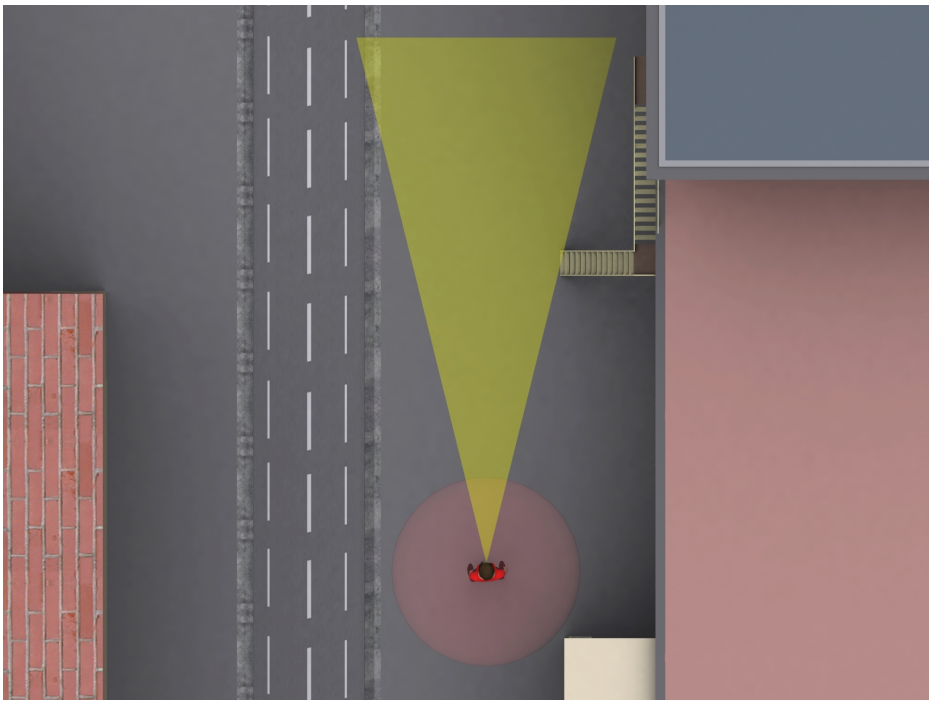


Figure 7.10: Modèle de perception 2D d'un piéton

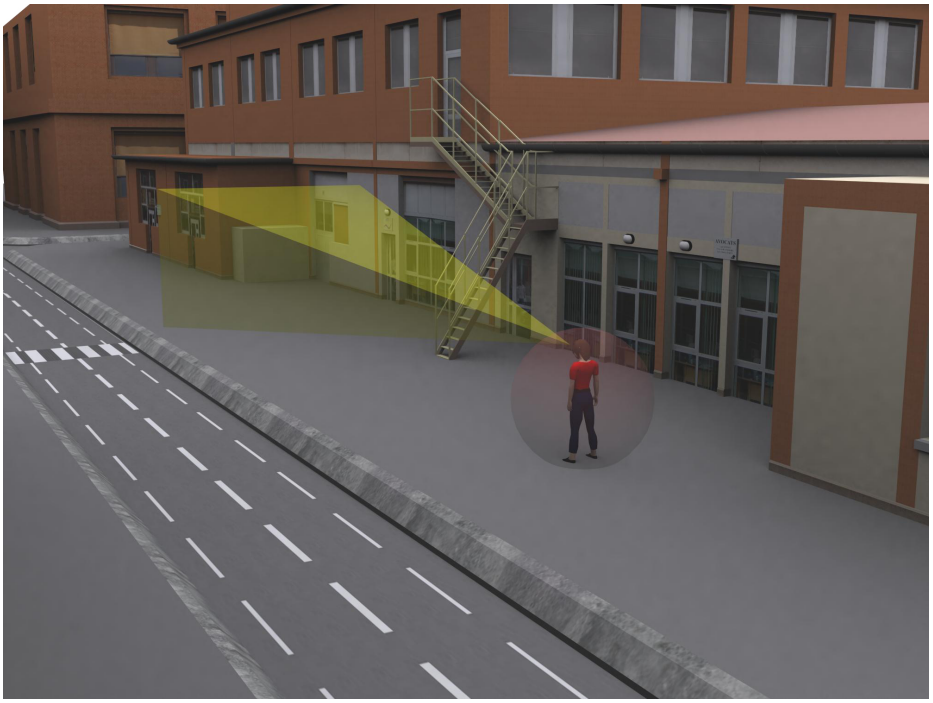


Figure 7.11: Modèle de perception 3D d'un piéton

lité de l'approximation fournie par un niveau d'abstraction plus élevé que le niveau microscopique. Ce dernier constitue le niveau de référence. En accord avec l'approche décrite au chapitre précédent (cf. section 6.6, page 174), l'ensemble de ces indicateurs est basé sur la notion d'énergie de sorte à éviter les inconvénients des méthodes basées sur l'entropie. En

effet, l'entropie ne peut pas être considérée comme une fonction d'état, et elle est principalement une mesure globale ne tenant pas compte des phénomènes locaux. Dans l'approche proposée, trois types d'énergie sont pris en compte : (i) l'énergie cinétique E_{c_i} liée à la vitesse du holon i considéré., (ii) l'énergie potentielle de l'objectif E_{pg_i} qui est dépendante de l'objectif du holon i, (iii) et l'énergie potentielle de contrainte E_{pc_i} qui est liée aux éléments qui entravent la progression du holon i vers son objectif (obstacles et autres holons). Sur la base de ces trois énergies, il est possible de calculer l'énergie globale E_k d'un holon k donné dans la holarchie : $E_k = E_{c_k} + E_{pg_k} + E_{pc_k}$. On peut considérer que cette énergie caractérise l'état courant du holon k.

L'évaluation de la qualité de l'approximation fournie par un niveau n dans la holarchie est basée sur la comparaison des énergies entre niveaux adjacents. Cette notion de similarité entre niveaux de simulation est définie par :

$$s_{n+1} = (\Delta E)_{n+1} = E_j^{n+1} - E_i^n$$

avec E_i^n l'énergie d'un holon i de niveau n, et E_j^{n+1} l'énergie de son super-holon j de niveau $n+1$.

En fonction des forces qui régissent le comportement des piétons (cf. section 7.2, 187), et les caractéristiques individuelles de ceux-ci, les trois précédentes énergies peuvent alors être définies telles que :

- L'**énergie cinétique** E_{c_i} :
$$E_{c_i} = \frac{1}{2}.m_i.(\overrightarrow{V_i}.\overrightarrow{V_i}) \tag{7.5}$$
avec m_i la masse du piéton i et $\overrightarrow{V_i}$ sa vitesse.
- L'**énergie potentielle** E_{pg_i} du piéton i, est calculée avec l'expression générale d'une énergie potentielle considérant une force conservative[3].
$$E_{pg_i} = -\delta W_{\overrightarrow{F_{obj}}} = -\overrightarrow{F_{obj}}.\overrightarrow{du} = -\frac{\beta_{obj}.\overrightarrow{A_iG}.\overrightarrow{V_i}}{\|\overrightarrow{V_i}\|} \tag{7.6}$$
avec $\delta W_{\overrightarrow{F_{obj}}}$ le travail[4] élémentaire de la force $\overrightarrow{F_{obj}}$, et $\overrightarrow{du}$ un vecteur unitaire[5].
- L'**énergie potentielle de contrainte** E_{pc_i} du piéton i, est calculée sur le même principe que la précédente en considérant la somme des travaux élémentaires des forces

[3]Une force conservative est une force dont le travail ne dépend pas du chemin suivi.

[4]Le travail d'une force est l'énergie fournie par cette force lorsque son point d'application se déplace.

[5]Pendant un pas de simulation dt, la force F_{obj} est supposée constante, un piéton ne se déplace que très peu, son déplacement peut alors être considéré comme unitaire et sa trajectoire comme rectiligne.

de répulsion des obstacles et des autres agents :

$$\begin{aligned} E_{pc_i} &= \sum_{k\in\{obstacles\}} \delta W_{\overrightarrow{F}_{rep_{ik}}} + \sum_{i\neq j} \delta W_{\overrightarrow{F}_{rep_{ij}}} \\ &= \sum_{k\in\{obstacles\}} \frac{\beta_{ik}.m_i.\overrightarrow{n_{ik}}.\overrightarrow{V_i}}{\|\overrightarrow{V_i}\|(d_{ik}.sin(\alpha_{ik}))^4} + \sum_{i\neq j} \frac{\beta_{ij}.m_i.m_j.\overrightarrow{d_{ij}}.\overrightarrow{V_i}}{\|\overrightarrow{V_i}\|\|\overrightarrow{d_{ij}}\|^2} \end{aligned} \tag{7.7}$$

L'analogie physique est relativement simple à appliquer dans le cas de simulations de piétons, car chaque piéton en mouvement peut être assimilé à un mobile. La formulation classique des énergies, utilisée en physique, peut être directement réutilisée. L'utilisation de la similarité permet de disposer d'un indicateur évaluant la qualité de l'approximation effectuée à un niveau supérieur de la holarchie. Il est évident que le calcul de ces énergies a un coût s'additionnant au coût de la simulation. Mais comme le montre la section suivante, il demeure relativement faible et l'information fournie pour l'adaptation dynamique des niveaux justifie sa mise en place.

7.5 Résultats expérimentaux

Les tests que nous avons réalisés quant à la simulation de piétons en environnement urbain virtuel présentée dans ce chapitre, ont été effectués sur un Pentium 4, 2.40 Ghz avec 512 Mo RAM, équipé d'une Geforce 3 Ti 200. Sur cette machine avec 212 piétons simulés, le taux de rafraîchissement de l'affichage 3D a été maintenu entre 18 et 23 images par seconde. Ce qui ne respecte pas tout à fait la contrainte de temps réel (25 images par seconde), mais qui s'en approche. Pour comparaison, [Musse and Thalmann, 2001] à l'aide d'un modèle hiérarchique de comportement de foules offre un taux de rafraîchissement de 20 images par seconde pour une population de 100 agents simulée sur un Onyx 2 (SGI). Pour davantage de détails sur la simulation de piétons en environnements urbains virtuels, le lecteur peut se référer aux travaux suivants : [Braun et al., 2003, Granieri et al., 1995, Hanson and Wernert, 1997, Jiang, 1999, Schelhorn et al., 1999, Shao and Terzopoulos, 2005, Xiao and Hubbold, 1998].

La figure 7.12 montre, malgré le coût de calcul des indicateurs énergétiques, que l'exécution de la simulation au niveau *macroscopique*, où les perceptions et les actions sont agrégées, est effectivement moins coûteuse que la simulation menée au niveau microscopique. Ces résultats confirment que le coût d'un niveau d'abstraction plus élevé avec un niveau de détail plus faible est effectivement moins important. Il est intéressant de souligner que le nombre de piétons considéré dans cet exemple est relativement faible, mais que la machine d'expérimentation l'est également. Sur un ordinateur plus performant, on peut supposer que davantage de piétons pourraient être simulés tout en conservant un taux de rafraîchissement au moins équivalent. Ce point nécessite cependant quelques approfondissements.

La figure 7.13 montre l'évolution de l'énergie d'un holon piéton simulé au niveau mi-

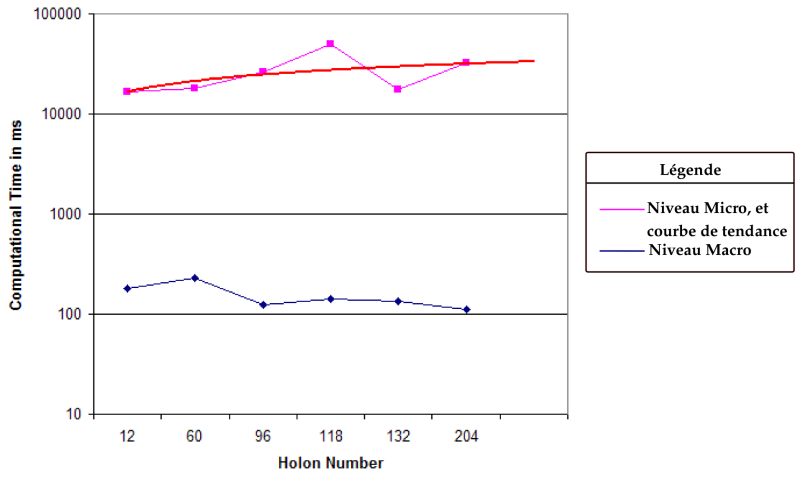


Figure 7.12: Évolution du coût de la simulation au niveau microscopique et macroscopique en fonction du nombre de piétons simulés

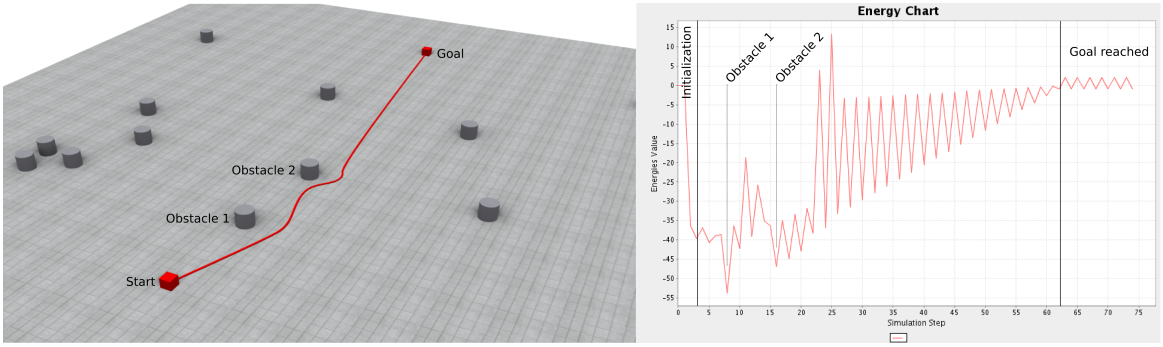


Figure 7.13: L'évolution de l'énergie d'un piéton simulé au niveau microscopique suivant le chemin décrit dans la partie gauche

croscopique et suivant le chemin décrit dans la partie gauche de la figure. Ces résultats révèlent que l'énergie du piéton oscille jusqu'à ce qu'il ait atteint son objectif. L'évolution moyenne de la courbe montre que l'énergie du piéton croit régulièrement au fur et à mesure de sa progression vers l'objectif. La figure 7.14 détaille l'évolution des énergies d'un groupe de 3 piétons simulés au niveau macroscopique, et suivant globalement le même chemin que le piéton seul. Au niveau macroscopique, les oscillations énergétiques ne sont plus présentes. Cette différence peut être expliquée par une compensation des oscillations locales par un déplacement "moyenné" du groupe et donc de ses membres. La courbe suit globalement une progression équivalente à la tendance moyenne de la courbe énergétique d'un piéton individuel. La différence entre l'énergie du super-holon (en jaune sur la figure 7.14) en charge du groupe et celle de ses membres (en rouge, bleu et vert) est relativement faible. L'approximation macroscopique réalisée ici peut être considérée comme acceptable. Lorsque le comportement agrégé du groupe ne satisfait plus celui d'un membre, cette différence augmente et alerte ainsi le super-holon que le membre en question requière un niveau

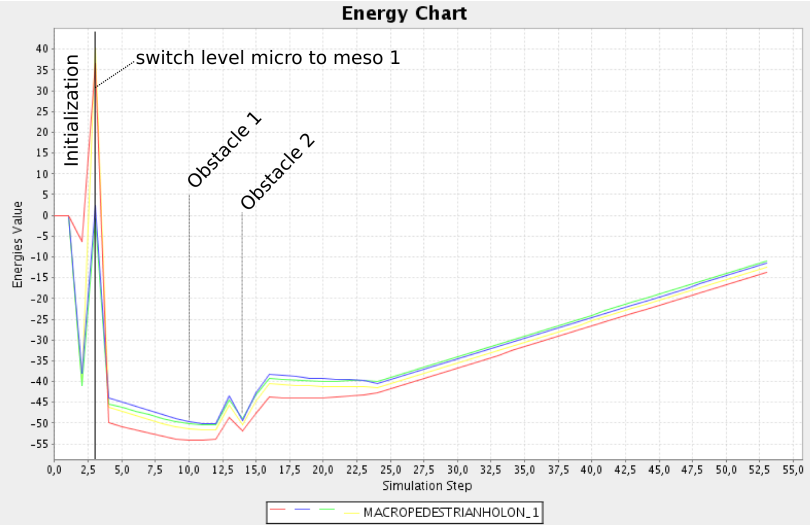


Figure 7.14: L'évolution de l'énergie de 3 piétons regroupés dans un super-holon en charge de les simuler au niveau macroscopique

de simulation plus fin ou un changement de groupe. Il est ainsi possible de déterminer le niveau d'abstraction optimal pour le comportement d'une entité en accord avec les autres contraintes de la simulation telles que le volume de ressources de calcul disponible. Ces résultats sont encourageants et viennent confirmer le potentiel de l'approche holonique pour la simulation multi-niveaux.

7.6 Conclusion

Ce chapitre décrit un ensemble de modèles multi-niveaux pour la simulation de piétons en environnement urbain virtuel. Cette simulation est décomposée en deux parties majeures, d'une part la simulation de la dynamique des piétons, et d'autre part celle de l'environnement urbain. Un modèle multi-niveaux est associé à chacune de ces composantes de la simulation, et une holarchie spécifique a été créée pour assurer l'exécution de chacune d'entre elles. Le comportement des piétons est modélisé à différents niveaux d'abstraction (individu, foule, population), et l'environnement urbain est abordé à plusieurs niveaux d'échelle (bâtiment, quartier, ville). La relation entre l'environnement et le système est modélisée par un ensemble d'interfaces environnementales, lesquelles permettent également à d'autres applications de s'interfacer avec l'environnement (interfaces graphiques, plates-formes de réalité virtuelle, etc.). Certaines de ces interfaces permettent notamment de fournir différents niveaux de complexité pour la perception et l'action des entités simulées. Elles fournissent ainsi un moyen supplémentaire de moduler la complexité d'une simulation en changeant simplement la précision des perceptions et des actions.

Le modèle d'exécution multi-niveaux est utilisé pour permettre la cohabitation de différents niveaux d'abstraction pour le comportement des piétons au sein d'une même simu-

lation et pour assurer une transition dynamique entre ces niveaux. Les indicateurs énergétiques, aisément calculables dans ce type d'application, permettent de garantir le meilleur compromis entre les ressources de calcul disponibles et le réalisme de la simulation. Ils facilitent l'identification des éléments clefs de la simulation qui nécessitent une allocation prioritaire des ressources. Le réalisme de la simulation est directement lié à la complexité des comportements des piétons ainsi qu'à la finesse de la décomposition environnementale et de sa représentation graphique.

Les résultats de cette application tendent à confirmer le bon fonctionnement des mécanismes d'exécution multi-niveaux et l'intérêt des systèmes multi-agents holoniques pour la modélisation de simulateurs et d'environnements urbains. Ces travaux nécessitent d'être approfondis, notamment sur les aspects concernant l'étude des transitions entre niveaux de simulation, et plus généralement l'établissement d'indicateurs statistiques permettant d'évaluer la cohérence d'une simulation. Les travaux futurs s'attacheront à enrichir le modèle de simulation urbaine par l'introduction de nouvelles entités, et notamment différents types de véhicules, tout en conservant les mécanismes de modélisation et d'ordonnancement multi-niveaux. L'objectif, à terme, consiste en effet à disposer d'un outil complet d'aide à la décision pour tester et valider différents scénarios de planification et d'aménagement au sein d'une ville virtuelle.

QUATRIÈME PARTIE

Conclusion et Perspectives

CHAPITRE 8

CONCLUSION

8.1 Bilan

Tout au long de cette thèse, nous avons tenté de fournir un ensemble d'abstractions et une méthodologie pour la modélisation, l'implantation et la simulation de systèmes complexes. Certains critères en matière de développement de logiciels ont guidé la réalisation de l'ensemble de ces travaux. La modularité et la réutilisation des modèles sont les deux exigences principales qui ont gouverné l'élaboration de nos propositions. Dans cette optique, l'approche organisationnelle a été largement adoptée, depuis l'analyse jusqu'à l'implantation. Ainsi, l'ensemble des abstractions proposées dans ce manuscrit repose essentiellement sur la combinaison des points de vue organisationnel et holonique. Cette section se propose d'examiner succinctement la démarche adoptée dans ce document.

Cette thèse commence par dresser un portrait de l'ingénierie logicielle orientée-agent et des standards qui la régissent. Quelques-unes des méthodologies orientées-agents existantes sont détaillées et comparées pour tenter d'appréhender leurs avantages et inconvénients. De cette comparaison émerge un ensemble de principes permettant de concevoir une méthodologie orientée-agent qui respecte les standards et les bonnes pratiques du domaine.

Sur la base de ces principes, le métamodèle CRIO est introduit pour l'analyse et la conception de systèmes multi-agents holoniques. Ce métamodèle adopte une approche dirigée par les modèles (*"Model Driven Development"*) et définit trois domaines, correspondant chacun à une étape du développement de logiciels. Le *domaine du problème* fournit les abstractions nécessaires pour collecter les connaissances liées au système, délimiter son périmètre, identifier les besoins à satisfaire et décomposer hiérarchiquement le système en fonction des besoins identifiés. Le cœur de ce domaine repose sur les concepts d'organisation et de rôle. L'approche organisationnelle est utilisée pour décomposer un système niveau par niveau. L'organisation représente un comportement global dont la complexité est répartie parmi un ensemble de rôles en interaction. De ce processus de décomposition découle la hiérarchie organisationnelle du système. Le second niveau du métamodèle, le *domaine agent*, décrit le modèle d'une société holonique d'agents en charge d'offrir une solution au problème précédemment modélisé. Le modèle du système à réaliser, incarné par la hiérarchie organisationnelle, est instancié et associé à un ensemble de holons pour

construire la ou les holarchies en charge de l'exécution du système. Le troisième et dernier domaine est consacré à l'implantation de l'application sur une plate-forme spécifique.

Nous avons ensuite introduit le processus de développement logiciel ASPECS, conçu pour fournir un guide méthodologique pour l'exploitation des abstractions fournies par le métamodèle CRIO. ASPECS est spécifiquement conçu pour le développement de systèmes logiciels complexes, et couvre l'intégralité du processus de développement, depuis l'analyse des besoins jusqu'au déploiement. ASPECS permet la modélisation d'un système à différents niveaux d'abstraction. Il favorise la réutilisation et la modularité des modèles et des connaissances. Dans cette perspective, les ontologies du problème et de la solution sont exploitées pour capitaliser les connaissances du domaine de l'application et de la solution considérée. L'organisation est considérée comme un module de conception à part entière, et la définition du comportement des rôles est basée sur la notion de capacité, permettant de s'abstraire de l'architecture des entités. Concernant l'environnement d'un système, ASPECS adopte une approche distribuée, où l'environnement est modélisé selon le point de vue des organisations qui le manipulent. Cette approche facilite la distribution des applications dans des environnements hétérogènes et distribués.

Pour mettre en œuvre les modèles résultats du processus ASPECS, nous avons proposé la plate-forme *Janus* destinée à l'implantation et au déploiement de systèmes multi-agents holoniques. Le métamodèle de cette plate-forme a été conçu de manière à réduire l'écart qui sépare l'analyse et la conception des SMA de leur implantation. *Janus* fournit ainsi une traduction aisée de l'ensemble des concepts utilisés pendant la phase de conception. Elle permet de considérer l'organisation comme un module Java à part entière. La gestion native du concept de capacité permet d'implanter un rôle, sans faire de supposition sur l'architecture des holons qui le joueront, et favorise ainsi la réutilisation des organisations dans diverses applications. L'implantation fidèle des concepts holoniques permet une implantation directe de la notion de holarchie.

La simulation des systèmes complexes constitue actuellement un enjeu majeur dans de nombreuses disciplines scientifiques. Les systèmes multi-agents revêtent d'ailleurs de plus en plus d'importance, dans ce domaine, pour leur capacité à aborder de tels systèmes. Dans ce contexte, nous avons montré comment exploiter les propriétés des systèmes multi-agents holoniques pour concevoir des modèles de simulation multi-niveaux. De tels modèles définissent les moyens nécessaires pour faire cohabiter, au sein d'une même simulation, différents niveaux d'abstraction pour les comportements d'un système, et pour assurer les transitions dynamiques entre ces niveaux. Les outils de gestion d'une simulation multi-niveaux reposent sur un modèle d'organisation, en charge de la gestion de l'ordonnancement et de la synchronisation des holons au sein de la simulation. Afin de faciliter la transition entre les différents niveaux d'abstraction, un ensemble d'indicateur est proposé pour évaluer l'écart entre deux niveaux de simulation adjacents, et ainsi déterminer le moment opportun pour réaliser la transition entre les niveaux. Ces outils participent à garantir le meilleur compro-

mis entre les contraintes d'exécution de la simulation et son niveau de précision.

L'intérêt des systèmes multi-agents holoniques pour la simulation a été exposé dans le cadre d'une simulation multi-niveaux de piétons en environnement urbain. Dans cette application, le modèle de l'environnement et du système cible sont clairement séparés. Un modèle multi-niveaux est associé à chacun de ces éléments, et une holarchie spécifique est créée pour assurer leur exécution. Le modèle d'exécution multi-niveaux est utilisé pour faire cohabiter différents niveaux d'abstraction pour le comportement des piétons, au sein d'une même simulation, et pour assurer une transition dynamique entre ces niveaux.

Les avantages principaux du processus de développement logiciel présenté dans cette thèse et des abstractions sous-jacentes peuvent être résumés comme suit :

- Réutilisation et modularité des modèles et des connaissances, en partie grâce à l'adoption de l'approche organisationnelle.
- Modélisation d'un système à plusieurs niveaux d'abstraction grâce à l'approche holonique.
- L'approche proposée permet d'exploiter la structure hiérarchique des systèmes complexes pour les analyser et les modéliser. L'approche holonique autorise ensuite la création de systèmes logiciels disposant de mécanismes complexes et fortement dynamiques pour la distribution des tâches, la prise de la décision , la coopération entre agents, la structure de l'application, etc.
- Respect de standards : UML, SPEM, FIPA, etc.
- Diminution du fossé séparant les phases de conception et d'implantation des systèmes multi-agents.

Ces travaux méthodologiques ne sont pas encore finalisés et plusieurs points devront être approfondis. La section suivante recense quelques-unes des améliorations possibles qui pourront faire l'objet de futures recherches.

8.2 Perspectives et travaux futurs

Deux axes majeurs peuvent être distingués pour guider les extensions des travaux présentés dans cette thèse : (i) La premier axe est méthodologique, et consiste à exploiter le potentiel des systèmes multi-agents holoniques pour la modélisation de systèmes. Pour se faire, le processus ASPECS devra être complété et sa qualité mesurée. L'objectif consiste à fournir une méthodologie complète ainsi que l'ensemble des notations et outils nécessaires pour supporter les activités de modélisation, depuis l'analyse des besoins jusqu'au déploiement. Ensuite, les différents fragments ou méthodes, qui composent la méthodologie, devront être extraits pour constituer une base de connaissances qui pourra s'intégrer dans une perspective de standardisation des méthodologies orientées-agent. (ii) Le second axe concerne la simulation multi-niveaux et l'enrichissement des modèles proposés pour aboutir au développement d'un simulateur multi-niveaux.

8.2.1 Vers une méthodologie complète

Extension des notations et méthodes formelles L'intégration des notations et méthodes formelles contribue directement à l'amélioration de la qualité du logiciel produit. Cette intégration des méthodes formelles peut concerner l'ensemble des phases du processus ASPECS.

En effet, une spécification formelle sur la base du multi-formalisme OZS (Object-Z et *"statechart"*) a déjà été proposée pour les aspects holoniques [Rodriguez et al., 2005a] et pour les concepts de rôle, d'interaction et d'organisation [Hilaire et al., 2000]. La spécification formelle des concepts du métamodèle CRIO devra être poursuivie, et certaines activités du processus devront être raffinées, pour intégrer les étapes de définition formelle de ces concepts. L'intégration des méthodes formelles permettra d'enrichir les phases de tests, en permettant notamment une génération automatique d'une partie des cas de test (*"Test case"*). Les procédures permettant de générer des modèles de code pour les rôles à partir des spécifications OZS sont en cours de développement.

Outre les langages formels, ASPECS devra également poursuivre son intégration dans les standards qui régissent le génie logiciel et notamment les standards en terme de langage de modélisation. Ce respect des standards est déjà l'un des points clefs du processus ASPECS, comme en témoigne d'ailleurs l'intégration d'UML et des spécifications de la FIPA.

Pour supporter et faciliter le processus de conception, une méthodologie seule peut difficilement satisfaire pleinement les attentes industrielles. Cette méthodologie devra nécessairement être associée à un ensemble d'outils destinés à faciliter le travail du concepteur et du développeur. La plate-forme *Janus* ainsi que l'Atelier de Génie Logiciel *Janeiro* s'intègrent dans cette perspective, mais leur développement devra être enrichi.

Évaluation de la qualité du processus Convaincre les développeurs et le secteur industriel de l'intérêt d'un processus méthodologique passe également par l'évaluation de sa qualité. ASPECS devra, à terme, se confronter aux modèles de référence qui permettent d'évaluer le développement logiciel et d'évaluer sa qualité, telles que CMMI[1]. Cette évaluation entraînera probablement des changements notoires dans le processus, et nécessitera la définition et l'intégration de nouvelles activités et de nouveaux rôles, davantage liés à la gestion de projet.

[1] CMMI : Capability Maturity Model Integration

8.2.2 Vers une intégration dans les standards méthodologiques

De nombreux travaux de recherche, notamment au sein des groupes de travail AOSE d'AgentLink[2] et de la FIPA[3], se concentrent sur l'unification des méthodologies multi-agents et sur le développement de méta-méthodologies. L'objectif de l'approche méta-méthodologique vise essentiellement à définir les abstractions nécessaires pour créer de nouvelles méthodologies. Cette approche repose notamment sur la constitution d'une base de connaissances intégrant les méthodes et fragments des méthodologies existantes. Cette base peut ensuite être exploitée pour assembler des fragments de méthodologies existantes pour concevoir une nouvelle méthodologie adaptée à un domaine cible, voire à un problème particulier. L'une des perspectives des travaux proposés vise à intégrer certains fragments de l'approche ASPECS au sein de cette base, et à orienter le développement des outils tels que Janeiro pour faciliter leur intégration dans ce méta-processus et favoriser leur interopérabilité[4].

8.2.3 Vers un simulateur multi-niveaux générique

En sus de la modélisation des systèmes complexes, cette thèse aborde également les problématiques liées à leur simulation. Les systèmes multi-agents holoniques sont utilisés pour gérer la mise en œuvre des mécanismes multi-niveaux au sein d'une simulation multi-agents. Les outils et modèles, introduits au chapitre 6, présentent les conclusions d'une expérience menée dans le cadre du développement de la simulation multi-niveaux de piétons en environnement virtuel. Des efforts sont nécessaires pour l'extension et la généralisation de ces conclusions en vue d'enrichir la plate-forme *Janus* par un simulateur multi-niveaux.

Par ailleurs, la création de nouveaux indicateurs et outils pour l'évaluation de la cohérence d'une simulation est nécessaire. En effet, l'approche multi-niveaux repose, en grande partie, sur la capacité à évaluer l'impact de l'action d'un individu sur le système et sur les autres individus, et inversement. L'adaptation des modèles inspirés d'autres domaines, et notamment de la physique, semble prometteuse. La physique statistique offre une palette importante d'outils qui pourront, par analogie, être transposés au domaine des SMA (ex : la fonction Z).

Ce problème peut d'ailleurs être élargi à la problématique de l'évaluation de la précision, de l'exactitude ou de l'efficacité d'un système multi-agents, relativement à la tâche qu'il doit effectuer et aux mécanismes locaux impliqués dans l'accomplissement de cette tâche.

---

[2] *"AgentLink AOSE Technical Forum Group"* : `http://www.pa.icar.cnr.it/cossentino/al3tf3/`

[3] *"FIPA Methodology Technical Committee"* : `http://www.fipa.org/activities/methodology.html`

[4] Notamment avec les outils existants tels que MetaMeth [Caico et al., 2006, Cossentino et al., 2006a], Java Workflow Editor (JaWE) : `http://www.enhydra.org/workflow/jawe/index.html` et Protégé : `http://protege.stanford.edu/`.

CINQUIÈME PARTIE

Annexes

ANNEXE A

LA PLATE-FORME D'IMPLANTATION JANUS

Ce chapitre complète le chapitre 5 en présentant le modèle UML complet du noyau de la plate-forme *Janus* et l'implantation du problème des enchères.

A.1 Modèle UML complet du noyau de la plate-forme *Janus*

Le modèle UML du noyau de la plate-forme *Janus* est présenté à la figure A.1.

A.2 Exemple : le problème des enchères

Le problème des enchères (souvent nommé *"Market"* ou *"Trading"*) est un exemple classique (également détaillé dans Madkit, cf. package *marketorg*). Cet exemple permet d'expliciter concrètement les spécificités liées à l'implantation des différents concepts du modèle du domaine de la solution de CRIO. L'ensemble des organisations et des rôles nécessaires à l'implantation de cet exemple est décrit à la figure A.2. Le modèle des enchères est ici appliqué à la commande de voyages. Un client qui souhaite obtenir la meilleure offre de voyage disponible soit en terme de prix, soit en terme de temps, formule sa proposition et l'envoie au *CBroker*. Ce dernier transmet l'information au *PBroker* qui publie l'offre aux différents *Provider* disponibles. En fonction du critère de choix du client (temps ou prix), il détermine la meilleure proposition et en informe le *Client*. Le *Client* et le meilleur *Provider* créent ensuite une instance de l'organisation de contractualisation pour finaliser la commande et effectuer le paiement.

Concrètement, l'implantation proposée de cet exemple nécessite la définition :

– de trois architectures de holons : le *Client*, le *Provider*, le *Broker* ;
– de trois organisations : *Client*, *Provider* et *Contract* ;
– et des six rôles définis par ces organisations (deux par organisation).

Chaque organisation dispose de son propre paquet Java contenant les classes de ses rôles et sa propre classe. Chaque architecture de holon est également implantée dans une

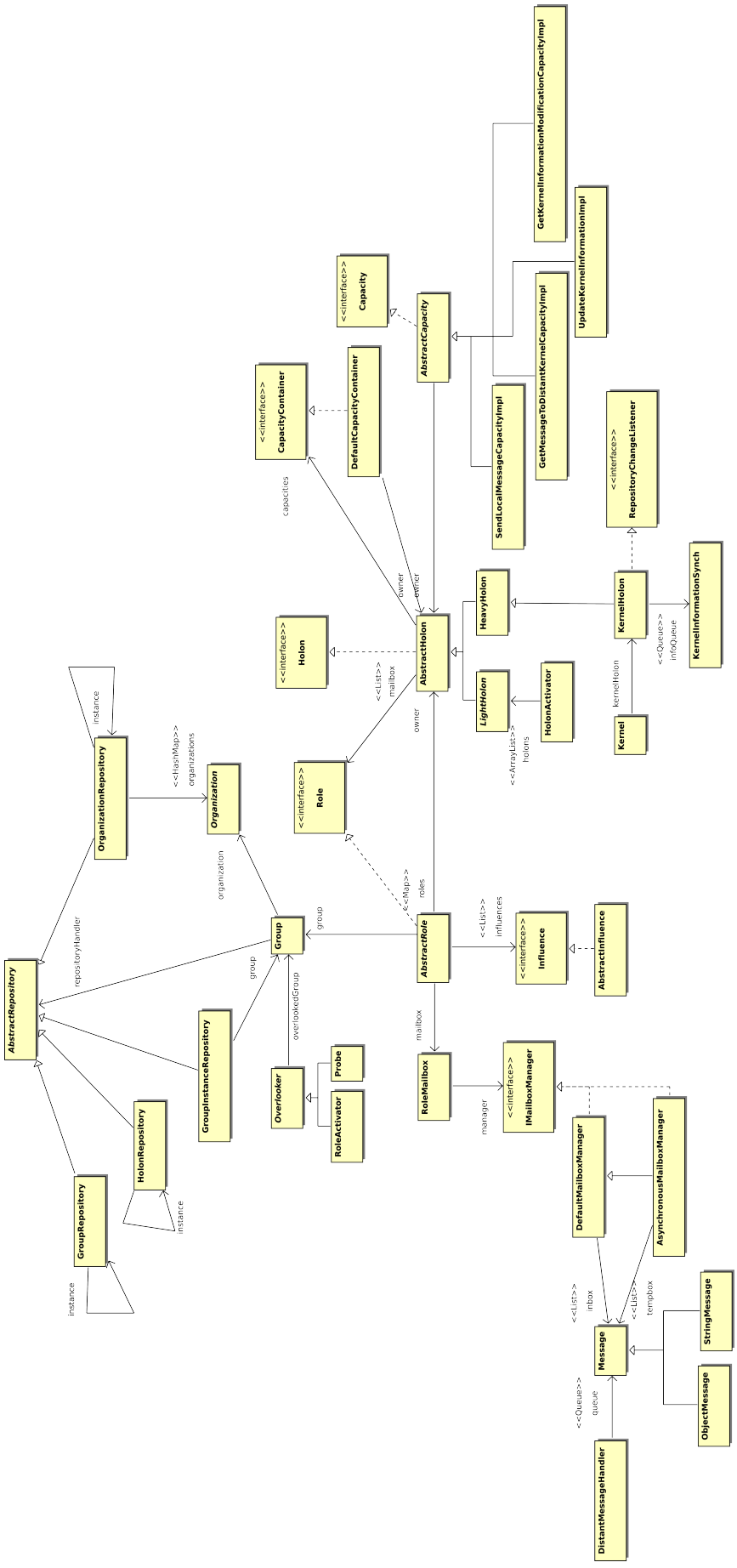


Figure A.1: Diagramme UML du noyau de *Janus*

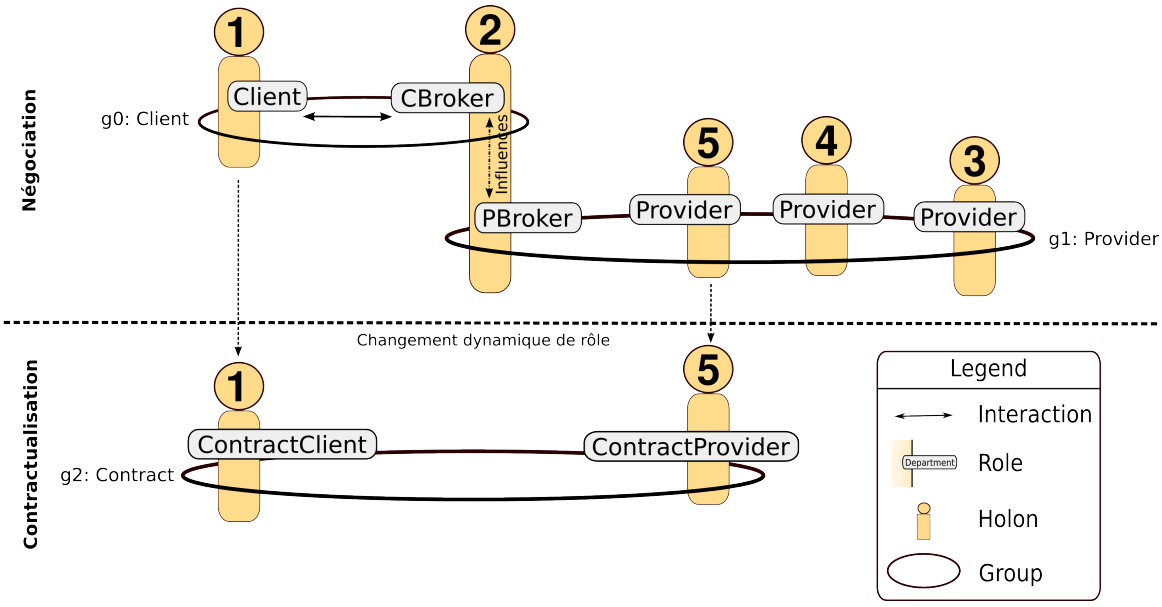


Figure A.2: Structure concrète du système de négociation et de contractualisation

classe indépendante. Dans la suite de ce document, l'implantation de deux rôles est plus particulièrement explicitée : le rôle *Client* et le rôle *PBroker*.

Dans l'implantation proposée, le rôle *PBroker* requiert deux capacités : *FindShortestTimeProposalCapacity* et *FindLowestCostProposalCapacity*. Ces capacités sont utilisées pour déterminer la meilleure proposition parmi celles proposées par les différents *Providers*. Cette détermination s'effectue en fonction du critère choisi par *Client*. Si le critère de choix est temps, la capacité *FindShortestTimeProposalCapacity* sera utilisée, si le critère est coût, c'est la capacité *FindLowestCostProposalCapacity*.

De plus le rôle *PBroker* impose à l'agent qui le joue, de posséder au préalable le rôle *CBroker*, car ce dernier est indispensable pour assurer le transfert d'information entre les deux organisations. C'est un exemple typique de dépendances entre rôles. Ces contraintes de capacités et de dépendances entre rôles sont implantées grâce à la notion de condition d'obtention d'un rôle (*ObtainConditions*).

Considérons désormais le code source du rôle *PBroker* de sorte à expliciter ces différents aspects.

```
1 public class PBroker extends AbstractRole {
2 // ... Attributs du rôle ...
3    private int current = 1; // the current state
4    public PBroker() {
5       super();
6       //Définition des dépendances du rôle
7       List<Class<? extends Role>> requiredRoles = new LinkedList<Class<?
            extends Role>>();
8       requiredRoles.add(CBroker.class);
```

```
9       SatisfyRoleDependenciesCondition roleCondi = new
            SatisfyRoleDependenciesCondition(requiredRoles);
10
11      //Définition des capacités requises par le rôle
12      List<Class<? extends Capacity>> requiredCapacities = new
            LinkedList<Class<? extends Capacity>>();
13      requiredCapacities.add(FindShortestTimeProposalCapacity.class);
14      requiredCapacities.add(FindLowestCostProposalCapacity.class);
15      HasAllRequiredCapacitiesCondition capCondi = new
            HasAllRequiredCapacitiesCondition(requiredCapacities);
16
17      //Ajout des conditions d'obtention
18      addObtainCondition(capCondi);
19      addObtainCondition(roleCondi);
20   }
21 //Le coeur du comportement du role.
22   public void behavior() {
23      current = Run();
24   }
25
26 //Cette méthode correspond à la traduction en java
27 //du statechart correspondant au comportement du rôle
28 //Ce statechart est directement issu de la phase de conception
29   private int Run() {
30      switch (current) {
31 //Le rôle s'enregistre pour signifier qu'il souhaite recevoir
32 //un type particulier d'influence, en l'occurence celles provenant
33 //du rôle CBroker d'où la dépendance entre ces rôles
34      case 1 : registerForRoleInfluence(TravelRequestInfluence.class);
35         return 2;
36 //Attente de l'arrivée de l'influence provenant du rôle CBroker
37      case 2 : influence = (TravelRequestInfluence)getNextInfluence();
38         if (influence != null) return 3;
39         return 2;
40 // ...
41
42      case 4 :providers = getRolePlayers(Provider.class, getGroup());
43         WaitedMsgNb = providers.size();
44         if (WaitedMsgNb > 0) {
45 //Être sur qu'au moins provider est present avec broadcast
46 //de l'appel d'offre
47            broadcastMessage(Provider.class, new
                  CallForTravelProposalMessage(requestType));
48            return 5;
49         }
50         return 4;
51 //Récuperation du prochain message de la boîte à lettre du rôle
```

```
52        case 5 :m = getNextMessage();
53  //...
54        case 6 :println("PBroker got a proposal of all providers, nb of
              provider "+ providers.size()+"\n List of Proposals : \n");
55  // ...
56           List input = new ArrayList();
57           input.add(proposalList);
58
59           if (requestType == TravelRequestType.LowestCost) {
60              selected = FindLowestCostProposalCapacity.class;
61  //Exécution de la capacité FindLowestCostProposalCapacity
62              callAndExecuteCapacity(selected, id, input);
63           } else if (requestType == TravelRequestType.ShostestTime) {
64              selected = FindShortestTimeProposalCapacity.class;
65  //Exécution de la capacité FindShortestTimeProposalCapacity
66              callAndExecuteCapacity(selected, id, input);
67           } else println("Error in Request Type");
68           return 7;
69  //Attente du résultat de l'exécution de la capacité
70        case 7 :if (isResultAvailable(selected, id)) {
71           output = getResult(selected, id);
72           return 8;
73           }
74        return 7;
75  //Informe le meilleur Provider qu'il a été choisi
76        case 8 :Best = (Proposal)output.get(0);
77           BestProvider = proposals.get(Best);
78           sendMessage(BestProvider, Provider.class, new StringMessage("
                Proposal Accepted"));
79           return 9;
80
81        case 9 :m = getNextMessage();
82           if ((m != null) && (m instanceof TransfertMessage)) {
83              return 10;
84           }
85           return 9;
86  //Le rôle émet une influence et informe ainsi tous les rôles
87  //à l'écoute de ce type d'influence (CBroker)
88        case 10 : influenceHolon(new TransfertInfluence(((TransfertMessage
              )m).getGroupAddress()));
89           println("PBroker Finish");
90           return 0;
91        default : return 0;
92        }
93     }
94  }
```

Le rôle *Client* présente l'intérêt de mettre en valeur la dynamique des rôles. En effet une fois le meilleur des *Providers* déterminé, ce dernier et le client se rejoignent dans un nouveau groupe pour finaliser la commande. Ceci impose qu'en cours d'exécution, les holons *Client* et *Provider* accèdent à de nouveaux rôles, et que le *Client* libère également son précédent rôle devenu inutile. Le code source d'un holon est réduit aux seuls rôles qu'il joue et aux seules capacités qu'il possède. Le code source du rôle *Client* est détaillé ci-dessous.

```
1  public class Client extends AbstractRole {
2  // Attributs du rôle et méthode behavior()
3
4     private int Run() {
5        switch (current) {
6  //Envoi d'un message au rôle CBroker
7        case 1 :mybroker = sendMessage(CBroker.class,new
             TravelRequestMessage(TravelRequestType.LowestCost));
8           if (mybroker != null) return 2;
9              return 1;
10       case 2 :println("MyBroker: "+mybroker);
11          return 3;
12 //Récupération du premier message dans la boîte à lettre du rôle
13       case 3 :m = getNextMessage();
14          if (m != null) return 4;
15          return 3;
16       case 4 :println("I got a message "+m.toString());
17          return 5;
18       case 5 :
19 //Demande d'accès au rôle ContractClient
20 //C'est ce dernier qui se chargera de libérer le rôle Client
21    if(requestRole(ContractClient.class,(GroupAddress)((ObjectMessage)m).
          getContent())) return 6;
22          return 5;
23       case 6 :println("role ContractClient assigned");
24          return 7;
25 //Emission d'une influence dans le contexte du holon
26 // à destination rôle ContractClient
27       case 7 : influenceHolon(new TransfertInfluence(getGroup()));
28          println("Client Finish");
29          return 7;
30 //Ne fais plus rien jusqu'à ce que le rôle soit effectivement libéré.
31       default : return 0;
32       }
33    }
34 }
```

ANNEXE B

PLATES-FORMES ET OUTILS MULTI-AGENTS

Cette annexe résume quelques-unes des plates-formes multi-agents existantes et celles qui firent l'objet d'une étude avant la conception de *Janus*. Une liste plus complète des plates-formes multi-agents existantes peut être trouvée aux adresses suivantes :

- `http://www.agentlink.org/resources/agent-software.php`
- `http://dsonline.computer.org`

Un descriptif complet des dernières avancées en termes de langages de programmation et de plates-formes multi-agents peut être trouvé dans [Bordini et al., 2006].

B.1 Langage de programmation agent

Concurrent METATEM [Fisher and Wooldridge, 1997] est un langage de programmation multi-agents basée sur une sémantique très proche de la logique temporelle. Un système basé METATEM contient un certain nombre d'agents exécutés de manière concurrente, et communiquant par l'intermédiaire de message asynchrone (en *"broadcast"*). Le comportement de chaque agent est programmé à l'aide d'une spécification en logique temporelle. METATEM permet également de prouver que l'exécution de la spécification d'un agent est correcte et conforme à la spécification initiale.

Jack [1] [Hodgson et al., 1999, Howden et al., 2001]. Le langage agent Jack est un langage de programmation qui étend Java avec les concepts agents tel que agent, capacité, événements, plans et introduit les mécanismes de gestion des ressources et de l'exécution concurrente. L'architecture des agents JACK est basée sur l'approche BDI. Jack intègre un ensemble d'outils graphiques pour l'analyse et la programmation.

Dima [2] [Guessoum, 1998, Guessoum and Briot, 1999] - Développement et Implantation de Systèmes Multi-Agents - L'architecture d'agents DIMA propose de décomposer chaque agent en différents modules, chacun représentant les différents comportements d'un agent

[1] Jack : `http://www.agent-software.co.uk`
[2] Dima : `http://www-poleia.lip6.fr/~guessoum/dima.html`

tels que la perception, la communication et la délibération. La représentation des agents est fondée sur des mécanismes déclaratifs ainsi le mécanisme de contrôle d'un agent peut être décrit par un automate et son interpréteur. La première version de DIMA était en Smalltalk-80, elle fut portée ensuite en JAVA. Une version répartie appelée DARX (Dima Agent Replication Extension) existe également.

Jason [3] [Bordini et al., 2005] est un interprèteur et un langage orienté-agent. C'est une extension du langage AgentSpeak [Rao, 1996]. Comme son prédécesseur, il est principalement dédié à la conception d'agents BDI. Il gère les communications à base d'actes de langage et peut s'interfacer avec d'autres plates-formes telles que Jade par exemple. Il fournit également une implantation du modèle organisationnel MOISE [Hannoun et al., 2000] (Rôles, Groupe, et Mission).

Oris [4] [Ballet et al., 1998, Harrouet, 2000] est un langage initialement créé pour développer un simulateur de réalité virtuelle à base d'entité autonome. Il est orienté objet, et dispose d'une syntaxe proche du C++. Il gère les environnements multi-tâches et propose une interface basique. Son objectif est clairement destiné à un type application donné plutôt dans le domaine de la simulation en environnement virtuel.

CLAIM [Seghrouchni and Suna., 2004] - Compuational Language for Autonomous Agent, Intelligent and Mobile Agents - est un langage déclaratif de haut niveau, partie intégrante du framework Himalaya (Hierarchical Intelligent Mobile Agents for building Large-scale and Adaptive sYstems based on Ambients). CLAIM est organisé de manière hiérarchique et gère les aspects relatifs à la mobilité, l'intentionnalité et la communication des agents.

B.2 Plates-formes pour agents cognitifs, communiquants, web ou mobiles

Synergic [Carpuat and Regis, 1995] L'architecture des agents y est plutôt cognitive. Chaque agent dispose de croyances sur son environnement représentées par des accointances. Les agents ne raisonnent pas sur l'organisation du SMA et n'en ont aucune représentation. Synergic était développée en C et n'est plus maintenue depuis 1993.

OSACA [Scalabrin and Barthès, 1997, Shen and Barthès, 1996] - Open System of Asynchronous Cognitive Agents - résulte de la combinaison des agents avec les architectures orienté-objet de type CORBA. OSACA est un environnement qui gère les systèmes ouverts

[3] Jason :`http://jason.sourceforge.net/`
[4] Oris :`http://portail.langage.oris.free.fr/`

et fournit un langage pour créer aisément des agents (LAG). Les agents sont cognitifs et développés par clonage de l'agent générique puis dotés de compétences. Le langage de programmation était initialement Lisp puis C ou C++. Des extensions ont ensuite été proposées tel que SMAS pour la simulation de SMA complexes et OMAS pour l'exécution temps réel.

Mocah [Abchiche, 1994, 1999] - MOdélisation de la Coopération entre Agents Hétérogènes - est un modèle de coopération permettant à plusieurs modèles de raisonnement hétérogènes de coopérer pour résoudre un problème commun. Chaque modèle de raisonnement est représenté par un agent particulier et manipule un type de connaissances particulier (le domaine) et un mode de raisonnement (les méthodes) approprié à ces connaissances. Chaque agent est caractérisé par un comportement coopératif. Cette plate-forme a été conçue pour traiter les problèmes de diagnostic de pannes électriques.

GTMas [Chevrier, 1993, 1994] - General Toolkit for MAS - offre un ensemble de modules indépendants où l'utilisateur puise les fonctionnalités qui lui sont nécessaires : représentation des agents, communication (point à point, *"broadcast"*, ou tableau noir), etc. Dans Gtmas un agent correspond à un processus, leur exécution est donc concurrente. L'outil propose également la possibilité d'exploiter les connaissances d'un agent sous forme de règles de production. Gtmas propose également un ensemble d'outils pour faciliter le développement tel que : sélection des informations utiles, leur visualisation après exécution et des statistiques, etc. Une extension de Gtmas nommée *Javama*, intégrant un modèle de réorganisation automatique a été proposé par Foisel [1998a,b].

Jatlite [5] [Jeon et al., 2000] - Java Agent Template, Lite - Destinée aux agent Web, Jatlite offre un ensemble de librairie Java pour faciliter la création de SMA communiquant sur internet pour effectuer des calculs distribués. Le cœur de Jatlite repose sur un type particulier de routeur agentifié : agent Message Router (AMR), permettant une connexion sécurisée entre les différents nœuds. La communication (basé sur KQML) peut être asynchrone et intermittente (tolérance aux fautes).

Mercure [Briffault et al., 2000] est une plate-forme générique globalement conforme aux spécifications architecturales proposées de la FIPA. Elle est développée en Smalltalk. Les communications entre agents sont basées sur KQML.

Hive [6] [Minar et al., 1999] est une plate-forme Java pour distribuer des applications dynamiques sur le Web. Elle intègre les agents mobiles et l'approche ad-hoc ainsi que la création dynamique d'ontologie sur les capacités des agents.

[5] Jatlite : `http://java.stanford.edu/`
[6] Hive : `http://hive.sourceforge.net`

GrassHopper [7] [Bäumer and Magedanz, 1999, Bäumer et al., 2000] est une plate-forme java pour Agent Mobile. elle est conforme au standard MAF (Mobile Agent Facility, successeur de MASIF) de l'OMG et aux spécifications FIPA. GrassHopper étend la machine virtuelle java (jvm), lui intégrant les mécanismes de gestion étendue des communications, de gestion du cycle de vie des agents et de migration des agents. Les services d'annuaire et de référencement sont également disponibles. Cette plate-forme supporte un grand nombre de standards de communications tel que Corba (Common Object Request Broker Architecture), IIOP (Internet Inter-ORB Protocol), Java RMI (Remote Method Invocation), ou les sockets traditionnelles.

Aglet [8] [Clements et al., 1997, Lange and Mitsuru, 1998] est originellement une extension des Applets Java développée par IBM. On peut la considérer comme une plate-forme pour agents Web. Un aglet est un objet java actif qui peut migrer sur internet entre les différents serveurs (Gestion du cycle et des interruptions d'exécution). Différents protocoles de transport sont disponibles notamment RPC, RMI et Corba. L'idée de base de la plate-forme est donc le code mobile. La plate-forme fournit un environnement graphique pour développer les applications d'agents mobiles ainsi qu'un serveur agent. La sécurité et l'intégrité du code sont assurées lors de la migration et de l'exécution. La mobilité est assurée grâce à un protocole de transfert d'agents (ATP) et à l'interface de communication et de transfert d'agents (J-ATCI).

Able [9] [Bigus et al., 2002] - Agent Building and Learning Environment - fournit un ensemble complet de bibliothèques Java pour le développement d'agents cognitifs dotés de mécanismes d'apprentissage. Des éditeurs permettent de créer des agents : les AbleBeans, en combinant des composants JavaBeans. Able est la technologie IBM qui succède à Aglet.

AgentBuilder [10] [Reticular, 1999] est une plate-forme commerciale développée en Java permettant de construire des agents intelligents. Le comportement d'un agent est décrit grâce au langage AGENT-0. L'architecture Agent est basée sur l'approche BDI. Les agents communiquent par messages KQML. AgentBuilder est composé d'un toolkit qui fournit un environnement intégré pour assister le processus de développement des agents et d'un système d'exécution qui fournit l'environnement d'exécution des agents.

Jade [11] [Bellifemine et al., 2001] - Java Agent DEvelopment framework - est un middleware développé en java. Il est conforme aux spécifications FIPA et gère donc le cycle de vie

[7] GrassHopper : `http://cordis.europa.eu/infowin/acts/analysys/products/thematic/agents/ch4/ch4.htm`
[8] Aglet : `http://www.trl.ibm.com/aglets/`
[9] Able : `http://www.alphaworks.ibm.com/tech/able`
[10] AgentBuilder : `http://www.agentBuilder.com`
[11] Jade :`http://jade.cselt.it`

des agents, les services d'annuaire (pages blanches et jaunes), mais également la migration des agents. Il est basé sur une architecture peer-to-peer et permet la distribution des applications et la configuration dynamique et à distance des différents nœuds de déploiement.

B.3 Plates-formes de simulation

Swarm [12] [Minar et al., 1996] est l'outil privilégié de la communauté américaine et des chercheurs en Vie Artificielle. Swarm est disponible en Java ou en Objective C. Swarm est une plate-forme destinée à la simulation des systèmes complexes adaptatifs. Un système Swarm est composé d'unités basiques appelées Swarm qui représentent une collection d'agents. Swarm gère la modélisation hiérarchique et holoniques des systèmes. Swarm fournit un ensemble de bibliothèques orientées-objet pour la construction, l'affichage, l'observation et le contrôle des simulations.

Cormas [Bousquet et al., 1998] est un environnement de programmation permettant la construction de modèles de simulation multi-agents. Il fut conçu en particulier pour simuler des écosystèmes ou des problèmes de gestion des ressources communes. Il est clairement inspiré de Swarm, et développé en SmallTalk. L'espace est géré par un automate cellulaire où chaque cellule peut contenir un autre automate. Cette plate-forme offre des outils pour définir les agents et leurs interactions, contrôler la dynamique globale de l'environnement et observer la simulation (liens d'accointance, proximité interactionnelle, etc).

Mobidyc [13] [Ginot and Le Page, 1998, Ginot et al., 2002] est une plate-forme de simulation spécialisée pour les domaines de l'écologie, de la biologie et de l'environnement. elle dispose d'outils pour la création et l'utilisation de modèles individus-centrés ainsi qu'un module dédié à l'étude statistique de plans d'expériences simulatoires raisonnés. Comme dans Cormas, l'environnement est considéré comme discret et il est géré par un automate cellulaire.

B.4 Plates-formes pour agent à base de composants

Maleva [Lhuillier, 1998] - Modular Architecture for Living and EVolving Agents - est une plate-forme développée à base de composants en Borland Delphi. Dans cette plate-forme, un agent est considéré comme un composant, lequel est lui-même construit par composition de différents composants. Un mécanisme de communication synchrone/asynchrone entre ces composants est assuré par un gestionnaire de message dédié. Les communications inter-agents fonctionnent sur le même principe que celles inter-composants.

[12] Swarm : `http://swarm.org/wiki/Main_Page`
[13] Mobidyc :`http://www.avignon.inra.fr/mobidyc/version_index_html`

Maleva comprend des outils pour la conception des composants de base et des agents par composition fonctionnelle et structurelle. Un ensemble d'outils pour simuler les agents dans des environnements topologiques est également proposé.

Comet [14] [Peschanski, 1999, 2000] tout comme Maleva, Comet est une plate-forme basée sur la notion de composant, mais le principe de communication est différent. Les composants communiquent ici par événements asynchrones. Une formalisation du modèle opérationnel de COMET a été réalisée à l'aide des réseaux de pétri et du langage Z. Une adaptation de la plate-forme DIMA intégrant les concepts de COMET a également été réalisée.

MASK [Occello et al., 2004] - Multi-Agent System Kernel - Projet complet à part entière également, MASK est basée sur l'approche de Conception Voyelle (AEIO) [Demazeau, 1997]) et vise à fournir un ensemble de bibliothèques d'Agents (Hybrides et Réactifs), de manipulation d'Environnement (E), d'Interaction (Langages à protocoles et Forces) (I) et d'Organisation(O) ainsi que les outils d'aide à la programmation. Pour chaque élément, MASK fournit des éditeurs qui permettent d'affiner le SMA de manière déclarative.

Volcano [Ricordel, 2001, Ricordel and Demazeau, 2002a,b] est une plate-forme basée sur la méthodologie de développement Voyelle (AEIO) [Demazeau, 1997]. Dans la continuité de MASK, Volcano couvre les différentes briques définies dans Voyelle. C'est une plate-forme orientée composant et elle utilise son propre langage de description d'architecture *Madel* (Multi-Agent DEscription Language) pour décrire les composants et leurs interactions.

B.5 Plates-formes complètes, génériques et généralistes

MACE [Gasser, 1988, Gasser et al., 1987] fut l'un des premiers environnements généraux de modélisation pour SMA (indépendant du domaine d'application). Il introduit l'idée que les agents peuvent être utilisés dans tous les aspects de la construction de modèle (analyse et conception) et du développement. Dans MACE, un agent est un objet actif qui communique par envoi de messages. Un agent dispose de trois types d'actions : modifier son état interne, envoyer des messages aux autres agents ou envoyer des requêtes au noyau pour contrôler les événements internes. MACE utilise un système d'exécution concurrente et introduit déjà les concepts liés à la composition récursive d'agents : Un groupe d'agents peut être considéré comme un agent. Les concepts liés à l'auto-organisation au sein d'un SMA sont également abordés.

[14] Comet : `http://www-oasis.lip6.fr/~pesch/data/phdwork.html`

Geamas [Marcenac, 1997, Marcenac and Giroux, 1998, Soulie et al., 1998] - GEneric Architecture for MultiAgent Simulation - est une plate-forme logicielle générique pour la modélisation et la simulation multi-agents, implantée en Java. L'architecture logicielle de la plate-forme s'appuie sur un micro-noyau générique *JAAFAAR* offrant les structures et mécanismes minimaux nécessaires à l'implantation de SMA. À ce dernier, un certain nombre d'extensions logicielles spécialisées ont été adjointes tel que des modules d'apprentissage, d'auto-organisation ou de conception assistée, etc. Geamas gère trois niveaux d'abstraction : niveau micro (réactifs), médium (cognitifs, groupe) et macro (société). Gameas propose également un environnement intégré et des outils d'observation des applications.

Magique [15] [Bensaid and Mathieu, 1997a,b] - Multi-AGent hiérarchIQUE - est un framework java de construction d'applications multi-agents. Elle fournit le support des notions de compétences d'agents, d'accointances et de services de délégation de compétences et gère la distribution. La base d'un SMA dans Magique est composé d'une hiérarchie d'agents. Dans Magique, un agent possède des compétences (composant logiciel) qui peuvent évoluer dynamiquement (par échanges entre agents). Un agent est construit dynamiquement à partir d'un agent de base "vide", par enrichissement/acquisition successif de ses compétences. Cette notion de compétence disponible dans Magique est l'une des inspirations pour l'implantation du concept de capacité dans *Janus*. Une extension de Magique, nommée *G* a été proposée par Secq [2003], elle est basée sur un métamodèle nommé également RIO [Mathieu et al., 2003] comme celui de [Hilaire, 2000] et basé sur les trois même concepts : rôle, interaction et organisation. Mais là encore le concept de rôle ne correspond pas véritablement à notre vision de comportement à part entière et de statut dans une organisation.

Zeus [16] [Collis et al., 1998, Nwana et al., 1998] est un environnement complet de développement ainsi qu'une méthodologie de conception. Un outil graphique a été créé pour assister l'ensemble phases du processus de développement. Zeus est avant tout destinée à la conception d'agent cognitif, rationnel et basée sur une conception orientée-objectifs (goal-oriented). Développé en Java, elle se destine avant tout aux applications orientées Web.

MAST [17] [Boissier et al., 1998, Vercouter, 2004] - Multi-Agent System Toolkit - est un environnement de développement et de déploiement pour des applications multi-agents. elle est écrite en Java. Elle offre un environnement réparti pour l'exécution des agents (DEMAS), des outils d'observation et d'administration (AdMAS), des modèles réutilisables permettant d'organiser les traitements, de définir les schémas de contrôle en termes de modèles d'agent, d'environnement, d'interaction, d'organisation (GeMAS) et des interfaces de développement sur la base d'une méthodologie d'analyse et de conception (Me-

[15] Magique : http ://www2.lifl.fr/SMAC/projects/magique/
[16] Zeus : `http://labs.bt.com/projects/agents/zeus/`
[17] MAST : `http://www.emse.fr/~vercouter/mast/index.html`

MAS). MAST est donc un projet complet dont les domaines d'applications privilégiés concernent l'entreprise étendue et l'aménagement du territoire. L'approche choisie dans ce projet est très proche de celle que nous adoptons. Elle vise à disposer d'un métamodèle (ici MOISE [Hannoun et al., 2000]), de méthodologies associées (ex : MaMA-S [Galland et al., 2005]) et de la suite complète d'outils pour assister le processus de développement depuis l'analyse jusqu'à l'implantation.

MASDK [Gorodetski et al., 2005, 2002] - Multi-Agent System Development Kit - implante et est basé sur la méthodologie Gaia [Wooldridge et al., 2000, Zambonelli et al., 2003]. Cette plate-forme est en fait constituée de deux outils complémentaires *Generic Agent* qui fournit une bibliothèque de classes java et MASDK qui fournit un environnement intégré et un ensemble d'outils graphiques pour assister le processus de développement (squelettes d'agents et débogage). MAS-DK gère la mobilité des agents, leur exécution concurrente et temps réel. Une évaluation complète de MAS-DK et sa comparaison avec AgentSheets, AgentBuilder, Jack et OpenCybele a été effectuée dans [Bitting et al., 2003].

Madkit [18] [Gutknecht and Ferber, 2000, Gutknecht et al., 2000, Gutknecht, 2001] est basé sur le métamodèle organisationnel AALAADIN ou AGR. Madkit est l'une de nos sources d'inspiration pour la conception de *Janus*. Elle est développée en Java et fournit un ensemble d'outils au développeur. Le cœur de Madkit est basé sur son micro-noyau qui fournit tous les services de base nécessaires aux agents : gestion des données organisationnelles (groupe, rôle), communication, ordonnancement (concurrent ou synchrone). Un ensemble de bibliothèques vient ensuite étendre les fonctionnalités de ce noyau pour notamment permettre la connexion de plusieurs noyaux et ainsi distribuer des applications, ou faciliter les simulations. L'une des originalités de Madkit est qu'il intègre les agents et ses principes de modélisation dans la conception même de la plate-forme. Madkit fournit également un environnement graphique qui permet de visualiser et de contrôler les agents. Le principal défaut de Madkit tient à son implantation du modèle organisationnel et notamment du concept de Rôle. Les rôles ne sont pas véritablement des comportements que les agents peuvent acquérir dynamiquement, mais sont réduits à de simples *tags*. Cette approche des rôles nuit gravement à la modularité et à la généricité des organisations.

MOCA [Amiguet, 2003, Amiguet et al., 2002] est une plate-forme qui vient étendre Madkit et tenter de corriger ce défaut d'implantation des notions de rôle et d'organisation, mais vient également renforcer le contrôle d'accès à un rôle. Il intègre également en partie le formalisme OZS de description formelle des rôles proposé dans [Hilaire et al., 2000]. MOCA considère désormais l'organisation comme une véritable structure réutili-

[18] Madkit : `http://www.madkit.org`

sable. MOCA est basé sur un ensemble de composants Java. Un composant est une boîte noire qui fournit et requiert des compétences. Le rôle est désormais considéré comme un composant particulier dédié à l'interaction avec l'extérieur de l'agent. Quand un agent décide de jouer un nouveau un rôle cela revient à lui ajouter un nouveau composant.

B.6 Plates-formes intégrant la vision holonique

Un comparatif complet entre les différentes plates-formes/approches qui intègre la vision holonique a été effectuée par Sebastian Rodriguez dans sa thèse [Rodriguez, 2005], le tableau B.1 résume ces travaux.

	GEAMAS	**SWARM**	**MORISMA**	**SOHTCO**	**MMAS**
Structure	Arbre	Arbre	Arbre	Arbre	Arbre binaire
Niveaux	3	n	n	3	n
Architecture Agent	3 (Societé, Groupe, Agent)	2 (agent, swarm)	1 (agent récursif)	sub-SOHTCO (contient 4 types d'agents)	2 (atomique et composé)
Communication	messages / signaux	N.A.	Messages / signaux	Messages (KQML)	none
Prise de décision	décentralisé	décentralisé	centralisé	centralisé	décentralisé
Création des holons	Prédéfini (pdt la phase de modélisation)	Prédéfini	composition / décomposition des agents récursifs	Prédéfini (pas de création dynamique de holons)	dynamique

Table B.1: Comparaison des approches holoniques [Rodriguez, 2005]

SIXIÈME PARTIE

Bibliographie

Bibliographie

M.P. Huget A. Garro, P. Turci. Meta-model sources : GAIA. Technical report, Foundation for Intelligent Physical Agents, March 2004. `http://www.pa.icar.cnr.it/~cossentino/FIPAmeth/docs/GAIA.pdf`.

Nadia Abchiche. Integrating heterogeneous reasoning in a multi-agent system. In *27th Hawai International Conference of System Sciences (HICSS)*, pages 55–62, Jan 1994.

Nadia Abchiche. *Elaboration, implémentation et validation d'une approche distribuée pour l'intégration de modèles de raisonnement hétérogènes : application au diagnostic de pannes électriques*. PhD thesis, Université Paris 8, Janvier 1999.

E. Adam. *Modèle d'organisation multi-agent pour l'aide au travail coopératif dans les processus d'entreprise : application aux systèmes administratifs complexes*. PhD thesis, Univ. de Valenciennes et du Hainaut-Cambresis, 2000.

Peter M. Allen. *Cities and Regions as Self-organising Systems : Models of Complexity*. Gordon and Breach Science Publishers, Amsterdam, 1997.

Frédéric Amblard. *Comprendre le fonctionnement de simulations sociales individus-centrées, Application à des modèles de dynamiques d'opinions*. PhD thesis, Cemagref, Clermont-Ferrand, France, Décembre 2003.

Matthieu Amiguet. *MOCA : un modèle componentiel dynamique pour les systèmes multi-agents organisationnels*. PhD thesis, Université de Neuchâtel, 2003.

Matthieu Amiguet, Jean-Pierre Müller, José-A. Baez-Barranco, and Adina Nagy. The MOCA Platform, Simulating the Dynamics of Social Networks. In *Multi-Agent-Based Simulation II, Third International Workshop (MABS)*, volume 2581 of *Lecture Notes in Computer Science*, pages 145–167, Bologna, Italy, July 2002. Springer Berlin, Heidelberg.

Egil P. Anderson and Trygve Reenskaug. System Design by Composing Structures of Interacting Objects. In *European Conference on Object-Oriented Programming*, volume 615 of *Lecture Notes*, pages 133–153. Springer Verlag, 1992.

Estefania Argente, Vicente Julian, and Vicente Botti. Multi-agent system development based on organizations. In *First International Workshop on Coordination and Organisation (CoOrg)*, volume 150 of *Electronic Notes in Theoretical Computer Science*, pages 55–71. Elsevier, May 2006.

T. Balch. Hierarchic social entropy : An information theoretic measure of robot group diversity. In *Autonomous Robots*, volume 8, July 2000.

P. Ballet, V. Rodin, and J. Tisseau. A multiagent system to simulate in-vitro experimentation. In *Conference on Systemics, Cybernetics and Informatics (SCI)*, Orlando, USA, July 1998.

T. Balser, K. McMahon, D. Bart, D. Bronson, D. Coyle, N. Craig, M. Flores-Mangual, K. Forshay, S. Jones, A. Kent, and A. Shade. Bridging the gap between micro- and macro-perspectives on the role of microbial communities in global change ecology. *Plant and Soil*, 289 :59–70, 2006.

M. Balzer and O. Deussen. Level-of-detail visualization of clustered graph layouts. In *Asia-Pacific Symposium on Visualisation (APVIS)*, pages 133–140, Feb 2007.

John S. Baras and Xiaobo Tan. Control of autonomous swarms using gibbs sampling. In *43rd IEEE Conf. on Decision and Control*, Bahamas, dec 2004.

Boldur Barbat, Ciprian Candea, and Constantin Zamfirescu. Holons and agents in robotic teams. a synergistic approach. In *Proceedings of ENAIS'2001*, pages 654–660, 2001. ISBN 3-906454-25-8.

Bernhard Bauer. Uml class diagrams revisited in the context of agent-based systems. In *Revised Papers and Invited Contributions from the Second International Workshop on Agent-Oriented Software Engineering II (AOSE)*, pages 101–118, London, UK, 2002. Springer-Verlag. ISBN 3-540-43282-5.

Bernhard Bauer, Jörg P. Müller, and James Odell. Agent UML : A formalism for specifying multiagent interaction. In Paolo Ciancarini and Michael Wooldridge, editors, *Agent-Oriented Software Engineering*, pages 91–103. Springer, 2001.

C. Bäumer and T. Magedanz. Grasshopper - A Mobile Agent Platform for Active Telecommunication Networks. In *ntelligent Agents for Telecommunication Applications : Third International Workshop (IATA)*, volume 1699 of *LNCS*, pages 19–32. Springer Berlin / Heidelberg, Stockholm, Sweden, August 1999.

C. Bäumer, M. Breugst, S. Choy, and T. Magedanz. Grasshopper : a universal agent platform based on OMG MASIF and FIPA standards. Technical report, IKV++ GmbH, 2000.

F. Bellifemine, A. Poggi, and G. Rimassa. JADE : a FIPA2000 compliant agent development environment. In *Agents*, pages 216–217, 2001.

Mustapha Ben Mahbous. Un système d'information en architecture et urbanisme basé sur la représentation de connaissance. In *Rencontre des doctorants des écoles d'architecture du sud de la France*, juin 1995.

Itzhak Benenson, Shai Aronovich, and Saar Noam. OBEUS : Object-based environment for urban simulations. In *6th International Conference on GeoComputation*, Brisbane, Australia, September 2001.

Itzhak Benenson, Shai Aronovich, and Saar Noam. Let's talk objects : Generic methodology for urban high-resolution simulation. *Computers, Environment and Urban Systems*, 29 :425–453, 2005.

Nourredine Bensaid and Philippe Mathieu. A framework for cooperation in hierarchical multi-agent systems. *Mathematical Modelling and Scientific Computing*, 8, 1997a.

Nourredine Bensaid and Philippe Mathieu. A hybrid architecture for hierarchical agents. In *International Conference on Computational Intelligence and Multimedia Applications (ICCIMA)*, 1997b.

F. Bergenti and M. Huhns. *Methodologies and Software Engineering for Agent System : The Agent Oriented Software Engineering Handbook*, chapter 2 : On the Use of Agents as Components of Software Systems, pages 19–32. Kluwer Academic Publisher, New York, 2004.

Federico Bergenti and Agostino Poggi. Exploiting uml in the design of multi-agent systems. In Andrea Omicini, Robert Tolksdorf, and Franco Zambonelli, editors, *Engineering Societies in the Agents' World*, Lecture Notes in Artificial Intelligence. Springer Verlag, 2000.

C. Bernon, V. Camps, M.-P. Gleizes, and G. Picard. Tools for Self-Organizing Applications Engineering. In G. Di Marzo Serugendo, A. Karageorgos, O. F. Rana, and F. Zambonelli, editors, *Engineering Self-Organizing Applications – First International Workshop (ESOA) at the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, volume 2977 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages 283–298, Melbourne, Australia, 2003. Springer-Verlag. ISBN 3-540-21201-9.

Carole Bernon, Maire-Pierre Gleizes, Sylvain Peyruqueou, and Gauthier Picard. ADELFE, a methodology for adaptive multi-agent systems engineering. In *Third International Workshop Engineering Societies in the Agents World (ESAW)*, volume 2577 of *LNAI*, pages 156–169, Madrid, Spain, September 2002. Springer-Verlag.

Carole Bernon, Massimo Cossentino, and Juan Pavón. An overview of current trends in european aose research. *Informatica*, 29(4) :379–390, July 2005.

Jean Bézivin. On the Unification Power of Models. *Software and System Modeling (SoSym)*, 4(2) :171–188, 2005.

Michel Bierlaire, Giancula Antonini, and Mats Weber. Behavioral dynamics for pedestrians. In Elsevier, editor, 10^{th} *International Conference on Travel Behavior Research*, Lucerne, August 2003.

J.P. Bigus, D.A. Schlosnagle, J.R. Pilgrim, W.N. Mills III, and Y. Diao. ABLE : A toolkit for building multiagent autonomic systems. *IBM Systems Journal*, 3 :350–371, 2002.

Elijah Bitting, Jonathan Carter, and Ali A. Ghorbani. Multiagent System Development Kits : An Evaluation. In *1st Annual Conference on Communication Networks and Services Research (CNSR)*, pages 80–92, Moncton, Canada, May 2003.

Barry W. Boehm. A spiral model of software development and enhancement. *Computer*, 21(5) :61–72, 1988. ISSN 0018-9162.

O. Boissier, P. Beaune, H. Proton, M. Hannoun, T. Carron, L. Vercouter, and C. Sayettat. The Multi-Agent System Toolkit. Technical report, SIC/ENSM-SE, 1998.

Grady Booch. *Object-Oriented Analysis and Design with Applications*. Addison-Wesley Professional, 2nd edition, September 1993.

Rafael Bordini, Lars Braubach, Mehdi Dastani, Amal El Fallah Seghrouchni, Jorge Gomez-Sanz, Joao Leite, Gregory O'Hare, Alexander Pokahr, and Alessandro Ricci. A survey of programming languages and platforms for multi-agent systems. In *Informatica 30*, pages 33–44, 2006.

Rafael H. Bordini, Antônio Carlos da Rocha Costa, Jomi F. Hübner, Álvaro F. Moreira, Fabio Y. Okuyama, and Renata Vieira. MAS-SOC : a social simulation platform based on agent-oriented programming. *Journal of Artificial Societies and Social Simulation*, 8(3), 2005.

François Bousquet, Innocent Bakam, Hubert Proton, and Christophe Le Page. Cormas : Common-pool resources and multi-agent systems. In *11th International Conference on Industrial and Engineering Applications of Artificial In telligence and Expert Systems (IEA/AIE)*, pages 826–837, London, UK, 1998. Springer-Verlag.

A. Braun, S.R. Musse, L.P.L. de Oliveira, and B.E.J. Bodmann. Modeling individual behaviors in crowd simulation. In *16th International Conference on Computer Animation and Social Agents*, pages 143–148, May 2003.

F. Brazier, B. Dunin-Keplicz, N. R. Jennings, and J. Treur. Formal specification of multiagent systems : a real-world case. In *First International Conference on Multi-Agent Systems (ICMAS)*, pages 25–32, San Francisco, USA, June 1995.

F. Brazier, B.D. Keplicz, N. Jennings, and J. Trueur. Desire : Modelling multi-agent systems in a compositional formal framework. *International Journal of Cooperative Information Systems*, 6 :67–94, 1997.

F.M.T. Brazier, C.M. Jonker, and J. Treur. Principles of compositional multi-agent system development. In J. Cuena, editor, *Conference on Information Technology and Knowledge Systems (IFIP,IT-KNOWS)*, page 14. Chapman and Hall, 1998.

P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos, and A. Perini. TROPOS : An agent-oriented software development methodology. *Journal of Autonomous Agents and Multi-Agent Systems*, 8(3) :203–236, May 2004.

Anne Bretagnolle, Eric Daudé, and Denise Pumain. From theory to modelling : urban systems as complex systems. *Cybergeo, 13th European Colloquium on Quantitative and Theoretical Geography*, 335, September 2003.

X. Briffault, N. Guichard, J.P. Kotowicz, and D. D. Pierre. La plate-forme MERCURE. In *Systèmes multi-agents, méthodologie, technologie et expériences, 8èmes JF IAD et SMA*, pages 273–276. Hermès, 2000.

Sjaak Brinkkemper, Kalle Lyytinen, and Richard Welke, editors. *Method Engineering : Principles of Method Construction and Tool Support (IFIP International Federation for Information Processing)*. Kluwer Academic Publishers, August 1996. ISBN 041279750X.

D.C. Brogan and J.K. Hodgins. Simulation level of detail for multiagent control. In *AAMAS*, pages 1–8. ACM Press, July 2002. ISBN 1-58113-480-0.

H. Van Brussel, J. Wyns, P. Valckenaers, L. Bongaerts, and P. Peeters. Reference architecture for holonic manufacturing systems : Prosa. *Computers in Industry*, 37 :255–274, 1998.

H.J. Bürckert, K. Fischer, and G.Vierke. Transportation scheduling with holonic MAS - the teletruck approach. In *Conf. on Practical Applications of Intelligent Agents and Multiagent*, pages 577–590, 1998.

Wilco Burghout. *Hybrid microscopic-mesoscopic traffic simulation*. PhD thesis, KTH Infrastructure, Royal Institute of Technology, Stockholm, Sweden, dec 2004.

Wilco Burghout, H. Koutsopoulos, and I. Andréasson. Hybrid mesoscopic-microscopic traffic simulation. In *World Conference on Transportation Research CTR2004*, volume 02, 2004.

Wilco Burghout, H. Koutsopoulos, and I. Andréasson. Hybrid mesoscopic-microscopic traffic simulation. In *Transportation Research Record, World Conference on Transportation Research CTR2005*, volume 01, pages 218–225, 2005.

Roberto Caico, Massimo Cossentino, Luca Sabatucci, Valeria Seidita, and Salvatore Gaglio. MetaMeth : a Tool For Process Definition And Execution. In *Proc. of the 7th Workshop From Objects to Agents (WOA)*, pages 21–24, Catania, Italy, Sept 2006.

Giovanni Caire, Wim Coulier, Francisco J. Garijo, Jorge Gomez, Juan Pavón, Francisco Leal, Paulo Chainho, Paul E. Kearney, Jamie Stark, Richard Evans, and Philippe Massonet. Agent Oriented Analysis Using Message/UML. In Michael Wooldridge, Gerhard Weiß, and Paolo Ciancarini, editors, *Agent-Oriented Software Engineering II, Second International Workshop, AOSE 2001, Montreal, Canada, May 29, 2001, Revised Papers and Invited Contributions*, volume 2222 of *Lecture Notes in Computer Science*, pages 119–135. Springer Verlag, 2002. ISBN 3-540-43282-5.

Ciprian Candea, Marius Staicu, and Boldur Barbat. Holon - like approach for robotic soccer. In *Proceedings of the RoboCup European Workshop 2000*, Amsterdam, Holland, 2000.

Kathleen M. Carley and Michael J. Prietula, editors. *Computational Organization Theory*. Lawrence Earlbaum, 1994.

B. Carpuat and C. Regis. Synergic : A multi-agent environment. In *International Workshop on Decentralized Intelligent and Multi-agent Systems (DIMAS)*, Krakow, Poland, November 1995.

L. Cernuzzi, M. Cossentino, and F. Zambonelli. Process models for agent-based development. *Journal of Engineering Applications of Artificial Intelligence (EAAI)*, 18(2), March 2005.

Gilbert Chauvet. *La vie dans la matière*. Flammarion, 1998.

A. Chella, M. Cossentino, and L. Sabatucci. Designing jade systems with the support of case tools and patterns. *Exp Journal*, 3(3) :86–95, September 2003.

A. Chella, M. Cossentino, L. Sabatucci, and V. Seidita. From PASSI to Agile PASSI : Tailoring a design process to meet new needs. In *IEEE/WIC/ACM International Joint Conference on Intelligent Agent Technology (IAT)*, Beijing, China, september 2004.

A. Chella, M. Cossentino, L. Sabatucci, and V. Seidita. Agile PASSI : An agile process for designing agents. *International Journal of Computer Systems Science and Engineering. Special issue on Software Engineering for Multi-Agent Systems*, 21(2), March 2006.

Stephen Chenney and David Forsyth. View-dependent culling of dynamic systems in virtual environments. In *Proceedings of the symposium on Interactive 3D graphics (SI3D)*, pages 55–58, New York, NY, USA, April 1997. ACM Press. ISBN 0-89791-884-3.

Stephen Chenney, Okan Arikan, and D.A. Forsyth. Proxy simulations for efficient dynamics. In *Eurographics, Short Presentations*, 2001.

V. Chevrier. *Etude et mise en oeuvre du paradigme multi-agents : de Atome à GTMAS*. PhD thesis, Université de Nancy I, Juin 1993.

V. Chevrier. Gtmas : A tool for prototyping and assessing design choices in multi-agent systems. In *Actes des journées Avignon'94*, pages 161–170, Juin 1994.

P. E. Clements, Todd Papaioannou, and John Edwards. Aglets : Enabling the virtual enterprise. In *Managing Enterprises – Stakeholders, Engineering, Logistics and Achievement (ME-SELA)*, Loughborough University, UK, 1997.

J. C. Collis, D. T. Ndumu, H. S. Nwana, and L. C. Lee. The ZEUS Agent Building Tool-kit. *BT Technology Journal*, 16(3) :60–68, 1998. ISSN 1358-3948.

R. Conte and N. Gilbert. Introduction : computer simulation for social theory. *Artificial societies - the computer simulation of social life*, pages 1–18, 1995. UCL Press.

R. Conte, N. Gilbert, and J. Simao Sichman. MAS and social simulation : A suitable commitment. In *First International Workshop on Multi-Agent Systems and Agent-Based Simulation*, volume 1534 of *LNCS*, pages 1–9. Springer-Verlag, 1998.

J.M. Contet, F. Gechter, P. Gruer, and A. Koukam. Physics inspired multiagent model for vehicle platooning. In *International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, May 2007.

Patrick Coquillard and David R.C. Hill. *Modélisation et Simulation des Ecosystèmes*. Masson, 1997.

Kelly Christine Correa e Silva Fernandes. *Systèmes Multi-Agents Hybrides : Une Approche pour la Conception de Systèmes Complexes*. PhD thesis, Université Joseph Fourier- Grenoble 1, 2001.

M. Cossentino and C. Potts. A CASE tool supported methodology for the design of multi-agent systems. In *The 2002 International Conference on Software Engineering Research and Practice (SERP)*, Las Vegas (NV), USA, June 24 - 27 2002.

M. Cossentino and V. Seidita. Composition of a new process to meet Agile needs using method engineering. In *Software Engineering for Large Multi-Agent Systems*, volume 3390 of *LNCS*, pages 36–51. Springer-Verlag GmbH, 2005.

M. Cossentino, L. Sabatucci, and A. Chella. Patterns reuse in the PASSI methodology. In *Engineering Societies in the Agents World IV, 4^{th} International Workshop, ESAW*, volume XIII of *Lecture Notes in Artificial Intelligence.* Springer-Verlag, 2004.

M. Cossentino, S. Gaglio, L. Sabatucci, and V. Seidita. The PASSI and Agile PASSI MAS Meta-models Compared with a Unifying Proposal. In *Multi-Agent Systems and Applications IV*, volume 3690 of *LNCS*, pages 183–192. Springer-Verlag, 2005.

M. Cossentino, L. Sabatucci, V. Seidita, and S. Gaglio. An Agent Oriented Tool for New Design Processes. In *Fourth European Workshop on Multi-Agent Systems (EUMAS'06)*, Lisbon, Portugal, December 2006a.

M. Cossentino, L. Sabatucci, V. Seidita, and S. Gaglio. An agent oriented tool for new design processes. In *Fourth European Workshop on Multi-Agent Systems (EUMAS)*, Lisbon, Portugal, December 2006b.

M. Cossentino, S. Gaglio, A. Garro, and V. Seidita. Method fragments for agent design methodologies : from standardization to research. *International Journal on Agent Oriented Software Engineering (IJAOSE)*, 1 (1) :91–121, April 2007.

Massimo Cossentino. From Requirements to Code with the PASSI Methodology. In B. Henderson-Sellers and P. Giorgini, editors, *Agent-Oriented Methodologies*, chapter IV, pages 79–106. Idea Group Publishing, Hershey, PA, USA, 2005.

S. Costantini and A. Tocchio. A logic programming language for multi-agent systems. In S. Flesca, S. Greco, N. Leone, and G. Ianni, editors, *Proc. of the 8th European Conference on Logics in Artificial Intelligence (JELIA)*, volume 2424 of *LNAI*, pages 1–13. Springer, 2002.

Luciano dos Reis Coutinho, Jaime Simão Sichman, and Olivier Boissier. Modeling organization in MAS : a comparison of models. In *Proc. of the 1st. Workshop on Software Engineering for Agent-Oriented Systems (SEAS)*, Uberlândia, October 2005.

K. H. Dam and M. Winikoff. Comparing agent-oriented methodologies. In *Fifth International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS)*, 2003.

Mehdi Dastani and Jorge J. Gomez-Sanz. Programming multi-agent systems (promas), a report of the technical forum meeting, April 2005. url : `http://people.cs.uu.nl/mehdi/tfg/ljubljanafiles/report.pdf`.

P. Davidsson. Multi agent based simulation : Beyond social simulation. *Multi Agent Based Simulation, Springer Verlag LNCS series,*, 1979, 2000.

Scott A. DeLoach, Mark F. Wood, and Clint H. Sparkman. Multiagent systems engineering. *The International Journal of Software Engineering and Knowledge Engineering*, 11(3), June 2001.

Y. Demazeau. Steps towards multi-agent oriented programming. In *1st International Workshop on Multi-Agent Systems (IWMAS)*, Boston, 1997.

Yves Demazeau. *VOYELLES*. PhD thesis, Institut National Polytechnique de Grenoble INPG, Grenoble, France, Avril 2001. Habilitation à Diriger des Recherches.

M.V. Dignum, J. Vazquez-Salceda, and F.P.M. Dignum. OMNI : Introducing social structure, norms and ontologies into agent organizations. In R.H. Bordini, M. Dastani, J. Dix, and A.E.F. Seghrouchni, editors, *Second International Workshopon Programming Multi-Agent Systems (ProMAS)*, volume 3346 of *LNAI*, pages 181–198, New York, USA, July 2005. Springer.

S. Donikian. VUEMS : a virtual urban environment modeling system. In *Computer Graphics International*, pages 84–92, Hasselt and Diepenbeek, Belgium, June 1997. IEEE Computer Society Press.

Kurt Dopfer, John Foster, and Jason Potts. Micro-meso-macro. *Journal of Evolutionary Economics*, 14(3) : 263–279, July 2004.

A. Drogoul. *De la simulation multi-agents à la résolution collective de problèmes*. PhD thesis, University Paris 6, Paris, France, November 1993.

A. Drogoul and J. Ferber. Multi-agent simulation as a tool for modeling societies : application to social differentiation in ant colonies. *Artificial Social Systems*, 830(8) :3–23, 1994.

A. Drogoul and S. Picault. Microbes : Vers des collectivités de robots socialement situés. In M.-P. Gleizes and P. Marcenac, editors, *VIIèmes Journées Francophones pour l'Intelligence Artificielle Distribuée et les Systèmes Multi-Agents (JFIADSMA)*, pages 265–278. Hermès, 1999.

Benoît Durand. *Simulation multi-agents et épidiémiologiue opérationnelle*. PhD thesis, Université de Caen, 1996.

P. Eades and Q. W. Feng. Multilevel visualization of clustered graphs. In S. North, editor, *Graph Drawing 96*, volume 1190 of *Lecture Notes in Computer Science*. Springer, 1996.

Mark Edwards. A brief history of holons, October 2003.

M.E. Epstein and R. Axtell. *Growing Artificial Societies : Social Science from the Ground Up*. MIT Press, 1996.

Marc Esteva, Juan A. Rodríguez-Aguilar, Carles Sierra, Pere Garcia, and Josep Lluís Arcos. On the formal specifications of electronic institutions. In *Agent Mediated Electronic Commerce, The European AgentLink Perspective.*, pages 126–147, London, UK, 2001. Springer-Verlag. ISBN 3-540-41671-4.

Nathalie Farenc, Ronan Boulic, and Daniel Thalmann. An informed environment dedicated to the simulation of virtual humans in urban context. In *Eurographics'99*, volume 18, pages 309–318, Milano, Italy, 1999.

J. Ferber and J.P. Müller. Influences and reactions : a model of situated multiagent systems. In *Second Internationnal Conference on Multi-Agent Systems (ICMAS)*, pages 72–79, 1996.

J. Ferber, O. Gutknecht, and F. Michel. From agents to organizations : an organizational view of multi-agent systems. In *Agent-Oriented Software Engineering IV 4th International Workshop*, volume 2935 of *LNCS*, pages 214–230, Melbourne, Australia, mar 2004. Springer Verlag.

Jacques Ferber. *Les Systèmes Multi-Agents : Vers une Intelligence Collective*. InterEditions, 1995.

Jacques Ferber and Olivier Gutknecht. A meta-model for the analysis and design of organizations in multi-agent systems. In Y. Demazeau, E. Durfee, and N.R. Jennings, editors, *Third International Conference on Multi-Agent Systems (ICMAS)*, pages 128–135, Paris, France, july 1998.

Klaus Fischer, Michael Schillo, and Jörg Siekmann. Holonic multiagent systems : A foundation for the organisation of multiagent systems. In *Holonic and Multi-Agent Systems for Manufacturing*, volume 2744 of *LNCS*, pages 71–80. Springer Berlin, Heidelberg, 2003.

M. Fisher and M. Wooldridge. On the formal specification and verification of multi-agent systems. *Intern. Journal of Cooperative Information Systems*, 6(1) :37–65, 1997.

Paul A. Fishwick. Computer simulation : growth through extension. *Trans. Soc. Comput. Simul. Int.*, 14(1) : 13–23, 1997. ISSN 0740-6797.

Rémy Foisel. Construire des systèmes multi-agents à partir de schémas d'interactions. In V. Chevrier et Chr. Brassac J.P. Barthès, editor, *Journées Francophones d'Intelligence Artificielle et Systèmes Multi-Agents (JFIADSMA)*, pages 295–308, Pont-à-Mousson, France, 1998a. Hermes.

Rémy Foisel. *Modèle de réorganisation de SMA : Une approche descriptive et opérationnelle*. PhD thesis, Université Henri Poincaré, Nancy I, novembre 1998b.

FIPA ACL, 2002. *FIPA ACL Message Structure Specification*. Foundation For Intelligent Physical Agents, 2002. Standard, SC00061G.

FIPA Com. Act, 2002. *FIPA Communicative Act Library Specification*. Foundation For Intelligent Physical Agents, 2002. Standard, SC00037J.

FIPA RDF, 2001. *FIPA RDF Content Language Specification*. Foundation For Intelligent Physical Agents, 2001. Experimental, XC00011B.

Alfonso Fuggetta. Software process : a roadmap. In ACM Press, editor, *Int. Conference on Software Engineering (ICSE), Future of Software Engineering Track*, pages 25–34, Limerick, Ireland, June 2000.

A. Fuxman, M. Pistore, J. Mylopoulos, and P. Traverso. Model Checking Early Requirements Specifications in Tropos. In *5th IEEE International Symposium on RequirementsEngineering (RE)*, pages 174–181, 2001.

Stéphane Galland, Frédérique Grimaud, Philippe Beaune, and Jean-Pierre Campagne. Simulation of distributed industrial systems - a multi-agent methodological approach. In Alexandre Dolgui, Jerzy Soldek, and Oleg Zaikin, editors, *Supply Chain Optimisation : Product/Process Design, Facility Location and Flow Control*, volume 94 of *Applied Optimization*. Springer, 2005. ISBN 0-387-23566-3.

Maddalena Garzetti, Paolo Giorgini, John Mylopoulos, and Fabrizio Sannicollò. Applying Tropos Methodology to a real case study : Complexity and Criticality Analysis. In *Italian Workshop on Dagli OGGETTI agli AGENTI - Dall'informazione alla Conoscenza (WOA)*, 2002.

L. Gasser. MACE : High-level distributed objects in a flexible testbed for distributed AI research. In *ACM SIGPLAN, workshop on Object-based concurrent programming*, pages 108–110, New York, NY, USA, 1988. ACM Press. ISBN 0-89791-304-3.

L. Gasser, C. Braganza, and N. Herman. MACE : A flexible testbed for distributed ai research. In M. Huhns, editor, *Distributed Artificial Intelligence*, pages 119–152. Pitman/Morgan Kaufmann, 1987.

Les Gasser. Boundaries, identity, and aggregation : plurality issues in multiagent systems. *SIGOIS Bull.*, 13 (3) :13, 1992. ISSN 894–819.

N. Gaud, S. Galland, and A. Koukam. Visual perception for virtual agents : Application to a pedestrian simulation. In Daniel Coutellier, Philippe Fuchs, Xavier Fischer, and Indira Thouvenin, editors, *Virtual Reality for Industrial Applications Workshop (VIA) of Virtual Concept Conference*, Compiègne, France, November 2004.

Christian Gerber, Jörg H. Siekmann, and Gero Vierke. Holonic multi-agent systems. Technical Report DFKI-RR-99-03, Deutsches Forschungszentrum für Künztliche Inteligenz - GmbH, Postfach 20 80, 67608 Kaiserslautern, FRG, May 1999.

Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, second edition, 2002. ISBN 0-13-305699-6.

S. Ghosh. On the concept of dynamic multi-level simulation. In *the 19th Annual Symposium on Simulation*, pages 201–205, Tampa, Florida, U.S.A, 1986. ISBN 0-8186-0715-7.

Nigel Gilbert and Klaus G. Troitzsch. *Simulation for the Social Scientist*. Open University Press, Maidenhead and New York, 2 edition, 2005.

V. Ginot and C. Le Page. Mobidyc, a generic multi-agent simulator for modeling communities dynamics. In *IEA-98-AIE*, volume 1416 of *LNAI*, pages 805–814. Springer, 1998.

V. Ginot, C. Le Page, and S. Souissi. A multi-agent architecture to enhance end-user individual-based modelling. *Ecological Modeling*, 157 :23–41, 2002.

Adriana Giret. *ANEMONA : A Multi-Agent Method for Holonic Manufacturing Systems*. PhD thesis, Universidad Politécnica de Valencia, Valencia, Spain, June 2005.

Adriana Giret and Vicente Botti. *Agent-Oriented Software Engineering IV*, volume 2935 of *Lecture Notes in Computer Science*, chapter Towards a Recursive Agent Oriented Methodology for Large-Scale MAS, pages 25–35. Springer Berlin / Heidelberg, 2003.

Adriana Giret and Vicente Botti. Holons and agents. *Journal of Intelligent Manufacturing*, 15 :645–659, 2004.

Adriana Giret and Vicente Botti. Analysis and design of holonic manufacturing systems. In *The Networked Enterprise : a challenge for sustainable development, 18th International Conference on Production Research*, 2005.

Adriana Giret and Vicente Botti. From system requirements to holonic manufacturing analysis. *International Journal of Production Research*, 44(18-19) :3917–3928, 2006.

Adriana Giret, Vicente Botti, and Soledad Valero. MAS methodology for HMS. In Springer-Verlag, editor, *Proc. of the Second International Conference on Applications of Holonic and Multi-Agent Systems (HoloMAS)*, volume 3593 of *LNAI*, pages 39–49, 2005.

F. Giunchiglia, J. Mylopoulos, and A. Perini. The Tropos Software Development Methodology : Processes, Models and Diagrams. Technical Report 0111-20, ITC - IRST, 2002. Submitted AAMAS Conference 2002. A Knowledge Level Software Engineering 15.

Marie-Pierre Gleizes and Gauthier Picard. OpenTool, outil pour la réalisation de systèmes multi-agents adaptatifs dans le cadre de la méthode ADELFE. *Technique et Science Informatiques*, 22(1) :249–253, 2003.

Christian Gloor, Duncan Cavens, Eckart Lange, Kai Nagel, and Willy Schmid. A pedestrian simulation for very large scale applications, December 2003.

Christian Gloor, Pascal Stucki, and Kai Nagel. Hybrid techniques for pedestrian simulations. In 4^{th} *Swiss Transport Research Conference STRC*, Monte Verità, Ascona, March 2004.

Vladimir Gorodetski, Oleg Karsaev, Vladimir Samoilov, Victor Konushy, Evgeny Mankov, and Alexey Malyshev. Multi-agent System Development Kit : MAS software tool implementing GAIA methodology. *Intelligent information processing II*, pages 69–78, 2005.

Vladimir I. Gorodetski, Oleg Karsayev, Igor V. Kotenko, and Alexey Khabalov. Software development kit for multi-agent systems design and implementation. In *CEEMAS'01 : Revised Papers from the Second International Workshop of Central and Eastern Europe on Multi-Agent Systems*, pages 121–130, London, UK, 2002. Springer-Verlag. ISBN 3-540-43370-8.

Abdelkader Gouaich, Fabien Michel, and Yves Guiraud. MIC* : a deployment environment for autonomous agents. *LNAI, Environments for Multi-Agent Systems, First International Workshop, E4MAS 2004*, 3374 : 109–126, July 2004.

John P. Granieri, Welton Becket, Barry D. Reich, Jonathan Crabtree, and Norman I. Badler. Behavioral control for real-time simulated human agents. In *SI3D'95 : Proceedings of the 1995 symposium on Interactive 3D graphics*, pages 173–180, New York, NY, USA, 1995. ACM Press.

Kasper Bilsted Graversen. *The nature of roles, A taxonomic analysis of roles as a language construct*. PhD thesis, IT University of Copenhagen, Denmark, 2006.

Thomas Gruber. Toward principles for the design of ontologies used for knowledge sharing. *International Journal Human-Computer Studies*, 43(Issues 5-6) :907–928, November 1995.

P. Gruer, V. Hilaire, A. Koukam, and P. Rovarini. Heterogeneous formal specification based on object-z and statecharts : semantics and verification. *Journal of Systems and Software*, 70(1-2) :95–105, 2004.

Z. Guessoum. DIMA : Une plate-forme multi-agents en smalltalk. *Revue Objet*, 3(4) :393–410, 1998.

Z. Guessoum and J.-P. Briot. From active object to autonomous agents. *IEEE Concurrency*, 7(3) :68–78, July/September 1999.

Christine Guilfoyle and Ellie Warner. Intelligent agents : The new revolution in software. Technical report, Ovum Ltd., London, May 1994.

O. Gutknecht and J. Ferber. Madkit : a generic multi-agent platform. autonomous agents. In *AGENTS 2000*, pages 78–79, Barcelona, 2000. ACM Press.

O. Gutknecht, J. Ferber, and F Michel. Madkit : une expérience d'architecture de plate-forme multi-agent générique. In *8ème JFIADSMA*, pages 223–236. Hermès, 2000.

Olivier Gutknecht. *Proposition d'un modèle organisationnel générique de systèmes multi-agents et examen de ses conséquences formelles, implémentatoires et méthodologiques.* PhD thesis, Université des Sciences et Techniques du Languedoc, september 2001.

Mahdi Hannoun, Olivier Boissier, Jaime Simao Sichman, and Claudette Sayettat. MOISE : An organizational model for multi-agent systems. In Monard M. et Sichman J., editor, *Advances in Artificial Intelligence, IBERAMIA-SBIA*, pages 156–165, Brazil, 2000.

Andrew J. Hanson and Eric A. Wernert. Constrained 3D navigation with 2D controllers. In *VIS'97 : Proceedings of the 8th conference on Visualization'97*, pages 175–ff., Los Alamitos, CA, USA, 1997. IEEE Computer Society Press.

David Harel. Statecharts : A visual formalism for complex systems. *Science of Computer Programming*, 8 (3) :231–274, June 1987. Preliminary version : Technical Report CS84-05, The Weizmann Institute of Science, Rehovot, Israel, February 1984.

Fabrice Harrouet. *oRis : s'immerger par le langage pour le prototypage d'univers virtuels à base d'entités autonome*. PhD thesis, École Nationale d'Ingénieurs de Brest (ENIB), Décembre 2000.

D. Helbing, I. Farkas, and T. Vicsek. Simulating dynamical features of escape panic. *Nature*, 407 :487–490, 2000.

Brian Henderson-Sellers. Method engineering for OO systems development. *Commun. ACM*, 46(10) :73–78, 2003. ISSN 0001-0782.

Vincent Hilaire. *Vers une approche de spécification, de prototypage et de vérification de Systèmes Multi-Agents*. PhD thesis, Université de Technologie de Belfort-Montbéliard, 2000.

Vincent Hilaire, Abder Koukam, Pablo Gruer, and Jean-Pierre Müller. Formal specification and prototyping of multi-agent systems. In Andrea Omicini, Robert Tolksdorf, and Franco Zambonelli, editors, *Engineering Societies in the Agents' World*, number 1972 in Lecture Notes in Artificial Intelligence. Springer Verlag, 2000.

A. Hodgson, R. Rönnquist, and P. Busetta. Specification of Coordinated Agent Behaviour (The SimpleTeam Approach). Technical report, Agent Oriented Software Pty. Ltd., Melbourne, Australia, october 1999.

John H. Holland. *Hidden order : how adaptation builds complexity*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, july 1995. ISBN 0-201-40793-0.

Serge P. Hoogendoorn and Piet H.L. Bovy. State-of-the-art of vehicular traffic flow modelling. *Special Issue on Road Traffic Modelling and Control of the Journal of Systems and Control Engineering*, 215(4) :283–303, August 2001.

N. Howden, R. Rönnquist, A. Hodgson, and A. Lucas. JACK Intelligent Agents – Summary of an Agent Infrastructure. In *Second International Workshop on Infrastructure for Agents, MAS, and Scalable MAS*, 2001.

J.F. Hübner, J.S. Sichman, and O. Boissier. A model for the structural, functional, and deontic specification of organizations in multiagent systems. In G. Bittencourt and G.L. Ramalho, editors, *Advances in Artificial Intelligence : 16th Brazilian Symposium on Artificial Intelligence (SBIA)*, volume 2507 of *LNAI*, pages 118–128. Springer, November 2002.

IEEE-729, 1983. *IEEE Standard 729-1983 : IEEE Standard Glossary of Software Engineering Terminology*. IEEE Computer Society, June 1983.

Carlos Iglesias, Mercedes Garrijo, and José Gonzalez. A survey of agent-oriented methodologies. In Jörg Müller, Munindar P. Singh, and Anand S. Rao, editors, *Proceedings of the 5th International Workshop on Intelligent Agents V : Agent Theories, Architectures, and Languages (ATAL)*, volume 1555, pages 317–330. Springer-Verlag : Heidelberg, Germany, 1999.

Carlos Argel Iglesias, Mercedes Garijo, Jose Centeno-Gonzalez, and Juan R. Velasco. A methodological proposal for multiagent systems development extending CommonKADS. In B. Gaines and M. Musen, editors, *10th Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW)*, Banoe, Canada, 1996.

Carlos Argel Iglesias, Mercedes Garijo, Jose Centeno-Gonzalez, and Juan R. Velasco. Analysis and design of multiagent systems using MAS-common KADS. In *Agent Theories, Architectures, and Languages*, pages 313–327, 1997.

I. Jacobson, M. Christerson, P. Jonsson, and G. Overgaard. Object-oriented software engineering : A use case driven appraoch. *Reading, MA : Addison Wesley*, 1992.

Ivar Jacobson, Grady Booch, and James Rumbaugh. *The Unified Software Development Process*. Addison-Wesley Professional, 1st edition, February 1999.

Mo Jamshidi. Control of large-scale complex systems – from hierarchical to autonomous. presentation, April 2003.

N. R. Jennings, K. Sycara, and M. Wooldridge. A roadmap of agent research and development. *Journal of Autonomous Agents and Multi-Agent Systems*, 1(1) :7–38, 1998.

N.R. Jennings. On agent-based software engineering. *Artificial Intelligence*, 177(2) :277–296, 2000.

N.R. Jennings. An agent-based approach for building complex software systems. *Communications of the ACM*, 44(4) :35–41, April 2001.

Heecheol Jeon, Charles Petrie, and Mark R. Cutkosky. JATLite : A Java Agent Infrastructure with Message Routing. *IEEE Internet Computing*, 04(2) :87–96, 2000. ISSN 1089-7801.

Bin Jiang. SimPed : Simulating pedestrian flows in a virtual urban environment. *Journal of Geographic Information and Decision Analysis*, 3(1) :21–30, 1999.

Thomas Juan, Adrian Pearce, and Leon Sterling. ROADMAP : Extending the GAIA methodology for complex open systems, 2002.

Stuart Kauffman. *At Home in the Universe : The Search for the Laws of Self-Organization and Complexity*. Oxford University Press, USA, October 1996. ISBN 0195111303.

R. Kemp and J. Van den Bergh. Economics and transitions : Lessons from economic sub-disciplines. UNU-MERIT Working Paper Series 038, United Nations University, Maastricht Economic and social Research and training centre on Innovation and Technology, 2006.

Elizabeth A. Kendall. Role modeling for agent system analysis, design, and implementation. *IEEE Concurrency*, 8(2) :34–41, 2000. ISSN 1092-3063.

J.-H. Kim. Micro-robot world cup soccer tournament. A Booklet on MiroSot'96, MiroSot Organizing Committee, KAIST, April 1996.

David Kinny, Michael Georgeff, and Anand Rao. A methodology and modelling technique for systems of BDI agents. In Rudy van Hoe, editor, *Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, Eindhoven, The Netherlands, 1996.

Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, and Eiichi Osawa. RoboCup : The robot world cup initiative. In W. Lewis Johnson and Barbara Hayes-Roth, editors, *Proceedings of the First International Conference on Autonomous Agents (Agents)*, pages 340–347, New York, 1997. ACM Press. ISBN 0-89791-877-0.

Tore Knabe, Michael Schillo, and Klaus Fischer. Inter-organizational networks as patterns for self-organizing multiagent systems. In *Proc. of the second international joint conference on Autonomous agents and multiagent systems (AAMAS)*, pages 1036–1037, New York, NY, USA, 2003. ACM Press. ISBN 1-58113-683-8.

Arthur Koestler. *The Ghost in the Machine*. Hutchinson, 1967.

Fred Kofman. Holons, heaps and artifacts (and their corresponding hierarchies), January 2001. URL `www.integralworld.net/kofman.html`.

Manuel Kolp, Paolo Giorgini, and John Mylopoulos. Multi-agent architectures as organizational structures. *Autonomous Agents and Multi-Agent Systems*, 13(1) :3–25, 2006. ISSN 1387-2532.

B.B. Kristensen and K. Osterbye. Roles : Conceptual abstraction theory and practical language issues. *Theory and Practice of Object Systems (TAPOS), Special Issue on Subjectivity in Object-Oriented Systems*, 2(3) : 143–160, 1996.

Pei Pei Kuan, Shanika Rarunasekera, and Leon Sterling. Improving goal and role oriented analysis for agent based systems. In *Proc. of the Australian conference on Software Engineering (ASWEC)*, pages 40–47, Washington, DC, USA, 2005. IEEE Computer Society.

James J. Kuffner, Jr, and Jean-Claude Latombe. Fast synthetic vision, memory, and learning models for virtual humans. In *CA'99 : Proceedings of the Computer Animation*, page 118, Washington, DC, USA, 1999. IEEE Computer Society. ISBN 0-7695-0167-2.

Danny B. Lange and Oshima Mitsuru. *Programming and Deploying Java Mobile Agents Aglets*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1998. ISBN 0201325829.

Marc Lhuillier. *Une approche à base de composants logiciels pour la conception d'agents. Principes et mise en oeuvre à travers la plateforme Maleva*. PhD thesis, Université de Paris 06, 1998.

J. Li, W. Ge, J. Zhang, and M. Kwauk. Multi-scale compromise and multi-level correlation in complex systems. In *A6 Special Issue : 7th World Congress of Chemical Engineering*, volume 83 of *Chemical Engineering Research and Design (ChERD)*, pages 574–582. Institution of Chemical Engineers, June 2005.

Jurgen Lind. *Iterative Software Engineering for Multiagent Systems : The Massive Method*. Springer-Verlag, Secaucus, NJ, USA, 2001. ISBN 3540421661.

Michael Luck and Mark d'Inverno. Conceptual framework for agent definition and development. *The Computer Journal*, 44(1) :1–20, 2001.

Michael Luck and Mark d'Inverno. A formal framework for agency and autonomy. In Victor Lesser and Les Gasser, editors, *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS)*, pages 254–260, San Francisco, CA, USA, 1995. AAAI Press.

B. I. Lundqvist, A. Bogicevic, S. Dudiy, P. Hyldgaard, S. Ovesson, C. Ruberto, E. Schröder, and G. Wahnström. Bridging between micro- and macroscales of materials by mesoscopic models. *Computational Materials Science*, 24(1), 2002.

Laurent Magne, Sylvestre Rabut, and Jean-François Gabard. Towards an hybrid macro-micro traffic flow simulation model. In *INFORMS Conference*, Salt Lake city, Utah, U.S.A, May 2000.

P. Marcenac. Modélisation de systèmes complexes par agents. *Techniques et Sciences Informatiques*, 16(8), 1997.

P. Marcenac and S. Giroux. Geamas : A generic architecture for agent-oriented simulations of complex processes. *Applied Intelligence*, 8(3) :247–267, May 1998.

Pierre Marcenac and Stéphane Calderoni. Self-organisation in agent-based simulation. In Walter Van de Velde Magnus Broman and Staffan Hägg, editors, *Poster Proceedings of the 8th European Workshop of Modelling Autonomous Agents in a MultiAgent World*, pages 116–131, Ronneby, Sweden, May 1997. Springer Verlag.

Katalin Martinas. Neumannian economy in multi-agent approach. investigation of stability and instability in economic growth. *Interdisciplinary Description of Complex Systems*, 2(1) :70–78, 2004.

Philippe Mathieu, Jean-Christophe Routier, and Yann Secq. Rio : Roles, interactions and organizations. In *3rd International/Central and Eastern European Conference on Multi-Agent Systems (CEEMAS)*, pages 147–157, june 2003.

F. Maturana. *MetaMorph : an adaptive multi-agent architecture for advanced manufacturing systems*. PhD thesis, The University of Calgary, 1997.

F. Maturana, W. Shen, and D. Norrie. Metamorph : An adaptive agent-based architecture for intelligent manufacturing, 1999.

D. McFarlane and S. Bussmann. *Agent Based Manufacturing - Advances In the Holonic Approach*, chapter Holonic Manufacturing Control : Rationales, Developments And Open Issues, pages 301–326. Springer-Verlag, deen, s. m. edition, 2002.

Fabien Michel. Le modèle influence/réaction pour la simulation multi-agents. In *Premières Journées Francophones Modèles Formels de l'Interaction (MFI)*, Toulouse, France, Mai 2001.

Fabien Michel. *Formalism, tools and methodological elements for the modeling and simulation of multi-agents systems*. PhD thesis, LIRMM, Montpellier, France, December 2004.

Fabien Michel. Le modèle irm4s : le principe influence/réaction pour la simulation de systèmes multi-agents. In *Journées Francophones sur les Systèmes Multi-Agents (JFSMA)*, 2006.

Nelson Minar, Roger Burkhart, Christopher G. Langton, and Manor Askenazi. The Swarm Simulation System : A Toolkit for Building Multi-Agent Simulations. Technical Report 96-06-042, Santa Fe Institute,, Santa Fe, Nouveau Mexique, US, June 1996.

Nelson Minar, Matthew Gray, Oliver Roup, Raffi Krikorian, and Pattie Maes. Hive : Distributed Agents for Networking Things. In *First International Symposium on Agent Systems and Applications (ASA)/Third International Symposium on Mobile Agents (MA)*, Palm Springs, CA, USA, 1999.

Pavlos Moraitis and Nikolaos Spanoudakis. The GAIA2JADE process for multi-agent systems development. *Applied Artificial Intelligence*, 20(2-4) :251–273, February-April 2006.

S. Raupp Musse and D. Thalmann. Hierarchical model for real time simulation of virtual human crowds. In *IEEE Trans. on Visualization and Computer Graphics*, volume 7, pages 152–164, 2001.

Glenford J. Myers. *The Art of Software Testing*. John Wiley and Sons Ltd., 2nd edition, 2004. ISBN 0471043281. Revised and updated by Tom Badgett and Todd Thomas, with Corey Sandler.

Allen Newel. The knowledge level. *Artificial Intelligence*, 18(1) :87–127, January 1982.

Hansrudi Noser, Olivier Renault, Daniel Thalmann, and Magnenat-Thalmann Nadia. Navigation for digital actors based on synthetic vision, memory, and learning. *Computers and Graphics*, 19(1) :7–19, 1995.

H. Nwana, D. Ndumu, and L. Lee. ZEUS : A collaborative agents toolkit. In *Third International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*, pages 377–392, London, UK, 1998.

MAF, 2000. *Mobile Agent Facility Specification, v1.0, formal/00-01-02*. Object Management Group (OMG), january 2000.

MDA, 2003. *MDA Guide, v1.0.1, OMG/2003-06-01*. Object Management Group (OMG), June 2003.

OCL, 2006. *Object Constraint Language (OCL) Specification, v2.0, OMG Available Specification, formal/06-05-01*. Object Management Group (OMG), May 2006.

SPEM, 2002. *Software Process Engineering Metamodel Specification, v1.0, Adopted Specification, formal/02-11-14*. Object Management Group (OMG), November 2002.

SPEM, 2007. *Software Process Engineering Metamodel Specification, v2.0, Final Adopted Specification, ptc/07-03-03*. Object Management Group (OMG), March 2007.

UML, 2003. *Unified Modeling Language (UML) Specification : Infrastructure, v2.0, OMG Adopted Specification, ptc/03-09-15*. Object Management Group (OMG), September 2003.

UML, 2007. *Unified Modeling Language : Superstructure, v2.1.1, formal/2007-02-03*. Object Management Group (OMG), February 2007.

Michel Occello, Christof Baeijs, Yves Demazeau, and Jean-Luc Koning. MASK : An AEIO toolbox to design and build multi-agent systems. In Cuena et al, editor, *Knowledge Engineering and Agent Technology*, IOS Series on Frontiers in AI and Applications, Amsterdam, Netherlands, 2004.

J. Odell. Objects and agents compared. *Journal of Object Technology*, 1(1) :41–53, 2002a.

J. Odell, H. Parunak, and B. Bauer. Extending UML for agents. In E. Y. Gerd Wagner and Yves Lesperance, editors, *Information Systems Workshop at the 17th National conference on Artificial Intelligence*, pages 3–17, 2000.

J. Odell, H.V.D. Parunak, M. Fleischer, and S. Breuckner. Modeling agents and their environment. *Agent-Oriented Software Engineering (AOSE) III*, 2585 :16–31, 2002.

James Odell. Agents and complex systems. *Journal of Object Technology*, 1(2) :35–45, July-August 2002b.

James Odell, H. Van Dyke Parunak, and B. Bauer. Representing agent interaction protocols in UML. In P. Ciancarini and M. Wooldridge, editors, *Agent-Oriented Software Engineering : First International Workshop, AOSE 2000, Revised Papers*, volume 1957 of *LNCS*, pages 201–218, Limerick, Ireland, 2001. Springer Berlin / Heidelberg.

James Odell, Marian Nodine, and Renato Levy. A metamodel for agents, roles, and groups. In James Odell, P. Giorgini, and Jörg Müller, editors, *Agent-Oriented Software Engineering (AOSE) IV*, Lecture Notes on Computer Science. Springer, 2005.

A. Omicini and E. Denti. From tuple spaces to tuple centres. *Science of Computer Programming*, 41(3) : 277–294, Nov. 2001.

Andrea Omicini. SODA : Societies and infrastructures in the analysis and design of agent-based systems. In Springer-Verlag, editor, *Agent-Oriented Software Engineering : First International Workshop, AOSE*, volume 1957 of *Lecture Notes in Computer Science*, pages 185–193, 2000.

ISO/IEC 2382-20, 1990. *Technologies de l'information - Vocabulaire - Partie 20 : Développement de système*. Organisation internationale de normalisation, 1990.

Q Feng P. Eades. Multilevel visualition of clustered graphs. Technical report 96-09, University of NewCastle, Australia, Sept 1996.

Lin Padgham and Michael Winikoff. Prometheus : A methodology for developing intelligent agents. In *Third International Workshop on Agent-Oriented Software Engineering*, July 2002a.

Lin Padgham and Michael Winikoff. Prometheus : A pragmatic methodology for engineering intelligent agents. In *Workshop on Agent-Oriented Methodologies (OOPSLA)*, pages 97–108, Seattle, November 2002b.

H. Van Dyke Parunak, Sven Brueckner, and John Sauter. Digital pheromone mechanisms for coordination of unmanned vehicles. In *First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 449–450, New York, NY, USA, 2002. ACM Press. ISBN 1-58113-480-0.

H.V.D Parunak and S. Brueckner. Entropy and self-organization in multi-agent systems. In *Autonomous Agents*, pages 124–130, 2001.

J. Pavón, J. Gómez-Sanz, and R. Fuentes. The INGENIAS methodology and tools. In Brian Henderson-Sellers and Paolo Giorgini, editors, *Agent-Oriented Methodologies*, pages 236–276. Idea Group Publishing, NY, USA, juin 2005. ISBN 1-59140-581-5.

Juan Pavón. *INGENIAS : Développement Dirigé par Modèles des Systèmes Multi-Agents*. Dossier d'habilitation à diriger des recherches, Université Pierre et Marie Curie, et Universidad Complutense Madrid, Paris, France et Madrid, Espagne, Décembre 2006.

Juan Pavón and Jorge Gómez-Sanz. Agent Oriented Software Engineering with INGENIAS. In *Multi-Agent Systems and Applications III : 3rd International Central and Eastern European Conference on Multi-Agent Systems (CEEMAS)*, volume 2691 of *LNCS*, pages 394–403. Springer Berlin Heidelberg, June 2003.

Julien Perret. *Modélisation d'environnements urbains virtuels*. PhD thesis, Université de Rennes 1, nov 2006.

F. Peschanski. COMET : A component-based reflective architecture for distributed programming. In *First OOPSLA Workshop on Object-Orientation for Software Engineering*, Denver, USA, November 1999.

Frédéric Peschanski. COMET : Architecture réflexive à base de composants pour la construction d'applications concurrentes et réparties. In *Langages et modèles à objet 2000*, pages 11–26. Hermès, 2000.

G.C. Pettinaro, I.W. Kwee, and L.M. Gambardella. Acceleration of 3D dynamics simulation of s-bot mobile robots using multi-level model switching. Technical Report IDSIA-20-03, IDSIA/USI-SUPSI, Switzerland, November 2003.

Gauthier Picard. *Méthodologie de développement de systèmes multi-agents adaptatifs et conception de logiciels à fonctionnalité émergente*. PhD thesis, Université Paul Sabatier de Toulouse III, 2004.

S. Poslad, P. Buckle, and R. Hadingham. FIPA-OS : the FIPA agent platform available as open source. In *Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM)*, pages 355–368, 2000.

M.J. Prietula, K.M. Carley, and L.E. Gasser. *Simulating Organizations : Computational Models of Institutions and Groups*. AAAI Press, 1998.

L.S.C. Pun-Cheng. A new face-entity concept for modeling urban morphology. *Journal of Urban and Regional Information Systems Association*, 12(3) :47–56, 2000.

Anand S. Rao. Agentspeak(l) : Bdi agents speak out in a logical computable language. In *MAAMAW*, pages 42–55, 1996.

O. Renault, N. Thalmann, and D. Thalmann. A vision-based approach to behavioral animation. *Visualization and Computer Animation*, 1 :18–21, 1990.

Reticular. AgentBuilder : An integrated toolkit for constructing intelligent software agents. white paper edition, Reticular Systems Inc, 1999. http ://www.agentbuilder.com.

Pierre-Michel Ricordel. *Programmation Orientée Multi-Agents : Développement et Déploiement de Systèmes Multi-Agents Voyelles*. PhD thesis, INPG, Grenoble, France, Octobre 2001.

Pierre-Michel Ricordel and Yves Demazeau. Volcano, a Vowels-Oriented Multi-agent Platform. In *CEEMAS'01 : Revised Papers from the Second International Workshop of Central and Eastern Europe on Multi-Agent Systems*, pages 253–262, London, UK, 2002a. Springer-Verlag. ISBN 3-540-43370-8.

Pierre-Michel Ricordel and Yves Demazeau. La plate-forme VOLCANO - modularité et réutilisation pour les systèmes multi-agents. *Technique et Science Informatiques (TSI)*, 21(4) :447–471, 2002b.

S. Rodriguez, V. Hilaire, and A. Koukam. Towards a methodological framework for holonic multi-agent systems. In *Workshop of Engineering Societies in the Agents World*, pages 179–185, 2003.

S. Rodriguez, V. Hilaire, and A. Koukam. Formal specification of holonic multi-agent system framework. In *Intelligent Agents in Computing Systems, ICCS(3)*, number 3516 in LNCS, pages 719–726, 2005a.

S. Rodriguez, N. Gaud, V. Hilaire, S. Galland, and A. Koukam. An analysis and design concept for self-organization in holonic multi-agent systems. In S. Bruckner, S. Hassas, M. Jelasity, and D. Yamins, editors, *Engineering Self-Organising Systems*, volume 4335 of *LNAI*, pages 15–27. Springer-Verlag, 2007.

Sebastian Rodriguez. *From analysis to design of Holonic Multi-Agent Systems : A Framework, Methodological Guidelines and Applications*. PhD thesis, Université de Technologie de Belfort-Montbéliard, December 2005.

Sebastian Rodriguez, Vincent Hilaire, and Abder Koukam. Multi-agent coordination is arbitration from a super-holon point of view. In S. Hassas and G. Di Marzo Serugendo, editors, *European Workshop on Multi-Agent For Modeling Complex Systems (MA4CS)*, November 2005b.

C. Rolland, N. Prakash, and A. Benjamen. A multi-model view of process modelling. *Requirements Engineering*, 4 :169–187, 1999.

Colette Rolland and Naveen Prakash. A proposal for context-specific method engineering. In *Proceedings of the IFIP TC8, WG8.1/8.2 working conference on method engineering on Method engineering : principles of method construction and tool support*, pages 191–208, Atlanta, Georgia, United States, 1996. Chapman and Hall, Ltd. ISBN 0-412-79750-X.

Stuart J. Russell and Peter Norvig. *Artificial Intelligence : A Modern Approach*. Prentice Hall, (second edition 2003), 1st edition, January 1995. ISBN 0137903952.

Hiroshi Sato and Akira Namatame. Finding micro-macro loop with market game. In *ICCIMA'01 : Proceedings of the Fourth International Conference on Computational Intelligence and Multimedia Applications*, page 236, Washington, DC, USA, 2001. IEEE Computer Society. ISBN 0-7695-1312-3.

Keith R. Sawyer. Artificial societies : Multiagent systems and the micro-macro link in sociological theory. *Sociological Methods Research*, 31(3) :325–363, February 2003.

Edson Scalabrin and Jean-Paul Barthès. An environment for building cognitive agents for cooperative work. In *2nd International Workshop on Computer Supported Cooperative Work in Design*, Bangkok, Thailand, Nov 1997.

Thorsten Schelhorn, David O'Sullivan, Mordechay Haklay, and Mark Thurstain-Goodwin. Streets : an agent-based pedestrian model. CASA Working Papers 9, Centre for Advanced Spatial Analysis UCL, London, UK, 1999.

M. Schillo. *Multiagent Robustness : Autonomy vs. Organisation.* PhD thesis, Department of Computer Science, Universität des Saarlandes, 2004.

Michael Schillo, Klaus Fischer, and Christof T. Klein. The micro-macro link in DAI and sociology. In *Second international workshop on Multi-agent based simulation (MABS)*, pages 133–148, Secaucus, NJ, USA, 2001. Springer-Verlag New York, Inc. ISBN 354041522.

Douglas C. Schmidt. Model Driven Engineering. *IEEE Computer*, 39(2) :25–31, February 2006.

Gero Schwenk. Micro-macro relations in the kirk-coleman model, 2004.

J.R. Searle. *Speech Acts*. Cambridge University Press, Cambridge, UK, 1969.

Yann Secq. *RIO : Rôles, Interactions et Organisations, une méthodologie pour les systèmes multi-agents ouverts*. PhD thesis, Université des Sciences et Technologies de Lille, Décembre 2003.

A. El Fallah Seghrouchni and A. Suna. Claim : A computational language for autonomous, intelligent and mobile agents. In M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors, *First International Workshop on Programming MAS (ProMAS)*, volume 3067 of *LNCS*, pages 90–110. Springer Verlag, 2004.

V. Seidita, M. Cossentino, and S. Gaglio. A repository of fragments for agent systems design. In *Workshop on Objects and Agents (WOA)*, Catania, Italy, September 2006.

J. M. Serrano and S. Ossowski. On the impact of agent communication languages on the implementation of agent systems. In Springer-Verlag, editor, *8th International Workshop on Cooperative Information Agents (CIA)*, volume 3191 of *Lecture Notes in Computer Science*, pages 92–106, 2004.

D. Servat, E. Perrier, J.P. Treuil, and A. Drogoul. When agents emerge from agents : Introducing multi-scale viewpoints in multi-agent simulations. In *MABS'98*, pages 1–16, Paris, France, July 1998.

Joshua Shagam. *Dynamic Spatial partitioning for real-time visibility determination*. Msc, New Mexico State University, Department of Computer Science, April 2003.

Robert E. Shannon. Simulation modeling and methodology. *SIGSIM Simul. Dig.*, 8(3) :33–38, 1977. ISSN 0163-6103.

Wei Shao and Demetri Terzopoulos. Autonomous pedestrians. In *SCA'05 : Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 19–28, New York, NY, USA, 2005. ACM Press. ISBN 1-7695-2270-X.

Weiming Shen and Jean-Paul Barthès. An experimental multi-agent environment for engineering design. *International Journal of Cooperative Information Systems*, 5(2,3) :131–151, 1996.

Y. Shoham. Agent-oriented programming. *Artificial Intelligence*, 60(1) :51–92, 1993.

Carles Sierra, Juan Antonio Rodríguez-Aguilar, Pablo Noriega, Marc Esteva, and Josep Lluís Arcos. Engineering multi-agent systems as electronic institutions. *European Journal for the Informatics Professional*, 4 :33–39, August 2004.

Carla Silva, Rosa Pinto, Jaelson Castro, and Patricia Tedesco. Requirements for multi-agent systems. In *6th International Workshop on Requirements Engineering (WER)*, pages 198–212, Piracicaba, 2003.

Herbert A. Simon. *The Science of Artificial*. MIT Press, Cambridge, Massachusetts, 3rd edition, 1996. ISBN 0-262-19374-4.

B. Singh. Interconnected Roles (ir) : A Coordination Model. Technical Report CT-084-92, MCC, July 1992.

Ian Sommerville. *Software Engineering*. International Computer Science Series. Addison Wesley, Pearson Education, seventh edition, 2004. ISBN 0-321-21026-3.

J.C. Soulie, P. Marcenac, S. Calderoni, and R. Courdier. Geamas v2.0 : an object oriented platform for complex systems simulations. In *Technology of Object-Oriented Languages*, pages 230–242, Aug 1998.

Tiberiu Stratulat. *Systèmes d'agents normatifs : concepts et outils logiques*. PhD thesis, Université de Caen, 2002.

V.S. Subrahmanian, Piero Bonatti, Jürgen Dix, Thomas Eiter, Sarit Kraus, Fatma Özcan, and Robert Ross. *Heterogenous Active Agents*. MIT-Press, 2000.

K. Sycara, M. Klusch, S. Widoff, and J. Lu. Dynamic service matchmaking among agents in open information environments. *SIGMOD Record (ACM Special Interests Group on Management of Data)*, 28(1) :47–53, March 1999a.

K. Sycara, J. Lu, M. Klusch, and S. Widoff. Matchmaking among heterogeneous agents on the internet. In *Proceedings of the 1999 AAAI Spring Symposium on Intelligent Agents in Cyberspace*, March 1999b.

Kardi Teknomo, Yasushi Takeyama, and Hajime Inamura. Review on microscopic pedestrian simulation model. In *Japan Society of Civil Engineering Conference*, Morioka, Japan, March 2000.

Kardi Teknomo, Yasushi Takeyama, and Hajime Inamura. Microscopic pedestrian simulation model to evalute lane-like segregation of pedestrian crossing. In *Infrastructure Planning Conference*, volume 24, Kouchi, Japan, November 2001.

Demetri Terzopoulos and Tamer F. Rabie. Animat vision : Active vision in artificial animals. In *Fifth International Conference on Computer Vision (ICCV)*, pages 801–808, June 1995.

Gwenola Thomas. *Environnements virtuels urbains : modélisation des informations nécessaires à la simulation de piétons*. PhD thesis, Université de Rennes, IFSIC/IRISA, 1999.

I. Trencansky and R. Cervenka. Agent Modeling Language (AML) : A comprehensive approach to modeling MAS. *Informatica*, 29(4) :391–400, 2005.

K.G. Troitzsch. Multilevel simulation. In K.G. Troitzsch, U. Mueller, N. Gilbert, and J.E. Doran, editors, *Social Science Microsimulation*. Springer-Verlag, Berlin, 1996.

Adelinde M. Uhrmacher and Corrado Priami. Discrete event systems specification in systems biology - a discussion of stochastic pi calculus and devs. In *WSC'05 : Proceedings of the 37th conference on Winter simulation*, pages 317–326. Winter Simulation Conference, 2005. ISBN 0-7803-9519-0.

Adelinde M. Uhrmacher, Daniela Degenring, and Bernard Zeigler. Discrete event multi-level models for systems biology. In C. Priami et al., editor, *Transactions on Computational Systems Biology I*, volume 3380 of *LNCS*, pages 66–89. Springer-Verlag Berlin Heidelberg, 2005.

M. Ulieru and A. Geras. Emergent holarchies for e-health applications : a case in glaucoma diagnosis. In *IEEE IECON'02*, volume 4, pages 2957–2961, 2002.

F. Van Aeken. *Les systèmes multi-agents minimaux*. PhD thesis, Leibniz / IMAG, Institut National Polytechnique de Grenoble - INPG., 1999.

H. Van Dyke Parunak, Robert Savit, and Rick L. Riolo. Agent-based modeling vs. equation-based modeling : A case study and users' guide. In J.S. Sichman, R. Conte, and N. Gilbert, editors, *Multi-Agent Systems and Agent-Based Simulation (MABS)*, pages 10–26, Paris, France, 1998. Springer Verlag.

Axel Van Lamsweerde. Goal-oriented requirements engineering : A guided tour. In *5th IEEE International Symposium on Requirements Engineering (RE)*, pages 249–263, Toronto, Canada, August 2001. IEEE Press.

L. Vercouter. MAST : Un modèle de composants pour la conception de SMA. In *1ère Journée Multi-Agents et Composants (JMAC)*, Paris, France, Novembre 2004.

Ludwig Von Bertalanffy. *General System Theory : Foundations, Developments, Applications*. George Braziller, New York, revised edition (march 1976), first edition, 1968.

Gerd Wagner. The agent-object-relationship metamodel : towards a unified view of state and behavior. *Information Systems*, 28(5) :475–504, 2003. ISSN 0306-4379.

Zhigang Wen and N.E. Mehdi, Q.H. Gough. A new animation approach for visualizing intelligent agent behaviours in virtual environment. In *Sixth International Conference on Information Visualisation (IV)*, pages 93–98. IEEE Computer Society Press, July 2002.

D. Weyns, H.V.D. Parunak, and F. Michel, editors. *Environments for Multi-Agent Systems (E4MAS), First International Workshop*, volume 3374 of *LNCS*. Springer Berlin / Heidelberg, 2005. ISBN 978-3-540-24575-9.

D. Weyns, H.V.D. Parunak, and F. Michel, editors. *Environments for Multi-Agent Systems II (E4MAS II), Second International Workshop*, volume 3830 of *LNCS*. Springer Berlin / Heidelberg, 2006. ISBN 3-540-32614-6.

D. Weyns, A. Omicini, and J. Odell. Environment as a first-class abstraction in multiagent systems. *Journal on Autonomous Agents and Multiagent Systems*, 14(1), 2007.

Danny Weyns and Tom Holvoet. Formal model for situated multi-agent systems. *Formal Approaches for Multi-agent Systems, Special Issue of Fundamenta Informaticae*, 63(2-3), 2004. Eds. B. Dunin-Keplicz, R. Verbrugge.

Danny Weyns, Elke Steegmans, and Tom Holvoet. Towards active perception in situated multi-agent systems. *EUMAS, Special Issue of Journal on Applied Artificial Intelligence (AAI)*, 18(9-10) :867–883, October–December 2004.

Ken Wilber. *Sex, Ecology, Spirituality*. Shambhala, 1995.

M. Wooldridge. *The Logical Modelling of Computational Multi-Agent Systems*. PhD thesis, Department of Computation, UMIST, Manchester, UK, October 1992.

M. Wooldridge. Agent-based software engineering. *IEEE Transactions on Software Engineering*, 144(1) : 26–37, February 1997.

M. Wooldridge and P. Ciancarini. Agent-oriented software engineering : The state of the art. In *Agent-Oriented Software Engineering : First International Workshop (AOSE)*, volume 1957 of *Lecture Notes in Computer Science*, pages 1–28. Springer-Verlag, 2001.

Michael Wooldridge and Nicholas R. Jennings. Intelligent Agents : Theory and Practice. *Knowledge Engineering Review*, 10(2) :115–152, 1995.

Michael Wooldridge, Nicholas R. Jennings, and David Kinny. The GAIA methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems*, 3(3) :285–312, 2000. ISSN 1387-2532.

Jianguo Wu. Hierarchy and scaling : Extrapolating information along a scaling ladder. *Canadian Journal of Remote Sensing*, 25(4) :367–380, 1999.

J. Wyns. *Reference architecture for Holonic Manufacturing Systems - the key to support evolution and reconfiguration*. PhD thesis, Katholieke Universiteit Leuven, 1999.

Dongbo Xiao and Roger Hubbold. Navigation guided by artificial force fields. In *CHI'98 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 179–186, New York, NY, USA, 1998. ACM Press/Addison-Wesley Publishing Co. ISBN 0-201-30987-4.

F. Zambonelli, N. Jennings, and M. Wooldridge. Developing multiagent systems : the GAIA methodology. *ACM Trans. on Software Engineering and Methodology*, 12(3), 2003.

Bernard P. Zeigler, Tag Gon Kim, and Herbert Praehofer. *Theory of Modeling and Simulation*. Academic Press, 2nd edition edition, 2000.

Athanasios Ziliaskopoulos, Jiang Zhang, and Huajing Shi. Hybrid mesoscopic-microscopic traffic simulation model : Design, implementation, and computational analysis. In *Transportation Research Board 85th Annual Meeting*, 2006.

Holonic Multi-Agent Systems : From the analysis to the implementation. Metamodel, Methodology and Multilevel simulation.

The work, presented in this PhD thesis, is concerned with the study of complex systems and aims at providing a full set of abstractions and the associated methodological guidelines for the analysis, design, implementation and simulation of Holonic MultiAgent Systems (HMAS). HMAS offers a promising software engineering approach for developing complex open software systems. This kind of systems consists in self-similar structures called holons. A set of holons may be seen, depending on the level of observation, as a unique entity or as a group of holons in interaction. A complex system is made up of a large number of parts that have many interactions. In such systems, the behavior of the whole cannot be directly understood only by knowing the behavior of the parts and their interactions. Complex systems often exhibit a hierarchical structure. The foundation of this thesis consist in exploiting the intrinsic hierarchical structure of complex systems to analyse and decompose them. In order to conceive modular and reusable models, an organizational approach is adopted. The principle of the analysis is based on the identification of a hierarchy of organizations, which the global behavior may represent the system under the chosen perspective. The behaviors of the system are recursively decomposed into a set of interacting sub-behaviors, each of these latter being in turn decomposed until we reach some lowest level of elementary sub-behaviors. At a given level, the composed behavior is modeled using an organization, and the associated sub-behaviors using roles. This hierarchical organization structure is then mapped to an holarchy (hierarchy of holons) in charge of its execution. The concepts presented are then used to study the issues related to the multilevel multiagent simulation. The resulting model is finally applied to the pedestrians simulation in virtual urban environment.

Keywords : Holonic Multi-Agent Systems, Agent-Oriented Software Engineering, Complex systems, Organizational approach, Multilevel simulation.

Systèmes Multi-Agents Holoniques : De l'analyse à l'implantation. Méta-modèle, Méthodologie, et Simulation multi-niveaux.

Cette thèse propose un guide méthodologique pour l'analyse, la conception, l'implantation et la simulation des Systèmes Multi-Agents Holoniques (SMAH). Ce type de système repose sur une structure hiérarchique auto-similaire, une structure gigogne où les agents sont composés d'agents. La brique de construction de tels systèmes est nommée *Holon*. Un holon est une entité qui, selon le niveau d'observation, peut être vu, soit comme une partie composante d'un élément de niveau supérieur, soit comme un tout composé d'autres holons. Les SMAH sont utilisés pour analyser les systèmes considérés comme complexes. Ces derniers exhibent généralement une structure hiérarchique où le système est composé de sous-systèmes qui, à leur tour, ont leurs propres sous-systèmes. L'approche adoptée dans cette thèse consiste à exploiter la nature intrinsèquement hiérarchique des systèmes complexes pour les analyser et les modéliser. Afin de concevoir des modèles modulaires et réutilisables, une approche organisationnelle est adoptée. Le principe de l'analyse repose sur l'identification d'une hiérarchie d'organisations, dont le comportement global est en mesure de représenter le système selon une certaine perspective. Les comportements du système sont récursivement décomposés en un ensemble de sous-comportements en interaction, chacun d'entre-eux étant à son tour décomposé jusqu'à atteindre un niveau où les comportements correspondants peuvent être considérés comme élémentaires. À un niveau donné, le comportement composé est représenté par une organisation, et les sous-comportements associés par des rôles. Cette hiérarchie d'organisations est ensuite projetée sur une holarchie (hiérarchie de holons) en charge de lui donner vie et d'exécuter les comportements qui la composent. En sus de la modélisation des systèmes complexes, cette thèse aborde également les problématiques liées à leur simulation. Simuler précisément de tels systèmes requiert généralement d'importantes ressources de calcul. La simulation multi-niveaux permet d'obtenir un compromis entre la précision de la simulation et les ressources de calcul disponibles. L'approche défendue dans ce manuscrit exploite les propriétés des SMAH pour concevoir un modèle de simulation multi-agents multi-niveaux, lequel est ensuite appliqué à la simulation de piétons en environnements urbains virtuels.

Mots-clés : Systèmes Multi-Agents Holoniques, Génie logiciel, Systèmes complexes, Approche organisationnelle, Simulation multi-niveaux.