Du semi-formel au formel : une Approche Organisationnelle pour l'Ingénierie de Systèmes Multi-Agents

Vincent Hilaire

Du semi-formel au formel : une Approche Organisationnelle pour l'Ingénierie de Systèmes Multi-Agents

Habilitation à Diriger les Recherches
de l'Université de Franche-Comté
préparée au sein de l' Université de Technologie de Belfort-Montbéliard
Discipline : Informatique

présentée et soutenue publiquement par

Vincent Hilaire

le 8/12/2008
devant le jury composé de :

Olivier Boissier	Rapporteur	Professeur à l'Ecole Nationale des Mines de Saint-Etienne
Marie-Pierre Gleizes	Rapporteur	Professeur à l'Université Paul Sabatier-Toulouse 3
Salima Hassas	Rapporteur	Professeur à l'université Claude Bernard-Lyon 1
Raphaël Couturier	Examinateur	Professeur à l'Université de Franche Comté
François Charpillet	Examinateur	Directeur de Recherche INRIA
Abderrafiâa Koukam	Examinateur	Professeur à l'UTBM
Jean-Pierre Müller	Examinateur	Senior scientist CIRAD

Remerciements

Une Habilitation à Diriger les Recherches est le résultat d'un long parcours tant personnel que professionnel. Il y a donc beaucoup de personnes à remercier et je vais sans doute en oublier, qu'elles m'excusent par avance.
Tout d'abord je tiens à remercier les membres du jury qui ont accepté d'évaluer mon travail.

- Olivier Boissier, pour avoir accepté de rapporter et pour avoir beaucoup apporté aux questions organisationnelles dans les SMA.
- Marie-Pierre Gleizes pour avoir accepté de rapporter et pour avoir animer la communauté française et européenne autour des systèmes auto-organisés.
- Salima Hassas pour avoir accepté de rapporter et pour son engagement autour de la thématique des systèmes complexes.
- Raphaël Couturier pour avoir accepter d'être membre de ce jury même si ses thématiques ne sont pas voisines des SMA.
- François Charpillet pour évaluer ce travail après avoir évalué ma thèse il y a quelques années de cela...
- Abderrafiâa Koukam, il y a beaucoup à dire pour celui qui a dirigé ma thèse et qui a grandement contribué à la construction de l'informatique et de la thématique SMA à l'UTBM et à son expansion. Ses qualités humaines et scientifiques alimentent et guident l'équipe et amènent un des pères des SMA en France à dire : « il se dégage une lumière de cette équipe ».
- Jean-Pierre Müller pour avoir collaboré dès les premières heures et pour avoir animé COLLINE qui a profondément inspiré mes travaux initiaux.

Je tiens à remercier également tous ceux avec qui j'ai pu travailler de près ou de loin durant ces années. Il y a bien sur les doctorants qui m'ont supporté. Sebastian a été le premier et je garderai un souvenir impérissable de notre expérience holonique. Je remercie Davy pour avoir ouvert la voie de la gestion des connaissances par agents à Belfort. Il y a également les permanents de la première heure comme Pablo et Belhassène et l'équipe SMA avec Fabrice,

Franck, Nicolas, Olivier et Stéphane. Bien sur Massimo fait aussi partie de ces remerciements pour sa gentillesse, sa persévérance et les grandes choses que nous avons faites (gages d'un futur productif !). Les stagiaires, master ou autre, qui sont venu apporter une brique a cet édifice, se reconnaîtrons sans doute : Julien, Lina, Lukasz, Rodolfo.

Que serait un laboratoire sans un coin café et ses discussions plus ou moins scientifiques ! Autour de ce lieu beaucoup de souvenirs et j'en remercie David, Mickaël, Philippe, Renan, Sana, Thomas et Yassine.

Bien sur, ma famille sans qui je n'aurais jamais eu la force de tenir jusque là. Mes roudoudous qui ont souvent entendu « Laisse papa tranquille il travaille ! » et Flo qui aurait préféré que je sois plus présent.

Table des matières

Chapitre 1

Introduction

1.1 Contexte

1.1.1 Historique

Après une thèse de doctorat préparée au sein de l'Institut Polytechnique de Sévenans (IPSé) et soutenue en janvier 2000, la suite de mon activité s'est déroulée à l'Université de Technologie de Belfort Montbéliard (UTBM). La thèse a consisté en la proposition d'une méthode de spécification qui permet d'exprimer pleinement et de manière formelle les aspects des SMA afin d'aider à leur développement. Cette méthode de spécification était basée sur le méta-modèle Rôle Interaction et Organisation (RIO). Ce méta-modèle est à la base des travaux postérieurs à la thèse. Ce méta-modèle et la réflexion qui a mené à sa définition ne seront donc pas présentés intégralement dans ce mémoire. L'UTBM est née de la fusion de l'IPSé et de l'Ecole Nationale d'Ingénieurs de Belfort. Cette création a donné lieu à une restructuration de la recherche dans le nord Franche Comté et de nouveaux laboratoires sont apparus. L'équipe de recherche en informatique dont je fais partie est intégrée au laboratoire Systèmes et Transports (SeT). La thématique des Systèmes Multi-Agents est toujours présente dans cette équipe et l'engagement pris est d'orienter nos applications, de façon prioritaire, vers le domaine du transport.

1.1.2 Problématique

Les Systèmes Multi-Agents, que nous désignons par la suite SMA, forment un paradigme prometteur pour la conception de logiciels complexes. En effet, ce paradigme propose de nouvelles stratégies pour analyser, concevoir et implémenter de tels systèmes (Henderson-Sellers and Giorgini, 2005; Bergenti et al., 2004; Ferber, 1997; Jennings et al., 1998). Les systèmes multi-

agents sont considérés comme des sociétés composées d'entités autonomes et indépendantes, appelées agents, qui interagissent en vue de résoudre un problème ou de réaliser collectivement une tâche. Grâce à la généricité de ces concepts, les domaines d'application des SMA sont vastes (Jennings and Wooldridge, 1998). Toutefois, il est illusoire de considérer les SMA comme applicables quelque soit le contexte. Ce fait a été souligné très tôt et notamment dans (Wooldridge and Jennings, 1998). Les SMA peuvent être, par exemple, considérés comme un outil viable pour la modélisation et la simulation de systèmes complexes.

Nous nous plaçons dans un cadre d'ingénierie logicielle pour ce mémoire. Pour nous, un agent est donc une entité logicielle qui possède les caractéristiques suivantes (Wooldridge, 1997) : autonomie, réactivité, proactivité et une certaine habilité sociale. L'autonomie consiste pour les agents en l'encapsulation de leur état interne (inaccessible pour les autres agents) et le fait que chaque agent prend des décisions sur la base de cet état interne. La réactivité signifie que les agents sont situés dans un environnement (monde physique, internet, ...) et sont capables de percevoir cet environnement. De plus ils peuvent évoluer en fonction de changements qui interviennent dans cet environnement. La proactivité implique que le comportement des agents n'est pas une simple réponse aux changements de leur environnement. En fait, ils poursuivent leur(s) propre(s) but(s). Enfin, l'habilité sociale indique que chaque agent peut interagir avec d'autres agents et mettre en place des processus de coopération ou négociation pour accomplir leurs buts.

Comme souligné par les auteurs de (Zambonelli et al., 2003) pour qu'un nouveau paradigme d'ingénierie logicielle soit pleinement appliqué et déployé il faut disposer de nouveaux modèles et d'abstractions nouvelles. Ces abstractions servent de base à l'analyse et à la conception des SMA. De plus, toute méthodologie[1] dédiée aux SMA doit prendre en compte ces abstractions pour pouvoir développer des SMA de manière sure, robuste et fiable. Pour répondre à la problématique d'ingénierie à base de SMA trois types d'approches ont été adoptés. Le premier type d'approche consiste en la définition de systèmes et/ou des architectures d'agents spécifiques à un problème ou une classe de problèmes comme signalé dans (Ferber and Gutknecht, 1998). Le deuxième type d'approche consiste à utiliser les abstractions et méthodologies existantes, notamment celles dédiées au paradigme objet (Bergenti and Poggi, 2000; Iglesias et al., 1999). Enfin, le troisième type d'approche consiste à définir des abstractions appropriées pour les SMA. Le principe général de la

[1]Le terme méthodologie est un abus de langage. En français il faudrait dire méthode. La communauté SMA emploie fréquemment le terme méthodologie en référence au terme anglais methodology.

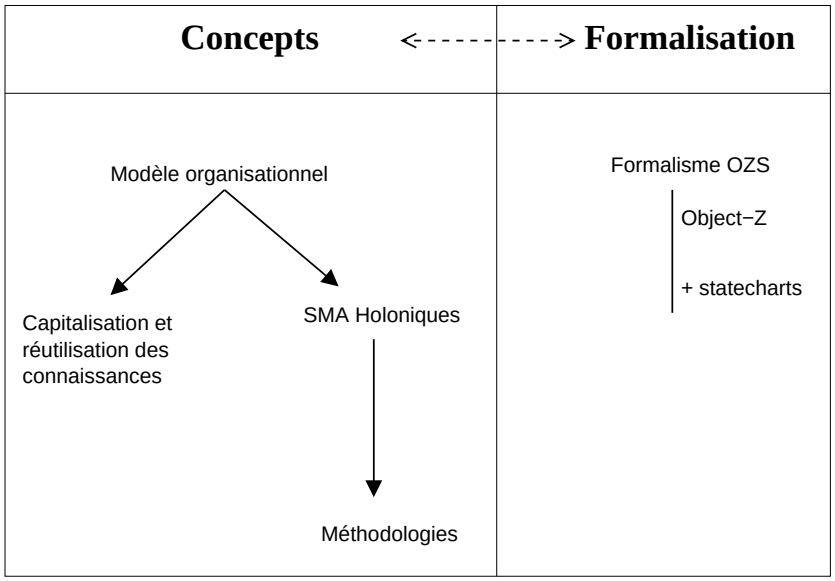


FIG. 1.1 – Aperçu des contributions

plupart de ces approches est de concevoir les SMA comme des société d'individus qui jouent des rôles dans des organisations. Nos travaux se situent dans ce dernier type d'approche. En effet, le modèle à la base de tous les travaux dans ce mémoire est composé de concepts organisationnels.

1.2 Contributions

Les objectifs de nos travaux sont guidés par la problématique de l'ingénierie de SMA. La figure 1.1 schématise l'organisation de ce mémoire et les différentes contributions présentées. Deux perspectives sont à distinguer. La première désignée par **Concepts** regroupe les concepts organisationnels à la base de nos travaux et leurs utilisations pour la capitalisation et la réutilisation des connaissances, la proposition d'une approche pour SMA Holoniques et la définition de deux méthodologies essentiellement dédiées à ce dernier type de système. La deuxième perspective, désignée par **Formalisation**, associe aux concepts de la première perspective un pendant exprimé avec le formalisme OZS. Cette relation entre semi-formel et formel est primordiale dans nos travaux car elle permet à partir d'approches semi-formelles de bénéficier d'outils de validation et de vérification. Le chapitre concernant le formalisme

OZS est donc présenté en premier lieu. En effet, OZS est ensuite utilisé, tout au long du mémoire, dans les chapitres concernant les aspects relatifs aux concepts.
Pour définir une sémantique non ambigüe et bénéficier d'outils de validation et de vérification, nous proposons de spécifier des concepts au travers d'un langage formel. Pour prendre en compte les caractéristiques des SMA, nous avons proposé un langage multi-formalismes, composé de deux langages existants : Object-Z (Duke et al., 1991) et les statecharts (Harel, 1988). Notre contribution consiste en la définition d'une syntaxe qui permet d'écrire des spécifications en utilisant les deux notations. De plus, une sémantique a été définie pour des spécifications syntaxiquement correcte. Cette sémantique, exprimée en terme de systèmes de transitions (Manna and Pnueli, 1991), est dite opérationnelle car elle permet une exécution des spécifications. Concernant la validation et la vérification, nous avons montré au travers de plusieurs outils logiciels la faisabilité de la simulation ou de l'animation de spécifications ainsi que la preuve automatique ou semi-automatique. Pour faciliter la preuve nous avons également défini des techniques d'abstraction qui réduisent la complexité algorithmique.
La définition des modèles organisationnels se focalisent sur la décomposition d'un SMA en unités de comportements en interaction. Ces unités de comportement sont appelées rôles. A partir de ce modèle, nous avons exploré plusieurs pistes. La première est la définition d'une sémantique pour un SMA dont les agents mettent en oeuvre des rôles de manière dynamique. La deuxième est l'utilisation du modèle RIO pour l'étude et la rétro-ingénierie d'architectures d'agents afin d'en comprendre les propriétés et de dégager des éléments réutilisables.
L'approche de capitalisation et de réutilisation des connaissances que nous proposons, est basée sur l'idée de modéliser un processus de conception de produits industriels avec le modèle RIO. Pour cela, RIO a été enrichie par les concepts de connaissance et compétence. Ce modèle permet d'une part d'identifier les connaissances à capitaliser et est le fondement d'une structure de mémoire de projet et d'une ontologie manipulées par un SMA en charge de capitaliser « au fil de l'eau »[2].
L'étude des systèmes complexes nous a mené au paradigme holonique. L'idée de SMA Holonique existait déjà (Gerber et al., 1999). Notre contribution dans ce cadre consiste en la définition d'une approche organisationnelle pour l'ingénierie de SMAH. Pour ce faire, nous avons ajouté le concept de capacité au modèle RIO. La capacité a pour double objectif de décrire ce qu'un comportement, rôle ou organisation, est capable de faire et ce qu'un rôle re-

[2]Au fil de l'eau est employé ici pour signifier durant les projets de conception.

quiert pour définir son comportement. Ces deux aspects duaux permettent de mettre en relation des rôles ou organisations à différents niveaux d'abstraction. Un framework organisationnel a également été défini pour l'ingénierie de SMAH. Pour compléter ce framework dédié à l'analyse de SMAH une architecture de SMAH à base de systèmes immunitaires est définie.
Nous avons également exploré le domaine des méthodologies afin de proposer des approches méthodologiques basées sur les abstractions et le langage formel défini. Ces méthodologies sont essentiellement dédiées aux SMAH. La première est une approche qui permet d'identifier et de construire les parties structurelles et comportementales des SMAH. La deuxième, ASPECS, est une méthodologie complète, de l'analyse à l'implémentation et au déploiement. ASPECS est une méthodologie pas à pas basée sur l'ingénierie dirigée par les modèles.

1.3 Organisation du mémoire

Ce mémoire est organisé en chapitres de la manière suivante.

Le chapitre 2 présente le langage formel OZS et les outils de validation et vérification. Pour ce faire, la syntaxe d'OZS est donnée et la sémantique de ce langage est définie en terme de systèmes de transitions. Cette sémantique opérationnelle permet d'utiliser des outils de validation et de vérification.

Le chapitre 3 présente les concepts organisationnels qui sont au coeur de nos travaux. Ces concepts sont présentés dans leur version de base et suivent quelques extensions propres à des domaines d'applications comme la sémantique de la dynamique des rôles et l'étude d'architectures d'agents en utilisant le modèle organisationnel et les outils formels.

Le chapitre 4 propose une approche de capitalisation et réutilisation des connaissances. Les principes de modélisation d'un processus de conception de produits industriels par un modèle RIO enrichi sont présentés. Ensuite, la démarche de construction d'une mémoire de projet, d'une ontologie et d'un SMA en charge de la capitalisation est détaillée.

Le chapitre 5 pose les bases d'une approche d'ingénierie de Systèmes Multi-Agents Holoniques. Cette approche est basée sur le concept de capacité ajouté au modèle RIO et sur la définition d'un framework organisationnel pour l'analyse et la conception de SMAH.

Le chapitre 6 pose les bases de deux méthodologies. La première est issue d'un travail initial concernant l'analyse et la conception de SMA Holoniques utilisant un cadre particulier. La deuxième est issue d'une collaboration avec l'équipe de recherche de Massimo Cossentino[3] et est plus générale.

[3]chercheur de l'ICAR/CNR Palerme.

Le chapitre 7 conclut et présente quelques pistes de recherches futures.
Le chapitre 8 présente mon curriculum vitae.

Chapitre 2

Outils formels

2.1 Introduction

Ce chapitre définit le formalisme de spécification que nous utilisons par la suite pour spécifier les systèmes multi-agents. Nous avons mis l'accent sur :

- la capacité d'expression concernant les principaux aspects des SMA,
- la possibilité de prototyper et vérifier une spécification,
- la possibilité de pouvoir raffiner la spécification.

Il existe plusieurs travaux dans le domaine des SMA qui tentent de définir un langage de spécification. Nous nous concentrons sur ceux qui spécifie l'ensemble des aspects relatifs aux agents et non seulement une partie spécifique comme la communication, les normes d'une organisation, ...
Dans (Wooldridge, 1992), la logique temporelle est utilisée pour spécifier des agents intentionnels. La spécification distingue la représentation des états mentaux de l'agent exprimée dans le langage IAL (pour Internal Agent Logic) et le raisonnement sur les agents exprimé dans le langage AL (pour Agent Logic). Les deux langages sont fondés sur la logique temporelle. Ainsi, en plus des opérateurs temporels, les modalités Connaît, Envoie et Fait sont introduites pour raisonner sur les croyances, la communication et l'action. D'un point de vue théorique, la spécification en logique temporelle est riche en ce sens qu'elle possède une sémantique formelle permettant d'effectuer des preuves de propriétés. D'autres formalismes allant dans ce sens ont été proposés pour la spécification d'agents intentionnels. Citons l'extension de la logique temporelle arborescente par des opérateurs de modalités (Kinny et al., 1996) pour représenter les croyances, les désirs et les intentions. L'approche développée dans (Fisher and Wooldridge, 1993; Fisher, 1994) est basée sur l'exécution de spécifications en logique temporelles. Malgré l'importante contribution de ces approches à la définition d'un cadre formel pour

spécifier les propriétés de systèmes multi-agents, des problèmes subsistent quant à la dérivation d'un modèle exécutable à partir de la spécification. Par ailleurs, la complexité du formalisme et le manque de règles méthodologiques pouvant assister le spécifieur dans sa tâche rend difficile l'élaboration des spécifications. Le langage Z (Spivey, 1992) est un formalisme qui a fait l'objet d'études assez importantes et a servi à la spécification d'applications industrielles. C'est pourquoi le langage dispose d'outils d'aide à la spécification et de documentation permettant de faciliter son utilisation. La spécification des SMA à l'aide du langage Z a été proposée dans (Luck and D'Inverno, 1995). Cette dernière proposition à l'avantage d'esquisser quelques éléments sur la démarche à suivre pour construire une spécification. En effet, la spécification s'effectue par raffinement d'une hiérarchie de schémas Z composée d'entités abstraites prédéfinies. L'environnement constitue le sommet de la hiérarchie et encapsule les attributs pouvant être perçus par les agents. Le regroupement d'un sous-ensemble de ces attributs et des actions associées forme un objet. Les actions représentent les capacités de l'objet. La troisième entité est l'agent qui possède un état mental auquel sont associés des buts à atteindre ou des tâches à réaliser. Lorsque ces buts résultent des motivations (ou désirs) de l'agent, on le considère comme étant un agent autonome. En effet, l'agent est un objet avec des buts à réaliser. L'agent autonome est un agent qui a un ensemble de motivations pouvant générer des buts à atteindre. Les qualités et les limites de l'approche de spécification des systèmes multi-agents telle qu'elle est proposée par Luck et D'Inverno sont essentiellement liées aux possibilités offertes par le langage Z. En effet, les spécifications sont compréhensibles et se construisent par raffinement d'un squelette de spécification de très haut niveau d'abstraction. Cette approche par raffinement permet aussi d'obtenir une description détaillée du système qui est proche d'une implémentation. Dans (Luck et al., 1997), une proposition pour l'implantation des spécifications des SMA en C++ a été esquissée. En revanche, l'inconvénient de cette approche de spécification réside essentiellement dans l'incapacité du langage Z à prendre en compte l'aspect réactif et la dynamique des systèmes multi-agents. La spécification d'un tel aspect nécessite l'utilisation d'un autre formalisme. Par conséquent, l'expression en Z des propriétés liées à la dynamique du système n'est pas possible. En effet, l'absence de la spécification du comportement du système dans le temps ne permet pas de prouver des propriétés telles que la sûreté et la vivacité. L'approche proposée dans (Herlea et al., 1999a) consiste à déduire à partir de la spécification informelle des besoins initiaux une spécification semi-formelle puis formelle de ces mêmes besoins. C'est donc un processus de structuration et de formalisation à partir d'un énoncé qui décrit le système comme un tout. Le résultat de la structuration est une hiérarchie de composants qui définit

l'architecture des agents cf. (Brazier et al., 1997). La formalisation consiste d'une part à définir les éléments de base pour le problème et d'autre part à spécifier le comportement des composants issus de la structuration. Chaque composant est défini en terme d'entrées/sorties et de contraintes temporelles. Les entrées/sorties des composants sont définies avec les éléments de base du problème. Les contraintes temporelles sont exprimées de deux façons : en tant que besoin initial du système et en tant que scénario d'exécution possible. Cette démarche conduit naturellement à vérifier, par la vérification par évaluation sur un modèle, les formules spécifiant les besoins initiaux vis à vis des scénarios d'exécution qui constituent le modèle. La démarche présentée est une approche structurée et modulaire (par composants) de spécification de SMA. L'organisation par composants de la spécification permet le raffinement d'une spécification globale en spécifications plus détaillées de sous-composants. Toutefois, le formalisme utilisé ne permet pas de déduire aisément une implémentation. De plus, la seule technique de vérification possible est la vérification par évaluation sur un modèle avec les modèles fournis par la spécification.
Nous avons choisi de composer deux formalismes existants : Object-Z (Duke et al., 1991) et les statecharts (Harel, 1988). L'approche multi-formalismes que nous avons adoptée pour la spécification de SMA consiste en l'utilisation de plusieurs langages pour spécifier un système. Chaque partie de la spécification écrite avec la même notation, c'est-à-dire syntaxe, est une spécification partielle du système. La spécification complète est le résultat de la composition de toutes les spécifications partielles.
Nous définissons dans la suite de ce chapitre les bases de notre approche de composition tant du point de vue syntaxique que du point de vue sémantique. La section 2.2 précise les principes de l'intégration des formalismes Object-Z et statecharts et définit la syntaxe et la sémantique d'OZS. La section 2.3 décrit les différents outils de validation et de vérification que nous avons utilisés et propose des pistes qui permettent de réduire la complexité algorithmique de la vérification. La section 2.4 conclut ce chapitre.

2.2 La notation OZS

2.2.1 Principes

Pour pouvoir bâtir une spécification multi-formalismes d'un système, il faut établir une relation de composition entre les spécifications partielles, cf. (Zave and Jackson, 1993). En effet, tout ensemble de spécifications partielles

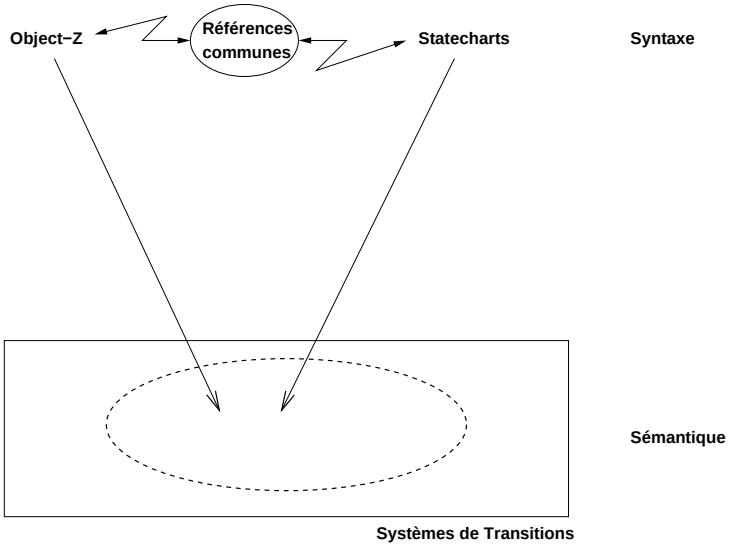


FIG. 2.1 – Approche de composition de formalismes

doit à un moment ou à un autre désigner ou faire appel à une partie du système spécifiée par un autre ensemble de spécifications partielles. Cette composition peut s'exprimer de plusieurs manières : du partage de variables aux contraintes exprimées sur une même entité ou encore avec la traduction vers un formalisme unique de tous les formalismes utilisés. Le procédé de composition que nous définissons est du type intégration. Le formalisme des statecharts est intégré au formalisme Object-Z pour spécifier des classes avec des aspects réactifs. Cette intégration, du point de vue syntaxique, est basée sur un *domaine syntaxique partagé* qui est constitué de deux parties :

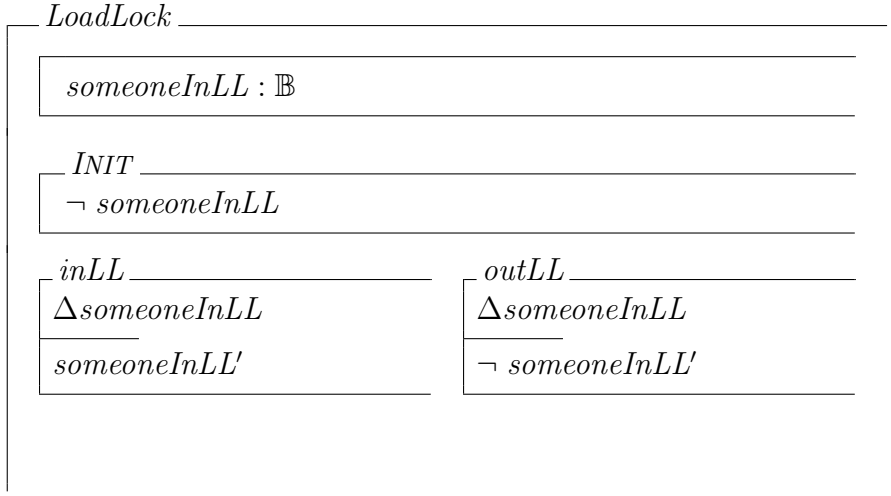
- un ensemble de types et classes Object-Z spécifiant les principaux aspects des statecharts,
- une fonction qui transforme un statechart en éléments du domaine syntaxique.

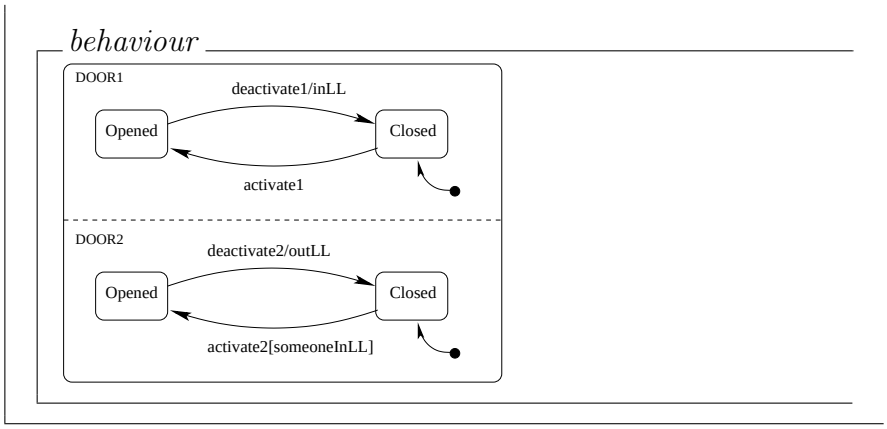
Ce domaine syntaxique partagé ne tient pas lieu de traduction des statecharts en Object-Z mais sert à référencer au sein d'une classe Object-Z les éléments du statechart inclus. Les classes de ce domaine ne sont utilisées que pour formuler des propriétés en Object-Z sur les éléments du statechart.

2.2.2 Syntaxe

La syntaxe du langage Object-Z est héritée du langage Z. Elle est basée sur le principe de schémas représentés par des boites. Dans le cas d'Object-Z, l'unité de construction est la classe qui est un schéma nommé qui contient

d'autres schémas. Ces schémas spécifient respectivement l'espace d'états de la classe, l'état initial de la classe et les opérations (un schéma par opération). Au sein de chaque schéma, la notation utilisée est la notation ensembliste de Z.
La syntaxe des statecharts est inspirée par celle des automates à états finis. Les éléments introduits pour enrichir sont la possibilité de définir une hiérarchie d'inclusion des états, du parallélisme ou produit cartésien d'états et un mécanisme de diffusion d'événements à l'ensemble du statechart.
L'exemple de la classe *LoadLock*, présentée ci-dessous, illustre l'approche multi-formalismes que nous proposons. Cette classe spécifie un sas composé de deux portes. Elle a comme attribut le booléen someoneInLL qui est vrai si et seulement si une personne se trouve entre les deux portes du sas. Le sous-schéma *Init* définit les valeurs initiales des attributs. Ici la contrainte stipule qu'initialement someoneInLL est faux. Les opérations *inLL* et *outLL* modifient l'attribut someoneInLL (notation Δ) et rendent respectivement someoneInLL vrai ou faux.
Le comportement spécifié par le statechart inclut dans le sous-schéma *behaviour* précise qu'initialement les deux portes sont dans l'état *Closed*. L'événement *activate*1 est déclenché suite à l'activation par un utilisateur de la première porte et a pour effet de faire passer la porte 1 de l'état *Closed* a l'état *Opened*. L'événement deactivate1 désactive la première porte et la fait passer de l'état *Opened* à l'état *Closed* et appelle l'opération *inLL*.
Les comportements de la porte 2 et la porte 1 sont définis comme s'exécutant simultanément. Cette simultanéité, ou composition **ET** pour les statecharts, s'exprime par une ligne en pointillés. La différence de comportement réside dans la condition rajoutée à l'ouverture de la porte 2 : il faut que le booléen someoneInLL soit vrai.





Le formalisme ainsi défini permet la spécification d'aspects réactifs, temporels et fonctionnels (Hilaire et al., 1999b; Gruer et al., 2000b; Gruer et al., 2000a; Hilaire, 2000). Nous pensons que cette composition de formalismes est bien adaptée pour spécifier les aspects qui caractérisent les SMA.

2.2.3 Sémantique

Afin de donner une sémantique formelle au langage OZS, nous avons choisi d'utiliser les systèmes de transition (Manna and Pnueli, 1991) qui permettent d'élaborer des modèles de comportement. Ils sont basés sur la définition d'un espace d'états et de transitions qui permettent le passage d'un état à un autre. Chaque état représente un état du système, c'est-à-dire une interprétation des variables du système. Avec un système de transition, on définit naturellement une sémantique opérationnelle. En effet, le système est décrit en terme de changements d'états. Dans une optique d'expression de propriétés d'un système spécifié par un système de transitions on utilise généralement la logique temporelle avec comme univers $\mathcal{V}$. La définition formelle d'un système de transition est la suivante :

Définition 1 *Un système de transition est défini en donnant un 4-uplet* $\langle \Pi, \Sigma, \mathcal{T}, \Theta \rangle$*, où :*

- $\Pi = \{u_1, u_2, \ldots, u_n\} \subseteq \mathcal{V}$ *est un ensemble fini de variables typées,*
- Σ *est un ensemble d'états. Chaque état* $s \in \Sigma$ *est une interprétation des variables de* Π*. L'interprétation par* s *de* $u \in \Pi$ *sera notée* $s[u]$*,*
- $\mathcal{T}$ *est un ensemble fini de transitions. Une transition* $\tau \in \mathcal{T}$ *représente une transformation d'état et peut être définie comme une fonction* $\tau : \Sigma \rightarrow 2^{\Sigma}$ *d'un état vers un sous-ensemble éventuellement vide d'états,*
- Θ *est une condition initiale qui caractérise les états initiaux possibles du système.*

Nous pouvons maintenant donner quelques éléments sur la définition de la sémantique du langage OZS. La spécification d'une classe Object-Z peut être vue comme la définition d'une machine abstraite. Les objets instances de la classe produisent des calculs selon cette machine abstraite. Ces calculs sont définis comme des changements d'états provoqués par les opérations de la classe. La transformation d'une classe Object-Z en un système de transitions donne une représentation compacte des calculs qu'une instance de la classe peut produire. Cette représentation est de la forme $S = \langle \mathcal{V}, \Sigma, \mathcal{T}, \Theta \rangle$ où S est un système de transition.
Soit cl une classe Object-Z et $S^{cl} = \langle \mathcal{V}^{cl}, \Sigma^{cl}, \mathcal{T}^{cl}, \Theta^{cl} \rangle$ le système de transitions correspondant. Cette classe est caractérisée par les ensembles suivants :

- $Attr(cl)$ est l'ensemble des attributs de la classe,
- $Param(cl)$ est l'ensemble des paramètres des opérations,
- $State(cl) \subseteq Attr(cl) \pfun Val$ est l'ensemble des états de la classe, un sous-ensemble des fonctions partielles finies des attributs vers leurs valeurs.

Finalement, $Op(cl)$ est l'ensemble des opérations de la classe. Les fonctions auxiliaires $\pi_i : Op(cl) \to \mathbb{P}\, Param(cl)$ et $\pi_o : Op(cl) \to \mathbb{P}\, Param(cl)$ donnent respectivement les ensembles de paramètres en entrée et en sortie d'une opération.
Nous complétons la définition des composants du système de transitions de la manière suivante :

- $Param(cl) = \mathcal{V}_{io}^{cl} = \bigcup_{\forall\, o \in Op(cl)} (\pi_i(o) \cup \pi_o(o))$,
- $Attr(cl) = \mathcal{V}_{p}^{cl}$,
- $\mathcal{V}^{cl} = \mathcal{V}_{io}^{cl} \cup \mathcal{V}_{p}^{cl}$,
- $\mathcal{A}^{cl} = \{Ax_1, \ldots, Ax_n\}$ où les Ax_i désignent les axiomes en logique du premier ordre relatifs aux attributs dans les schémas d'états,
- $\Theta^{cl} = Pr_1 \wedge \ldots \wedge Pr_m$ où les Pr_i désignent les prédicats du schéma INIT de la classe.

Chaque opération est transformée en un ensemble de transitions. Soit $\mathcal{T}^o = \{\tau_1, \ldots, \tau_n\}$ l'ensemble des transitions élémentaires résultat de la transformation de l'opération o. On a $\mathcal{T} = \bigcup_{\forall\, o \in cl} \mathcal{T}^o$.

La transformation d'un statechart en système de transitions est basée sur la configuration du statechart. La configuration est une fonction qui donne pour un instant t les états actifs du statechart. À chaque état du statechart est donc associé une variable booléenne du système de transitions. Cette variable est vraie si et seulement si l'état est actif.
A chaque transition du statechart est associée une transition du système de transitions. Une version plus complète de la sémantique est donnée dans (Gruer et al., 2004).

2.3 Outils

2.3.1 Validation

La validation est un ensemble de techniques qui permettent de s'assurer que le système en construction est conforme aux besoins initiaux. Cela consiste à se poser la question "Est-ce que nous concevons la bonne chose ?". On bâtit pour cela un prototype sur lequel on peut faire un certain nombre de tests. Ces tests consistent en l'observation du comportement du prototype à partir de scénarios prédéfinis. Le but de cette observation est de comparer le comportement attendu et le comportement effectif de la spécification.
Parmi les techniques existantes de validation le formalisme OZS permet d'utiliser le prototypage et la simulation. Le prototypage est basé sur l'exécution des spécifications en particulier les statecharts à l'aide d'environnements logiciels appropriés tels que STATEMATE (Harel et al., 1990). On parle alors d'animation car l'exécution des statecharts spécifiant le comportement du système donne lieu à une animation graphique qui montre l'évolution en terme de changements d'états et de déclenchement de transitions. L'avantage de cette technique est qu'elle est visuelle et qu'une spécification est facilement interprétable. L'inconvénient est qu'il faut raffiner la partie Object-Z pour la rendre exécutable et manipulable par l'environnement d'exécution des statecharts. Cette technique nous a permis de valider des SMA (Hilaire et al., 2001; Hilaire et al., 2000a; Campero et al., 2000)
La deuxième technique de validation possible avec OZS consiste en l'utilisation d'outils logiciels tel que SAL (de Moura et al., 2004) qui permet d'analyser des systèmes de transitions. Cette technique revient à simuler le système de transitions issu de la sémantique opérationnelle d'OZS. Le logiciel SAL explore les différents chemins ou traces d'exécutions générés par le système de transitions qui représente la sémantique de la spécification. A partir de l'état initial du système, SAL fait évoluer les variables représentant l'état du système de transition en déclenchant les transitions possibles au vu des conditions et de l'état courant. Cette deuxième technique permet également de valider la spécification respectivement aux besoins initiaux. L'avantage de cette technique est qu'elle est complète pour le langage OZS puisque la sémantique d'une classe OZS est entièrement exprimée par un système de transitions. L'inconvénient est que cette technique n'est pas visuelle. Nous avons expérimenté cette technique sur différents types de SMA dont des SMA Holoniques (Rodriguez et al., 2007b; Hilaire et al., 2008a).

2.3.2 Vérification

La vérification est un ensemble de techniques qui permettent de tester si le système en construction est conforme aux spécifications. Cela revient à se poser la question "Est-ce que nous avons réalisé ce que nous avions spécifié ?". Pour notre approche la vérification repose sur la sémantique formelle d'OZS. En effet, les systèmes de transitions sont pris en entrée par de nombreux logiciels de preuve automatique comme STeP (Manna et al., 1995) ou SAL (de Moura et al., 2004). Ces prouveurs automatiques utilisent deux types de technique. La première est le model checking ou vérification par évaluation sur un modèle. Cet outil génère l'ensemble des états possibles du système de transition en entrée et vérifie s'il n'existe pas de contre-exemple aux propriétés dont on demande la vérification. La vérification par évaluation sur un modèle ne peut se faire que pour des systèmes dont l'espace d'états est fini et de préférence de petite taille.
La deuxième technique est la preuve automatique ou semi-automatique de théorème, avec intervention de l'utilisateur. Cette technique permet de prouver des propriétés pour des systèmes dont l'espace d'états est infini mais est plus difficile à utiliser. En effet, par opposition au model checking pour lequel on fournit uniquement la propriété et la spécification du système il faut choisir une technique particulière de preuve et fournir des lemmes ou paramètres pour faciliter la preuve.
Certains outils comme SAL propose l'utilisation combinée du model checking et de la preuve automatique (Rushby, 2003). Le principe de cette démarche consiste à prouver par induction un théorème à l'aide d'un moteur borné de model checking et en utilisant la preuve automatique. Le fait de pouvoir borner le moteur de model checking permet d'éviter une exploration coûteuse dans un espace d'états qui peut alors être infini. La preuve se fait en deux étapes :

- Initialisation de la propriété : la propriété est démontrée par l'absence de contre-exemple pour l'état initial de la spécification à l'aide du model checker borné.
- Démonstration de la relation d'induction : on démontre par absence de contre-exemple que si le théorème est vrai à un pas n de la spécification, il est également vrai pour $n + 1$. La relation d'induction porte sur les transitions du système.

Cette technique nous a permis de démontrer des propriétés pour des SMA dont les espaces d'états sont infinis (Hilaire et al., 2008a).
Dans un but d'efficacité et de calculabilité, nous avons exploré la technique d'abstraction appliquée au framework de spécification CRIO présenté au chapitre 3. L'objectif général de cette technique (Loiseaux et al., 1995;

Clarke et al., 1992) est de trouver une relation d'abstraction qui permet de décomposer une spécification en plusieurs spécifications aux propriétés équivalentes et qui génèrent un espace d'état plus réduit et sont donc plus facilement gérables par des outils de preuve. Nous avons utilisé le concept de capacité comme moyen d'abstraire des parties de spécification dont les propriétés peuvent être prouvées séparément (Gaud et al., 2008b).

2.4 Conclusion

Le langage formel OZS que nous avons défini permet de spécifier les différents aspects des SMA (fonctionnels, réactifs, temporels, ...). Ce langage est doté d'une sémantique formelle qui permet de spécifier sans ambiguïté des SMA. Il hérite des possibilités de raffinement propres à Object-Z et aux statecharts. Cette notation n'est donc pas sans lien avec une implémentation. Il existe même des outils de génération de code à partir des formalismes Object-Z et statecharts. De plus, la sémantique d'OZS est opérationnelle ce qui permet de bénéficier d'outils de validation et de vérification. Nos travaux ont montré que ces outils constituent un apport important lors de l'ingénierie d'un SMA. En effet, avec différents types de SMA nous avons validé des spécifications écrites en OZS et prouvé des propriétés.
Le langage OZS est une des briques de base de notre approche car il permet de donner une sémantique formelle aux concepts organisationnels que nous utilisons pour l'ingénierie de SMA.
Nos travaux futurs concernent d'une part d'éventuelles extensions de la notation OZS comme par exemple l'utilisation de réseaux de Petri que nous avons étudié dans (Mazigh et al., 1999a; Mazigh et al., 1999b) et d'autre part la définition de nouvelles techniques de validation et de vérification comme la preuve compositionelle.

Chapitre 3

Modèle organisationnel

3.1 Introduction

Les approches pour l'ingénierie de SMA à base de concepts organisationnels sont nombreuses (Demazeau, 1995; Ferber and Gutknecht, 1998; Zambonelli et al., 2003; Hübner et al., 2007; Bresciani et al., 2004a; Molesini et al., 2005; Cossentino, 2005; DeLoach, 1999). Ces approches diffèrent par les concepts utilisés et leurs sémantiques respectives. En effet, chaque approche est influencée par le type de SMA à concevoir. Les outils proposés par chaque approche diffèrent également et dépendent de la sémantique et du formalisme utilisé pour modéliser les SMA de manière organisationnelle. Les auteurs de (Coutinho et al., 2005) distinguent deux grandes familles d'approches. La première considère une organisation comme un groupe de rôles et la seconde associe une organisation à un ensemble de contraintes exprimées sous la forme de règles ou de normes. Ces deux familles ne sont pas exclusives. On peut ainsi associer à un groupe de rôles un ensemble de contraintes comme dans GAIA (Zambonelli et al., 2003) ou OMNI (Dignum et al., 2004). Ces deux dernières approches ont adopté une notation formelle ou du moins intégrant des aspects formels. Toutefois, la contribution de cette formalisation se limite à la clarification de la sémantique des modèles présentés. Aucune approche n'est proposée quant à la validation et la vérification de ces modèles. GAIA et OMNI comportent plusieurs niveaux de modélisation à différents niveaux d'abstraction. Cette approche permet de raffiner les modèles depuis des besoins identifiés jusqu'à une conception initiale. Cependant, ce raffinement ne permet pas d'aboutir à une implémentation des modèles organisationnels.
Notre objectif est de définir une approche basée sur des concepts organisationnels pour l'ingénierie de SMA. Pour clairement définir les concepts organisationnels utilisés et bénéficier d'outils de validation et de vérification, nous

avons développé une approche combinant notation semi-formelle de type diagrammatique et description formelle en OZS (voir chapitre 2). Le principe est que tout concept organisationnel a à la fois une version semi-formelle sous forme de composant d'un diagramme et une version formelle qui définit sa sémantique. Chaque concept organisationnel aura ainsi une sémantique non ambigüe qui permet le raffinement jusqu'à aboutir à une implémentation grâce à une plateforme, comme JANUS (Gaud et al., 2008a), qui supporte les concepts du méta-modèle. La section 3.2 présente le méta-modèle RIO dans sa version originelle (Hilaire et al., 2001). La section 3.3 enrichit ce méta-modèle par la définition d'une sémantique pour une relation dynamique de mise en oeuvre des rôles par les agents. La section 4 présente l'étude selon une approche de rétro-ingénierie de deux architectures d'agents suivant le modèle RIO. La section 3.5 conclut ce chapitre.

3.2 Définitions du méta-modèle RIO

Le méta-modèle dans sa version originelle (Hilaire et al., 2001) était composé par les concepts de rôle, d'interaction et d'organisation. Le rôle est l'unité de description de comportement.

Définition 2 *Un rôle est une abstraction, d'un comportement ou d'un statut.*

Les interactions entres les rôles sont représentées par le concept d'interaction.

Définition 3 *Une interaction est une séquence d'actions dont les conséquences ont une influence sur le comportement futur des rôles.*

Une organisation est une unité logique définissant un ensemble de rôles en interactions ainsi que leur contexte. Ce contexte représente tout ce qui est commun aux rôles de l'organisation : lois, normes, règles, ressources, ...

Définition 4 *Une organisation est un contexte et un ensemble de rôles en interaction, tel que tout rôle de l'organisation interagit avec au moins un autre rôle.*

Les relations entre les concepts du méta-modèle sont schématisés par le diagramme de la figure 3.1. Cette figure montre qu'un organisation est composée de rôles et d'interactions. Chaque interaction a pour origine et destination un rôle. La classe suivante spécifie le concept de rôle. Un rôle est spécifié par un ensemble d'attributs, des stimulus auxquels le rôle peut réagir

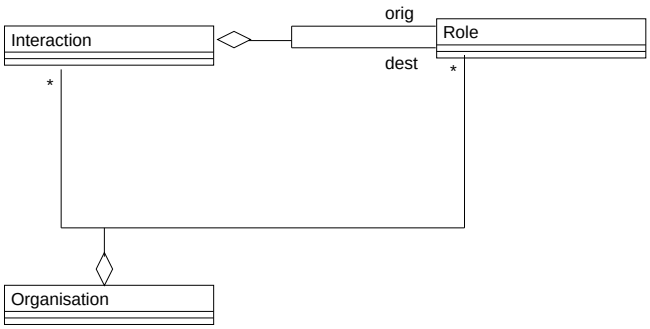


FIG. 3.1 – Méta-modèle RIO

et un ensemble d'actions que le rôle peut réaliser[1]. La première contrainte du schéma d'état spécifient que pour tout élément de stimulus il existe une transition qui possède ce stimulus comme événement déclencheur[2]. La deuxième contrainte spécifie que toute action d'une transition est décrite par une opération de la classe. Tout rôle possède également des ensembles de conditions (éventuellement vides) pour qu'une entité puisse respectivement le mettre en oeuvre et cesser de le mettre en oeuvre. Le sous-schéma *behaviour* spécifie à l'aide d'un statechart le comportement du rôle. La contrainte liée au stimulus précise que tout élément de l'ensemble stimulus doit appartenir à au moins une étiquette de transition du statechart du sous-schéma *behaviour*. La deuxième contrainte spécifie que toute action appartenant à une étiquette de transition du statechart du sous-schéma *behaviour* doit appartenir à l'ensemble actions.

Role

$$
\begin{array}{l}
attributes : \mathbb{P}\, Attribute \\
stimulus : \mathbb{P}\, Event \\
actions : \mathbb{P}\, Action \\
obtainConditions, leaveConditions : Condition \\
\hline
\forall\, s \in stimulus, \exists\, e \in behaviour \bullet \\
\qquad (\exists\, t \in e.transitions \bullet t.label.event = s) \\
\forall\, e \in behaviour \bullet \\
\qquad (\forall\, t \in e.transitions \bullet t.label.action \subseteq actions)
\end{array}
$$

[1]Le symbole $\mathbb{P}$ désigne l'ensemble des parties d'un ensembles.
[2]Le symbole $\bullet$ signifie tel que.

behaviour

La classe Interaction spécifie le concept d'interaction. Elle est composée de deux rôles, le rôle origine et le rôle destination ainsi que de deux actions appartenant respectivement aux rôles origine et destination. Une interaction est spécifiée par le fait qu'une action du rôle d'origine entraîne une réaction du rôle destination. Cette affirmation est spécifiée par l'invariant temporel de la classe (dernière ligne de la classe) qui stipule qu'à partir d'un certain état de la classe l'opération op_2 prendra en entrée les résultats de l'opération op_1 (opérateur $\|$).

Interaction

$orig, dest : Role$
$op_1, op_2 : Action$

$op_1 \in orig.action$
$op_2 \in dest.action$

$\Diamond(orig.op_1 \| dest.op_2)$

Organisation

$roles : \mathbb{P}\, Role$
$interactions : \mathbb{P}\, Interaction$

$\forall\, r \in roles,$
$\quad \exists\, i \in interactions \bullet r = i.orig$
$\qquad\qquad \vee\ r = i.dest$
$\forall\, i \in interactions \bullet$
$\quad (i.orig \in roles \wedge i.dest \in roles)$

Une organisation est composée de rôles et d'interactions tels que tout rôle doit être en interaction avec un autre rôle de l'organisation et qu'une interaction fait forcément intervenir deux rôles de l'organisation.

3.3 Dynamique des rôles

Un des inconvénients des approches organisationnelles est que la relation entre les agents et leurs rôles est statique. En d'autres termes, un agent a initialement des rôles associés et il ne peut en changer. Nous avons introduit dans le modèle RIO les concepts nécessaires à la définition d'une relation dynamique de mise en oeuvre des rôles (Hilaire et al., 2002c; Hilaire et al., 2007).
Cette relation dynamique est basée sur la définition du concept de RoleContainer voir figure 3.2. Un RoleContainer est une entité qui hérite du concept de rôle et qui est donc une abstraction de comportement. Un RoleContainer compose un certain nombre de rôles. La relation de composition permet de définir des rôles joués simultanément. Nous avons défini cette relation de

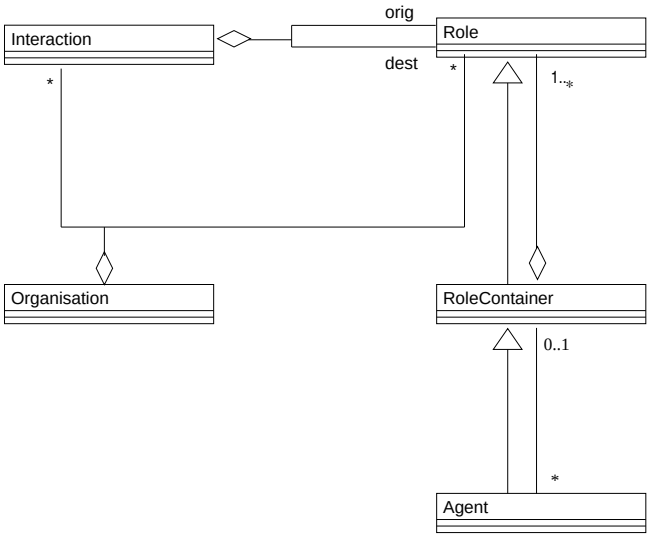


FIG. 3.2 – Méta-modèle RIO avec RoleContainer

composition de manière explicite et formelle. La sémantique de la dynamique des rôles est ainsi clairement définie.

La relation de composition est spécifiée au sein de la classe *RoleContainer* par la fonction *Retrieve*. Cette fonction permet de composer les différentes parties des spécifications des rôles de l'ensemble *playing* en un ensemble cohérent. Ces parties de spécifications correspondent aux attributs, opérations, stimulus et comportement. L'idée suggérée est inspirée de la spécification à base de points de vues (Ainsworth et al., 1994; Derrick et al., 1995; Koukam et al., 2003). Le principe de la spécification par points de vues consiste en la séparation de spécifications partielles dans le cas de systèmes complexes. Le problème de composer dynamiquement des rôles est similaire à la composition de points de vues. Cela revient à fusionner des spécifications partielles et vérifier leurs cohérences.

RoleContainer

Role

$$
\begin{array}{|l}
\hline
playing : \mathbb{P}\, Role \\
\hline
\exists\, Retrieve_{att} : Attribute \rightarrow Attribute \bullet \\
\qquad \forall\, r \in playing \bullet \\
\qquad \forall\, a \in r.attributes, \exists\, a' \in attributes \bullet a' = Retrieve_{att}(a) \land a \preccurlyeq a' \\
\exists\, Retrieve_{op} : Action \rightarrow Action \bullet \\
\qquad \forall\, r \in playing \bullet \\
\qquad \forall\, a \in r.actions, \exists\, a' \in actions \bullet a' = Retrieve_{op}(a) \land a \preccurlyeq a' \\
\exists\, Retrieve_{stim} : Stimulus \rightarrow Stimulus \bullet \\
\qquad \forall\, r \in playing \bullet \\
\qquad \forall\, a \in r.stimulus, \exists\, a' \in actions \bullet a' = Retrieve_{stim}(a) \land a \preccurlyeq a' \\
\exists\, Retrieve_{beh} : Statechart \rightarrow Statechart \bullet \\
\qquad \forall\, r \in playing \bullet \\
\qquad \forall\, Retrieve_{beh}(r.behaviour) \preccurlyeq behaviour \\
\hline
\end{array}
$$

La fonction *Retrieve* associe a chaque attribut, opération, stimulus ou comportement d'un rôle un attribut, opération, stimulus ou comportement qui le raffine dans le *RoleContainer*. La relation de raffinement est dénotée par le symbole $\preccurlyeq$. Un objet o' qui raffine un objet o est représenté par $o \preccurlyeq o'$. Nous avons alors les relations suivantes où rc désigne un RoleContainer et r_1 et r_2 deux rôles. Dans ce contexte, les attributs, actions, stimulus et comportement du RoleContainer sont définis comme l'union des raffinements des attributs, actions, stimulus et comportements des rôles.

$$
\begin{array}{l}
rc.attributes = Retrieve_{att}(r_1.attributes) \cup Retrieve_{att}(r_2.attributes) \\
\land\ rc.actions = Retrieve_{op}(r_1.actions) \cup Retrieve_{op}(r_2.actions) \\
\land\ rc.stimulus = Retrieve_{stim}(r_1.stimulus) \cup Retrieve_{stim}(r_2.stimulus) \\
\land\ rc.behaviour = Retrieve_{beh}(r_1.stimulus) \cup Retrieve_{beh}(r_2.stimulus)
\end{array}
$$

Les différents cas de raffinement sont présentés ci-dessous.
Pour les attributs, il y a trois cas différents. Quand les attributs sont disjoints alors la relation de raffinement est l'identité. Le second cas considère des attributs spécifiant le même aspect dans des rôles différents et avec des domaines de définition différents du même type. La relation de raffinement consiste alors à créer un attribut pour le *RoleContainer* définit sur l'union des deux domaines précédents. On peut citer par exemple pour les rôles Enseignant et Chercheur l'attribut diplôme. Le troisième cas est similaire au second mais avec des domaines de type différents. La relation de raffinement consiste alors en un produit cartésien des domaines de définition initiaux. On peut citer par exemple pour les rôles Enseignant et Chercheur l'institut

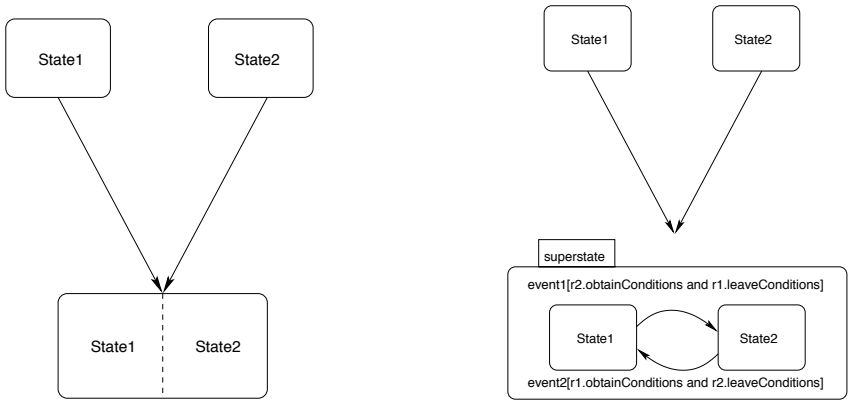
d'appartenance. L'opérateur $\upharpoonright$ est la projection d'un tuple sur son n-ième élément. Par exemple, si $a = (a_1, a_2)$ alors $a \upharpoonright 1 = a_1$. Dans les équations suivantes a désigne un attribut de RoleContainer et a_1 et a_2 désignent des attributs de rôles.

$$\begin{cases} a = Retrieve_{att}(a_1) = Retrieve_{att}(a_2) \\ (a_1 = \perp \Rightarrow a = Retrieve_{att}(a_2)) \vee (a_2 = \perp \Rightarrow a = Retrieve_{att}(a_1)) \\ a = (a_1, a_2) \wedge Retrieve_{att}(a_1) = a \upharpoonright 1 \wedge Retrieve_{att}(a_2) = a \upharpoonright 2 \end{cases}$$

Pour les opérations, le premier cas, trivial, est quand les opérations sont équivalentes. Le raffinement consiste alors en l'identité. Pour les autres cas, au moins un des attributs du *RoleContainer* résulte d'un raffinement et fait partie des paramètres de plusieurs opérations. Si le raffinement est une union de domaine alors l'opération appliquée pour le *RoleContainer* est celle concernée par le domaine restreint. Si le raffinement est un produit cartésien, alors chaque opération est appliquée en séquence à chaque composante du produit cartésien. La notation $\widehat{=}$ permet de définir un schéma (au sens Object-Z) et $\fatsemi$ permet de composer des schémas. Dans les équations suivantes *op* désigne une opération de RoleContainer et op_1 et op_2 des opérations de rôles.

$$\begin{cases} op \mathrel{\widehat{=}} Retrieve_{op}(op_1) \mathrel{\widehat{=}} Retrieve_{op}(op_2) \\ (pre\ op_1 \Rightarrow op \mathrel{\widehat{=}} Retrieve_{op}(op_1)) \vee (pre\ op_2 \Rightarrow op \mathrel{\widehat{=}} Retrieve_{op}(op_2)) \\ op \mathrel{\widehat{=}} op_1(a \upharpoonright 1) \fatsemi op_2(a \upharpoonright 2) \end{cases}$$

Le cas des stimulus est trivial puisque nous considérons des événements simples. $Retrieve_{stim}$ est une fonction de renommage.
Pour les comportements, spécifiés par des statecharts, il y a deux cas différents. Le premier cas est associé à des comportements indépendants. Dans le raffinement résultat les comportements des deux rôles doivent exister simultanément. Dans ce cas le comportement résultat est une composition AND (parallélisme au sens statecharts) des deux statecharts comme décrit dans la figure 3.3a.
Le second cas consiste en deux comportements exclusifs. En terme de raffinement cela se traduit par une composition XOR (ou exclusive au sens des statecharts) comme décrit dans la figure 3.3b. Les transitions qui permettent de passer du comportement d'un rôle à un autre sont conditionnées par les conditions d'obtention et d'abandon des rôles.
Dans les équations suivantes *stc* désigne le statechart spécifiant le comportement du RoleContainer et stc_1 et stc_2 les statecharts spécifiant les



a. raffinement and b. raffinement xor

FIG. 3.3 – Raffinement de statecharts

comportements des rôles.

$$\begin{cases} stc = (AND(Retrieve_{beh}(stc_1), Retrieve_{beh}(stc_2)) \\ stc = (XOR(Retrieve_{beh}(stc_1), Retrieve_{beh}(stc_2)), \\ (stc_1, stc_2, [r2.obtainConditions \wedge r1.leaveConditions], \\ (stc_2, stc_1, [r1.obtainConditions \wedge r2.leaveConditions])) \\ (stc_1 \subseteq stc_2 \Rightarrow stc = Retrieve_{beh}(stc_2)) \vee (stc_2 \subseteq stc_1 \Rightarrow stc = Retrieve_{beh}(stc_1)) \end{cases}$$

Les classes suivantes proposent un exemple de deux rôles, *Chercheur* et *Enseignant*, et un RoleContainer, *EnseignantChercheur*. Les rôles *Chercheur* et *Enseignant* sont spécifiés par des attributs disjoints comme *publications* pour *Chercheur* et des attributs communs comme *diplomes*, *institut* mais avec des domaines de définition différents.

Chercheur

Role

$publications : \mathbb{P}\ Publication$
$contrats : \mathbb{P}\ Contract$
$diplomes : \mathbb{P}\{DEA, PhD, Hdr\}$
$institut : \mathbb{P}\ Laboratoire$

Enseignant

$cours : \mathbb{P}\ Cours$
$responsable : \mathbb{P}\ Cours$
$diplomes : \mathbb{P}\{ing, DEA, PhD, Hdr\}$
$institut : \mathbb{P}\ Department$

$responsable \subseteq courses$

Au sein du RoleContainer, l'attribut *diplomes* est défini par l'union des domaines de définition des attributs *diplomes* des rôles *Chercheur* et *Enseignant*. L'attribut *institut* est défini par le produit cartésien des domaines de définition des attributs *institut* des rôles *Chercheur* et *Enseignant*.

EnseignantChercheur
RoleContainer

$$
\begin{array}{l}
diplomes : \mathbb{P}(\{DEA, PhD, Hdr\} \\
\qquad\qquad \cup \{ing, DEA, PhD, Hdr\}) \\
institut : Department \times Laboratory \\
\hline
playing = \{Enseignant, Chercheur\} \\
Retrieve_{att}(Enseignant.diplomes) \\
\qquad\qquad = Retrieve_{att}(Chercheur.diplomes) \\
\qquad\qquad = diplomes \\
Retrieve_{att}(Enseignant.institut) = 1 \upharpoonright institut \\
Retrieve_{att}(Chercheur.institut) = 2 \upharpoonright institut
\end{array}
$$

3.4 Rétro-ingénierie d'architectures

Par rétro-ingénierie d'architectures nous entendons l'étude des architectures existantes afin d'obtenir un modèle organisationnel correspondant. On peut qualifier cette approche de rétro-ingénierie organisationnelle. Le premier avantage obtenu avec ce modèle est de fournir une description de l'architecture et éventuellement de la comparer à d'autres. Nous avons mené une telle étude en considérant une architecture d'agents cognitifs en collaboration avec l'équipe SMA de Crakovie (Pologne) (Gruer et al., 2002b). Nous donnons en détail la description de deux architectures : l'architecture satisfaction/altruisme (Simonin and Ferber, 2000) principalement utilisée en robotique mobile collective et une architecture inspirée du système immunitaire (Watanabe et al., 1999).
Une architecture est une description des composants d'un agent. Ces composants constituent la partie contrôle de l'agent ainsi que la représentation des données, la perception et le traitement des informations. Depuis leurs débuts, de nombreuses architectures ont été proposées pour les SMA. Les architectures complexes à l'image de l'humain sont qualifiées de cognitives (Rao and Georgeff, 1995b; Lopez et al., 2003; Hübner et al., 2007), celles inspirées par des phénomènes naturels ou physiques de type stimulus/réponse sont dites réactives (Brooks and Connell, 1986; Ferber and Jacopin, 1991; Chapelle et al., 2002). En plus de ces différentes familles, ces architectures peuvent être caractérisées par des propriétés comme l'auto-organisation (Georgé et al., 2003; Mamei and Zambonelli, 2004; Gleizes et al., 2008) ou des mécanismes particuliers comme l'apprentissage (Sathyanath and Sahin, 2002; Maes, 1989). La plupart de ces architectures introduisent des concepts et des mécanismes

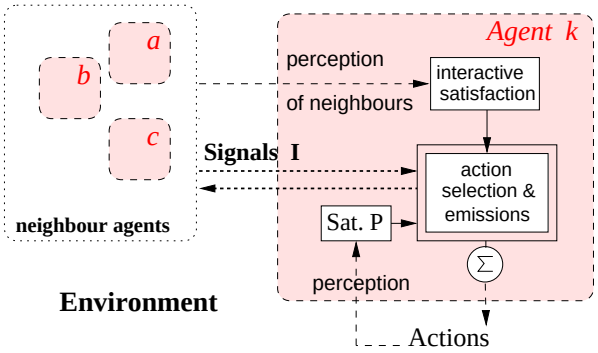


FIG. 3.4 – Architecture Satisfaction-Altruisme : interactions agent-agent et agent-environnement

différents, quelquefois dédiés à des problèmes spécifiques et exprimés dans des langages hétérogènes. Même si elles ont fait leurs preuves et que la communauté a reconnu leur utilité et leur efficacité pour certaines classes de problèmes la compréhension et la réutilisation de ces architectures reste une tâche difficile.

3.4.1 Satisfaction/Altruisme

Aperçu de l'architecture

L'architecture satisfaction/altruisme propose un mécanisme générique pour traiter les situations de conflits intervenant dans le cas de SMA situés (Simonin and Ferber, 2000). Cette approche repose sur des communications par diffusion de signaux d'attraction et de répulsion. Les signaux peuvent être considérés comme des champs de potentiel générés par les agents. L'architecture est définie par deux modules dédiés respectivement aux comportements individuels et collectifs. Le premier module calcule la satisfaction personnelle (boite Sat. P dans la figure 3.4) et le second mesure la satisfaction générée par les interactions de l'agent avec ses voisins (boite Interactive Satisfaction).
La satisfaction personelle est une mesure en temps réel de la progression de l'agent dans ses tâches. Cette satisfaction $P(t)$ est exprimée par une valeur appartenant à l'intervalle $[-P_{max}, P_{max}]$, où P_{max} est une valeur limite dépendant de l'application. La satisfaction personelle est calculée à chaque exécution de la boucle décision-action.
Le second module évalue les interactions de l'agent avec ses voisins. Cette satisfaction peut être négative (si les voisins gênent l'agent) ou positive (si les voisins apportent une aide potentielle) ou neutre.

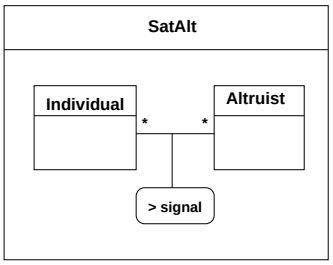


FIG. 3.5 – Diagramme RIO de l'architecture Satisfaction Altruisme

Les deux satisfactions sont utilisées pour prendre deux décisions : (i) l'agent doit-il continuer l'accomplissement de son but actuel (Chapelle et al., 2002) ? (ii) agent doit-il émettre des signaux pour influencer ses voisins ?
Les agents diffusent localement un signal d'attraction ou de répulsion, appelé $I(t)$, définit par une valeur numérique appartenant à l'intervalle $[-P_{max}, P_{max}]$ (noté I dans la figure 3.4). La sémantique des signaux I est la suivante : les valeurs positives dénotent une attraction et les valeurs négatives une répulsion. Une réaction coopérative consiste à réagir à la perception d'un tel signal. Cette réaction est définie par un déplacement selon la sémantique du signal (aller vers un signal d'attraction ou s'éloigner d'un signal de répulsion). Si plusieurs signaux sont reçus, l'agent choisit alors celui qui a la plus grande valeur absolue notée $I_{ext}(t)$. Cette réaction coopérative est appelée comportement altruiste et est sélectionnée si et seulement si

$$|I_{ext}(t)| \geq P(t) \ \wedge \ |I_{ext}(t)| \geq I(t)$$

Cette condition exprime le fait qu'un agent réagit à un signal externe si l'intensité est plus grande que sa satisfaction courante. La figure 3.5 représente l'organisation Satisfaction Altruisme. Elle est composée de deux rôles : Individual et Altruist. Chaque rôle peut interagir avec d'autres entités mettant en oeuvre les rôles Individual et Altruist.

Spécification

La classe DiscreteSensor spécifie un senseur discret. La première ligne introduit l'abréviation Intensity avec le symbole ==. Intensity est définit dans l'intervalle $[-P_{max}, P_{max}]$ où P_{max} est une constante réelle. Le schéma d'état spécifie une séquence de signaux discrets ou stimulus . Chaque senseur, indexe par un nombre entier $i \in \mathbb{N}$, donne un signal différent, $stimulus(i)$. L'opération *getMax* permet de sélectionner le plus grand signal, en valeur absolue.

DiscreteSensor

$Intensity == [-P_{max}, P_{max}]$

$stimulus : \text{seq}\, Signal$

getMax

$s! : Intensity$

$s \in ran\ stimulus$
$\nexists s' \in ran\ stimulus \bullet |s'| \geq |s|$

La classe *Individual* spécifie le rôle Individual. Elle introduit une action courante (current) et plusieurs fonctions qui permettent d'affecter respectivement les poids initiaux et courant d'une action ainsi que la mesure du gain apporté par une action. La classe spécifie également un senseur défini précédemment et les indices de satisfaction présentés en section 3.4.1. Les conditions d'obtention et d'abandon spécifient le choix entre les comportements individuel et altruiste.

Individual

role

$current : Action$
$initialWeight, weight : Action \rightarrow [0, 1]$
$progressionReward : Action \rightarrow BMValue$
$s : DiscreteSensor$
$satisfaction, I : [-P_{max}, P_{max}]$

$current \subseteq actions$
$obtainCondition = \{|I_{ext}()| < P() \vee |I_{ext}()| < I\}$
$leaveCondition = \{|I_{ext}()| \geq P() \wedge |I_{ext}()| \geq I\}$

I_{ext}

$ext! : \mathbb{R}$

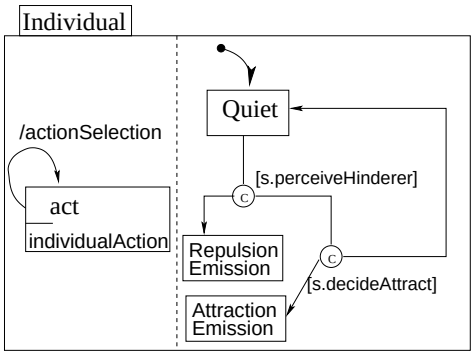
$ext! = s.getMax()$

actionSelection

$\Delta(current, weight)$

$$
\begin{aligned}
&(current' = current \land (\nexists a : Action \bullet\ a \in actions \\
&\qquad\qquad \land\ pre\ a \\
&\qquad\qquad \land\ weight(a) - \gamma \geq P())) \\
&\lor\ (current' = b \bullet\ b \in actions \\
&\qquad \land\ pre\ b \\
&\qquad \land\ (\nexists a : Action \bullet\ a \in actions \\
&\qquad\qquad \land\ pre\ a \\
&\qquad\qquad \land\ weight(a) \geq weight(b)) \\
&\qquad \land\ (P() \leq 0 \Rightarrow weight'(current) = 0))
\end{aligned}
$$

behaviour



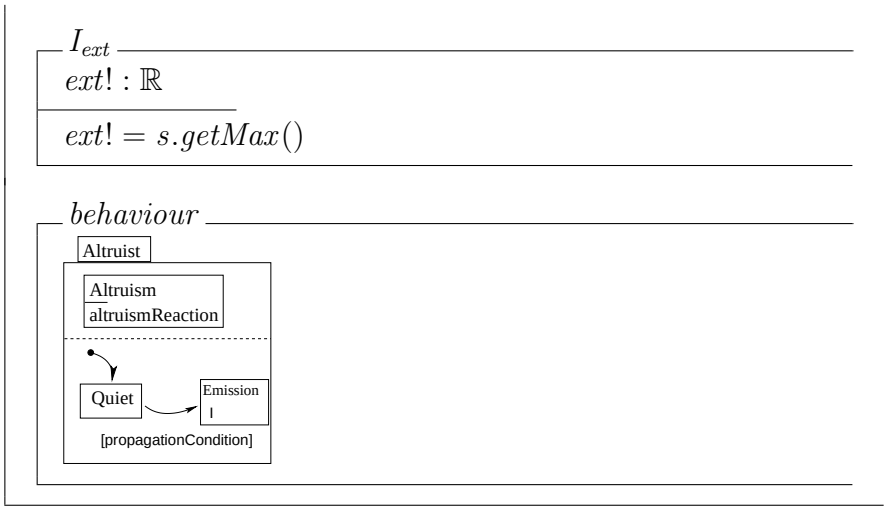
Le rôle Altruist est spécifié par la fonction *progressionReward*, un senseur discret *DiscreteSensor* et deux valeurs *satisfaction* et I qui mesurent respectivement la satisfaction et les influences extérieures.

Altruist

role

$$
\begin{aligned}
&progressionReward : Action \rightarrow BMValue \\
&s : DiscreteSensor \\
&satisfaction, I : [-P_{max}, P_{max}]
\end{aligned}
$$

$$
\begin{aligned}
&current \subseteq actions \\
&obtainCondition = \{|I_{ext}()| \geq P() \land |I_{ext}()| \geq I\} = \neg\ leaveCondition
\end{aligned}
$$



L'analyse menée à partir de cette spécification a permis de valider le comportement spécifié des rôles relativement à des expériences conduites avec des robots réels (Hilaire et al., 2005). D'autre part, l'utilisation de l'environnement SAL a permis de prouver que cette architecture permet à des robots navigant dans des environnements contraints d'éviter les situations de blocage (Hilaire et al., 2008a). Cette preuve a été faite grâce au démonstrateur automatique de théorèmes en prouvant au préalable les deux lemmes suivants.
Le premier lemme peut être interprété comme "le robot le plus contraint est le moins satisfait". Ce lemme est exprimé par la formule suivante[3].

$$\begin{aligned}\Diamond(\forall\, i,j \in [1..\sharp Robots], i \neq j \\ \wedge\ left(r_i) = wall \\ \wedge\ right(r_i) = robot \\ \Rightarrow\ sat(r_i) < sat(r_j))\end{aligned}$$

Le deuxième lemme peut être interprété comme "le robot qui devient altruiste est celui qui est le moins contraint"[4].

$$\Box(\forall\, i,j \in [1..\sharp Robots], i \neq j \wedge s(i) = altruist \Rightarrow sat(i) > sat(j))$$

Pour deux robots dans un corridor, si un est en état d'altruisme alors sa satisfaction est forcément plus faible que la satisfaction de l'autre. Ces deux

[3]Le symbole $\Diamond P$ indique que la propriété P sera vraie à partir d'un certain état du système. C'est le mode d'expression typique d'une propriété de vivacité.

[4]Le symbole $\Box P$ indique que la propriété P est toujours vraie. C'est le mode d'expression typique d'une propriété de sûreté.

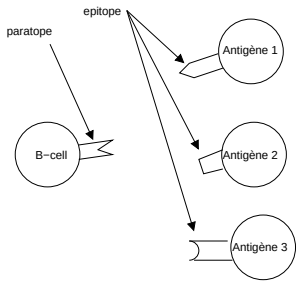


FIG. 3.6 – Reconnaissance anticorps/antigène

Paratope	Comportement	Idiotope
precondition de stimulation	attributs, code, données, comportement	affinités

TAB. 3.1 – Description d'un anticorps

lemmes, prouvés par SAL en utilisant l'induction, ont permis de prouver le théorème suivant "quand un robot est en état d'altruisme le conflit est résolu".

$$\Box(\forall\, i, j \in [1..\sharp \mathit{Robots}], i \neq j \wedge s(i) = \mathit{altruist} \Rightarrow \mathit{direction}(i) = \mathit{direction}(j))$$

Ce qui veut dire qu'un robot en état d'altruisme est repoussé et prend comme direction la direction de l'autre robot. Il en résulte le déblocage de la situation de conflit.

3.4.2 Architecture Immunitaire

Le système immunitaire naturel a été étudié pour ses propriétés d'adaptation au niveau local et d'émergence de comportement complexe au niveau global. Il existe plusieurs théories qui tentent d'expliquer les phénomènes immunologiques et plusieurs modèles qui simule le comportement d'un système immunitaire (Suzuki and Yamamoto, 2000b). Parmi les composants de base du système immunitaire les anticorps ont pour rôle de reconnaître les antigènes et se lier à eux afin de les neutraliser. La partie des antigènes reconnue par les anticorps est appelée epitope. Le paratope est la partie d'un anticorps qui correspond à un antigène spécifique. La figure 4.1 illustre ce processus de reconnaissance. Des études ont montré que chaque anticorps a également une partie idiotope. Cela veut dire qu'un anticorps est reconnu

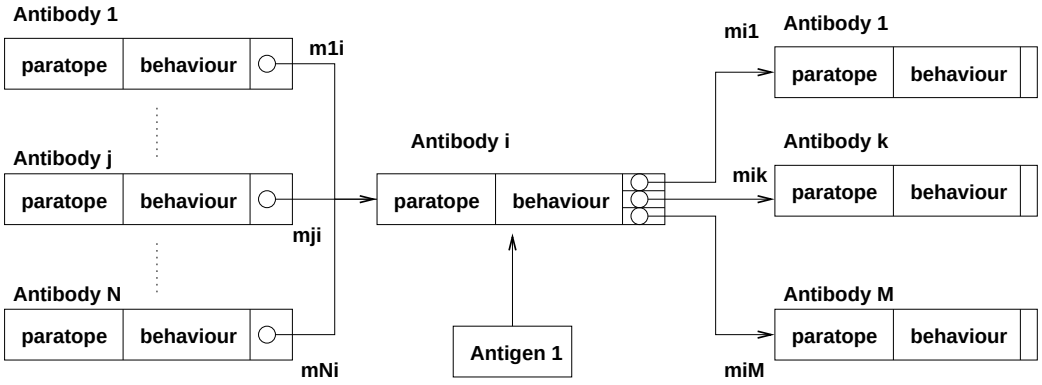


FIG. 3.7 – Interactions au sein d'un réseau idiotypique

comme un antigène par un autre anticorps (Farmer et al., 1986). Le prix Nobel N. K. Jerne a proposé le concept de réseau idiotypique (Jerne, 1974) qui fait l'hypothèse de l'existence d'interactions entre anticorps. Ces interactions prennent la forme de stimulations et d'inhibitions par le biais des paratopes et des epitopes. Ces liens entre anticorps forment un réseau dit idiotypique. Le modèle de Jerne a déjà été utilisé comme architecture d'agent (voir par exemple (Watanabe et al., 1999)). Cette architecture est une interprétation de la théorie de Jerne. Nous utilisons les concepts issus de (Watanabe et al., 1999) afin d'en extraire une version organisationnelle et formelle. Un anticorps est divisé en trois parties. La première partie, associée au paratope, stipule sous quelles conditions un anticorps est stimulé. La deuxième partie définit le comportement associé à l'activation de l'anticorps. La troisième partie, associée à l'epitope, définit les affinités (liens de stimulations et inhibitions) de cet anticorps envers les autres anticorps. Le mécanisme d'adaptation est basé sur l'apprentissage des valeurs d'affinités. Initialement, lorsque le système immunitaire est vierge, les valeurs d'affinités sont nulles. Au fur et à mesure que le système apprend, par un mécanisme de renforcement, les valeurs d'affinités sont mises à jour. Les interactions entre les différents anticorps sont matérialisées sur la figure 3.7. L'anticorps i stimule M anticorps et inhibe N anticorps. m_{ij} et m_{ik} représentent, respectivement, les affinités entre les anticorps j et i et les anticorps i et k. m_i est l'affinité entre un antigène et l'anticorps i. La population d'anticorps est représentée par le concept de concentration. Dans (Farmer et al., 1986), les auteurs proposent une méthode de calcul pour la concentration des anticorps. On note a_i la concentration de l'anticorps i. On a :

$$\frac{dA_i(t)}{dt} = (\alpha \frac{1}{N} \sum_{j=1}^{N} m_{ji} a_j(t) - \alpha \frac{1}{M} \sum_{k=1}^{M} m_{ik} a_k(t) + \beta m_i - k_i) a_i(t) \qquad (3.1)$$

$$a_i(t) = \frac{1}{1 + exp(0.5 - A_i(t))} \tag{3.2}$$

Dans la première équation, le premier et le deuxième terme de la partie droite dénote la stimulation et l'inhibition par d'autres anticorps. m_{ji} et m_{ik} sont des valeurs positives appartenant a l'intervalle $[0, 1]$. Le troisième terme, m_i , vaut 1 ssi i est stimulé et 0 sinon. Le quatrième terme, k_i , est un facteur de dissipation qui représente la mort naturelle des anticorps. La seconde équation est utilisée pour ramener le paramètre $A_i(t+1)$, issu de la première équation, entre 0 et 1. L'anticorps active est alors celui qui a la plus grande concentration. L'équation suivante décrit le mode de calcul de l'affinité m_{12} entre un anticorps Ab_1 et un anticorps Ab_2.

$$m_{12} = \frac{T_p^{Ab1} + T_r^{Ab2}}{T_{Ab1}^{Ab2}} \tag{3.3}$$

- T_p^{Ab1} est le nombre de pénalités reçues lorsque Ab_1 était sélectionné.
- T_r^{Ab2} est le nombre de pénalités reçues lorsque Ab_2 était sélectionné.
- T_{Ab1}^{Ab2} est le nombre de sélections communes de Ab_1 et Ab_2.

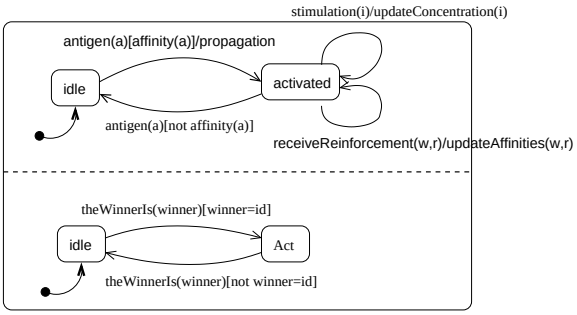
La classe Antibody spécifie un anticorps. Chaque anticorps est spécifié par ses interactions avec d'autres anticorps. Ces interactions sont soit des affinités soit des stimulations et sont spécifiées par des fonctions. Ces fonctions associent des anticorps à une valeur numérique. La fonction *timesBothSelected* associe à tout anticorps le nombre de sélections communes avec l'anticorps courant. La valeur concentration correspond à la concentration de l'anticorps et les valeurs α, β et k sont celles utilisées dans les formules précédemment citées. Par défaut, un anticorps est inactif. S'il reçoit un antigène qui le stimule alors il est activé. L'activation déclenche la propagation des affinités et des stimulations. Ces affinités font évoluer les concentrations des anticorps. A l'issu du processus de propagation, le réseau choisit parmi les anticorps activés celui qui a la plus grande concentration. L'anticorps sélectionné exécute son comportement et est ensuite évalué. Cette évaluation est traduite par des renforcements ou des pénalités qui permette de mettre à jour les valeurs d'affinités. L'opération *affinity* renvoie vraie ssi l'anticorps peut réagir avec l'antigène en entrée. Les opérations *updateConcentration* et *updateAffinities* recalculent les valeurs de concentration et les affinités. Ces deux opérations prennent en entrée un anticorps. L'opération *updateAffinities* prend en plus une valeur reinforcement ? en entrée et calcule l'affinité entre l'anticorps courant et l'anticorps sélectionné.

Antibody

Role

$id : Id$
$affinities, stimulation : Id \rightarrow \mathbb{R}$
$timesBothSelected : Id \rightarrow \mathbb{N}$
$concentration, \alpha, \beta, k : \mathbb{R}$
$numbersOfPenalty, numbersOfRewards : \mathbb{N}$

behaviour



INIT

$\forall\, i : Id \bullet timesBothSelected(i) = 0$

antigenAffinity

$antigen? : Antigen$
$stimulated! : \mathbb{B}$

updateConcentration

$i? : Id$
$\Delta concentration, timesBothSelected$

$$concentration' = \alpha \sum_{j \in \{x \mid 1 \leq x \leq \#Id \wedge x \neq id\}} stimulations(j) - \alpha \sum_{k \in \{x \mid 1 \leq x \leq \#Id \wedge x \neq id\}} affinities(k) + \beta - k$$

$$timesBothSelected'(i?) = timesBothSelected(i?) + 1$$

updateAffinities

$winner? : Id$
$reinforcement? : \{Penalty, Reward\}$

$reinforcement = Penalty \Rightarrow nbPenalty' = nbPenalty + 1$
$reinforcement = Reward \Rightarrow nbRewards + 1$
$affinities'(winner) = \frac{nbRewards + nbPenalty}{timesBothSelected}$

propagation

$affinities!$

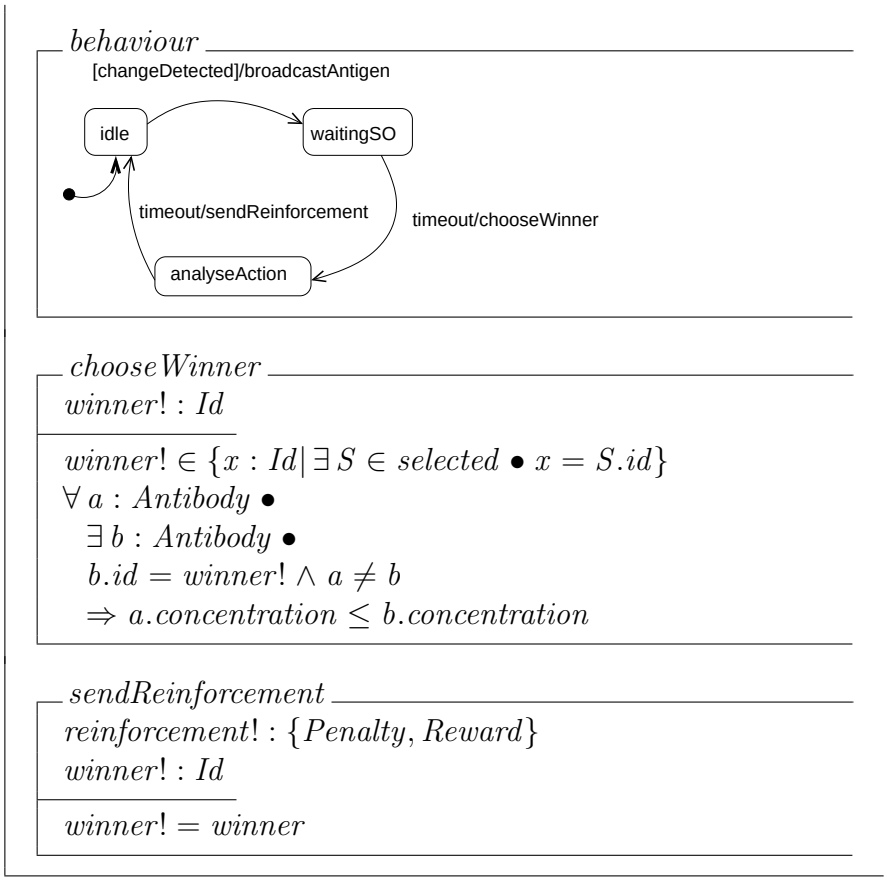
La classe IdiotypicNetwork spécifie un réseau idiotypique. Un réseau est spécifié par un ensemble d'anticorps, *antibodies*. Cet ensemble contient tous les anticorps du réseau. Parmi ces anticorps certains sont activés, ils appartiennent a l'ensemble *activated*, et parmi les activés il y a un sélectionné désigné par *winner*. Le comportement du réseau consiste en la diffusion d'antigènes et l'attente de la réaction des anticorps. Après un délai spécifié par *timeout* le réseau sélectionne un anticorps, analyse les résultats de l'exécution et envoie des pénalités ou des renforcements. L'opération *chooseWinner* choisit parmi les anticorps activés celui avec la plus grande concentration. L'opération *sendReinforcement* envoie aux anticorps activés des pénalités ou des renforcements selon les résultats de l'analyse du comportement de l'anticorps sélectionné.

IdiotypicNetwork

Role

$antibodies, selected : \mathbb{F}\, Antibody$
$winner : Id$

$selected \subseteq antibodies$
$winner \in \{x : Id \mid \exists\, S \in selected \bullet x = S.id\}$



Une implémentation de cette architecture a été faite (Rodriguez et al., 2005b) et prend la forme d'une API JAVA écrite pour la plateforme MadKit (Gutknecht and Ferber, 2000a).

3.5 Conclusion

Les concepts organisationnels présentés dans ce chapitre nous ont permis de modéliser des SMA allant de la gestion des connaissances (Monticolo et al., 2007a) à la simulation de systèmes complexes (Gruer et al., 2001a; Rodriguez et al., 2007c) et la résolution de problèmes (Koźlak et al., 2006; Rodriguez et al., 2007b). Notre démarche de modélisation de SMA s'appuie en effet sur des problèmes et des systèmes réels afin de valider les concepts choisis.
Les modèles obtenus permettent de modéliser et décomposer en unités plus simples les aspects pertinents des SMA tels que les interactions et les comportements complexes. Chaque organisation représente une projection selon

un certain point de vue des interactions entre agents. Les rôles sont donc des projections des comportements des agents dans ce contexte. Nous avons pris en compte la dynamique de la relation entre rôles et agents. Nos travaux concernant les architectures d'agents et de SMA sont motivés par les problématiques d'analyse, de comparaison et de réutilisation.
Du point de vue de l'analyse, les outils de validation et de vérification proposés permettent de mener une étude très complète. En effet, la validation permet de prototyper les architectures et assure que l'architecture correspond aux intentions de son concepteur. La vérification permet d'établir des propriétés concernant ces architectures. On peut ainsi s'assurer de propriétés de vivacité de sûreté ou encore que l'architecture exhibe bien ce qu'on attend d'elle comme l'auto-organisation par exemple.
La représentation à l'aide d'un langage unique de ces différentes architectures permet de les comparer plus facilement. En fait on a deux représentations l'une diagrammatique et semi-formelle qui est très intuitive et l'autre formelle et donc non ambigüe.
Enfin, pour ce qui est de la réutilisation, elle résulte d'une analyse organisationnelle qui permet de décomposer en schémas de conception organisationnels aux propriétés clairement identifiées et exprimés avec des concepts facilitant l'implémentation. Cette analyse peut mener, par un processus de rétro-ingénierie, comme cela a été le cas pour l'analyse de l'architecture Satisfaction-Altruisme (Hilaire et al., 2005) à l'identification de composants réutilisables au sein même de l'architecture proposée.

Chapitre 4

Capitalisation et réutilisation des connaissances

4.1 Introduction

L'intensification de la concurrence impose aux entreprises un renouvellement rapide de leurs produits à des coûts toujours plus compétitifs. Actuellement les industriels développent des produits de plus en plus performants avec des délais de plus en plus courts afin de susciter l'intérêt des clients et des ventes. Dans l'objectif d'améliorer leur rentabilité, les entreprises doivent s'imposer dans des marchés émergents en exposant leur capacité d'innovation. Cette capacité requiert la maîtrise de plusieurs dimensions ; l'optimisation des organisations, le contrôle des procédés industriels et le développement de l'entreprise apprenante (Nonaka and Takeuchi, 1995) où les organisations utilisent les connaissances acquises pour réaliser leurs activités. En effet, apprendre est devenu pour l'entreprise, le meilleur moyen de rester compétitive. Il ne s'agit donc plus aujourd'hui, de simplement progresser, mais de développer une culture apprenante, dans laquelle chaque collaborateur, chaque équipe et à terme, toute l'entreprise, pourront optimiser leurs potentiels.

Par ailleurs, les efforts pour réduire le temps de développement et améliorer la qualité du produit ainsi que les coûts d'industrialisation n'ont jamais été aussi nombreux. Ils conduisent à l'optimisation du processus de conception. Ce dernier est une ligne de conduite pour les équipes projets définissant les activités, les méthodologies et les objectifs à atteindre ; de la définition des besoins jusqu'à l'industrialisation du produit. Pour exécuter ce processus les équipes projets sont composées d'acteurs maîtrisant des domaines métier différents (mécanique, automatisme, injection, ergonomie, emboutissage, ...).

Ils ont par conséquent des rôles spécifiques qu'ils vont interpréter lors d'une ou plusieurs activités du processus. De plus, ces acteurs utilisent des outils dédiés à leur métier, générant un environnement où les sources d'informations sont hétérogènes et distribuées. Ces mêmes acteurs partagent leurs résultats et leurs savoir-faire. Ainsi l'accomplissement des activités métier se fait par la constitution d'organisations dans lesquelles les acteurs interagissent et partagent leurs connaissances pour atteindre un objectif commun qui est le développement du produit. La gestion de ces connaissances est un atout pour l'optimisation du processus de conception (Ermine, 2000). Elle a pour objectif de capitaliser les connaissances lors du processus et de permettre leur réutilisation. Cependant, la capitalisation et la réutilisation sont complexes puisqu'elles doivent prendre en compte la diversité et la nature des informations ainsi que les caractéristiques organisationnelles et sociales des acteurs métier.

Nous proposons une approche de gestion des connaissances fondée sur l'assistance des acteurs métier avec une capitalisation et une réutilisation semi-automatisée des connaissances. Cette approche est basée sur la modélisation du processus de conception comme un SMA. En effet ce processus fait intervenir des acteurs hétérogènes, distribués et collaborant à la réalisation d'un objectif commun. Pour modéliser ce processus, nous avons opté pour l'utilisation du méta-modèle RIO. Une telle approche permet de concevoir un système de gestion des connaissances qui prend en compte les rôles des acteurs métier et leurs collaborations tout au long du processus de conception. Ce chapitre est organisé comme suit : la section 2 présente l'approche de modélisation qui aboutit à la définition du composant OrgaDesign, modèle organisationnel du processus de conception de produit. La section 3 présente les composants MemoDesign, OntoDesign et le SMA KATRAS qui sont respectivement la structure de mémoire de projet, l'ontologie utilisée pour représenter les connaissances et le SMA chargé de la capitalisation et de l'assistance semi-automatique. La section 4 conclut ce chapitre.

4.2 Approche de modélisation

4.2.1 Principes

La capitalisation des connaissances vise à sauvegarder les connaissances acquises et détenues par les collaborateurs dans la pratique quotidienne de leur activité, principalement les savoir-faire et les retours d'expériences[1].
La capitalisation est utilisée dans le milieu industriel afin de constituer un

[1]Définition extraite de wikipédia

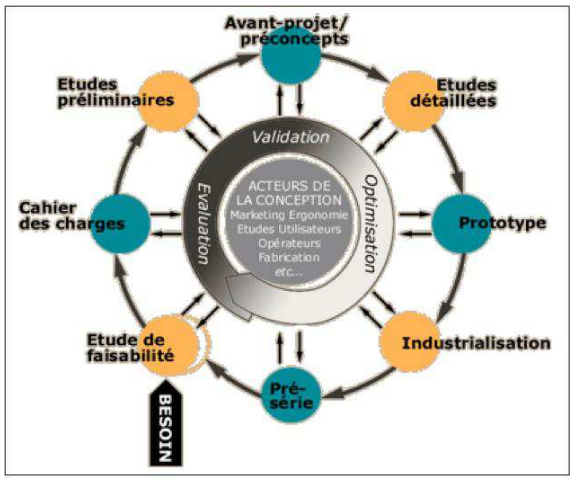


FIG. 4.1 – Processus de conception

patrimoine de connaissances en vue de diminuer les coûts et les risques liés à la conception de produits. On parle alors d'acteurs métiers à la place de collaborateurs pour les différents corps de métiers qui interviennent lors du processus de conception de produit.
Le processus de capitalisation est composé de plusieurs étapes (Grundstein, 2000). La première étape consiste à repérer ou identifier la connaissance. Pour faciliter cette identification un modèle qui met en évidence les interactions entre les acteurs métiers est nécessaire. Ce modèle doit prendre en compte la dimension organisationnelle du processus de conception et ses aspects hétérogènes et distribués. Ces raisons nous ont conduit à modéliser un processus de conception en utilisant le méta-modèle RIO (Monticolo et al., 2007c).
En conception les entreprises utilisent fréquemment un processus qui décrit les différentes phases à aborder pour à partir d'un besoin d'un client industrialiser un produit. Chaque phase est décomposée en activités qui sont réalisées par des acteurs métiers possédant un statut particulier. Par exemple, chaque projet doit avoir un chef de projet, statuts qui peut être occupé par des acteurs métiers différents lors de différents projets. Le séquencement des phases et activités ainsi que les différents statuts définissent un processus de conception. Le processus auquel nous nous sommes intéressés est présenté dans la figure 4.1. Il est extrait de (Gomes et al., 1999a). C'est un processus d'ingénierie concourante classique utilisé dans une entreprise où nous avons mis en place un système de capitalisation des connaissances (Monticolo et al., 2007a).

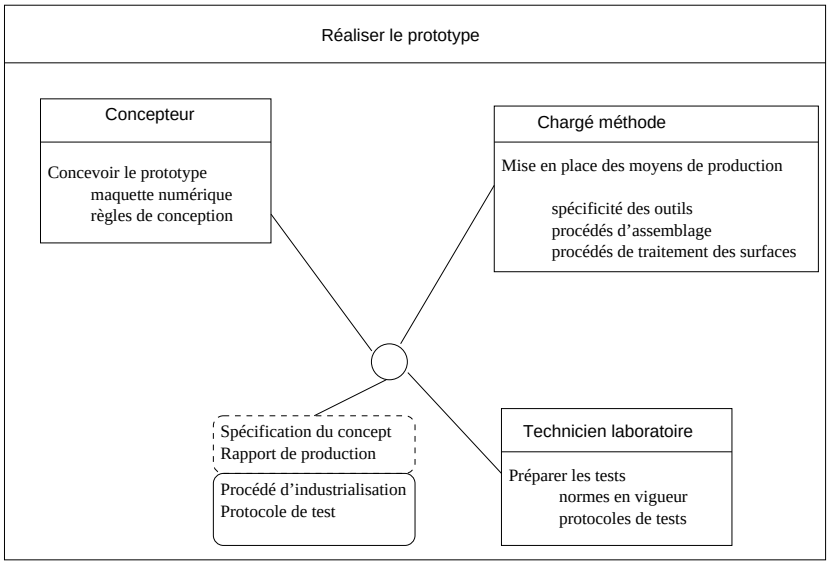


FIG. 4.2 – Exemple d'organisation liée au processus de conception

4.2.2 Connaissances et compétences

Le principe de l'approche est de modéliser le processus de conception à l'aide du modèle RIO. En effet, le fait de concevoir un produit est le résultat des interactions d'acteurs métiers occupant des statuts particuliers lors des différentes activités.
Chaque phase peut donc être vue comme une organisation composée de rôles en interaction. Ces rôles représentent les activités et sont donc des sous-organisations composées de rôles mis en oeuvre par des acteurs métiers. Cette modélisation permet de prendre en compte les différents rôles et leurs interactions lors du processus de conception. Un exemple d'organisation est présenté dans la figure 4.2.
Cette organisation modélise l'activité "Réaliser le prototype" qui fait partie de la phase "Prototypage". Cette activité est composée des rôles : Concepteur, Chargé méthode et Technicien laboratoire. Pour chacun de ces rôles, on identifie les connaissances pertinentes à capitaliser. Une étude menée sur le terrain, au sein d'équipes projets a permis de mener à bien l'identification de ces connaissances.
Le principe adopté est d'associer à chaque rôle un ensemble de compétences qui sont les savoir-faire de chaque rôle ou en d'autre termes leurs contributions aux objectifs de l'organisation. A chacune de ces compétences est associé un ensemble de connaissances requises pour mettre en oeuvre la compétence.

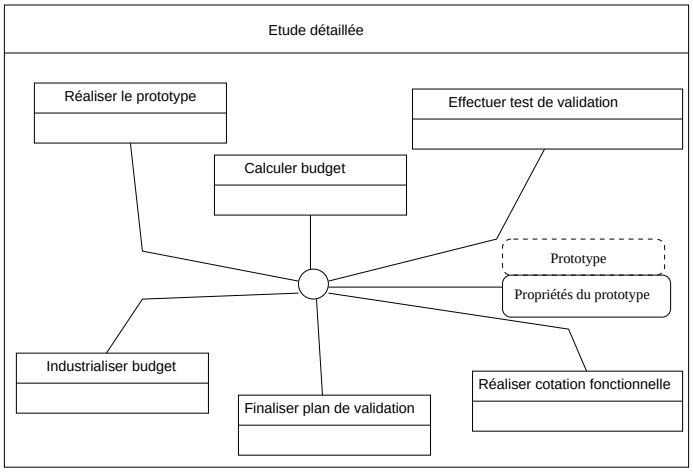


FIG. 4.3 – Exemple d'organisation liée au processus de conception

Par exemple, pour le rôle Chargé méthode de l'organisation Réaliser le prototype, figure 4.2, la compétence identifiée est "Mettre en place des moyens de production". Pour mettre en oeuvre cette compétence, l'acteur humain qui joue ce rôle doit posséder des connaissances à propos : de la spécificité des outils, des procédés d'assemblage et des procédés de traitement des surfaces. Les trois rôles de l'organisation Réaliser le prototype sont en interaction pour réaliser l'objectif. Cette interaction est matérialisée (cf figure 4.2) par un cercle lié aux rôles et au descriptif de l'interaction. Ce descriptif consiste en l'énumération des résultats de l'interaction avec, dans sa partie supérieure, les livrables, c'est-à-dire les objectifs de l'organisation et dans sa partie inférieure, la liste des connaissances associées au résultat de l'interaction entre les rôles. Dans l'exemple, les livrables sont la spécification du concept et le rapport de production. Les connaissances liées à l'interaction sont le procédé d'industrialisation et le protocole de test.

L'activité Réaliser le prototype fait partie de la phase Etude détaillée qui est également modélisée par une organisation cf figure 4.3. Les rôles de cette organisation sont des activités. L'objectif de cette organisation est de produire un prototype et les connaissances associées au prototype constituent ses propriétés. L'ensemble des phases du processus de conception, cf. figure **??**, ainsi modélisé constitue le composant OrgaDesign. Dans ce mémoire nous ne présentons que quelques organisations.

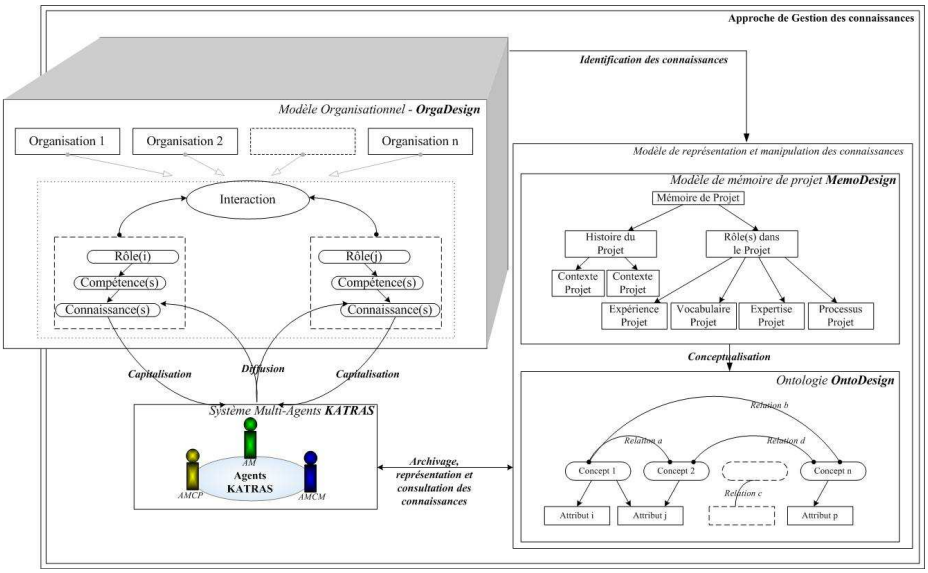


FIG. 4.4 – Aperçu du SMA KATRAS

4.3 Architecture du SMA pour la capitalisation

4.3.1 Aperçu du SMA

Le modèle OrgaDesign permet d'identifier la connaissance à capitaliser lors des projets utilisant le cycle de vie décrit dans la figure 4.1. Afin de pouvoir capitaliser ces connaissances nous avons construit une ontologie propre à la conception de produit (Monticolo et al., 2007b). Cette ontologie a été produite à partir du modèle organisationnel qui permet d'obtenir une cartographie des connaissances susceptibles d'être capitalisées.
Ces connaissances sont stockées au sein d'une mémoire organisationnelle ou mémoire de projet dont la structure est également déduite du modèle organisationnel (Monticolo et al., 2006b). Cette mémoire organisationnelle est constituée à partir des actions des acteurs métiers par des agents intégrés à un collecticiel de manière transparente ou « au fil de l'eau »(Monticolo et al., 2006a; Monticolo et al., 2008; Gomes et al., 2007). La figure 4.5 illustre notre approche globale de gestion des connaissances. Elle présente le premier élément fondateur de notre travail de recherche à savoir le modèle organisationnel du processus de conception (présenté dans ce chapitre). C'est un composant principal de notre solution de gestion des connaissances. Ces connaissances sont ensuite organisées sous la forme d'une mémoire de projet

appelée MemoDesign (second élément fondateur de notre approche). Elles sont ensuite conceptualisées et spécifiées grâce à une ontologie de domaine appelée OntoDesign (troisième élément fondateur). Le quatrième composant de notre approche est le système multi-agents, appelé KATRAS. Il permet de capitaliser, archiver et proposer une assistance à la réutilisation des connaissances.

4.3.2 Mémoire de projets

Les équipes projets sont des organisations dans lesquelles évoluent les acteurs métier. Une des approches de gestion des connaissances consiste à construire des mémoires contenant les connaissances utilisées dans ces organisations. Ces mémoires dites partagées ou organisationnelles sont définies dans (Dieng et al., 1999) par : « Une représentation et indexation explicites et persistantes des connaissances et informations (ou de leurs sources) dans une organisation, dans l'objectif de faciliter leurs accès, partages et réutilisation par les membres de l'organisation, pour leurs tâches collectives ou individuelles. »
Restreinte au monde du projet, la mémoire organisationnelle devient une mémoire de projet (Conklin and Begeman, 1988). Les travaux sur les mémoires de projet en conception mécanique proposent des approches de capitalisation de la logique de conception (Matta et al., 2000b) et de l'historique du projet (les expériences passées).
Un modèle de mémoire de projet fournit une indexation des connaissances pour faciliter leur partage et leur réutilisation. Dans ce sens, nous envisageons une capitalisation et une réutilisation des connaissances en fonction des rôles acteurs métier, ce qui nous conduit à définir un modèle de mémoire positionnant les connaissances en fonction des rôles des acteurs dans l'organisation. Par capitalisation au fil de l'eau nous entendons au fur et à mesure de l' avancement d'un projet. Ces méthodes fournissent une structuration des connaissances selon deux axes : la logique de conception et le contexte de la prise de décision ou logique de décision. La structure de MemoDesign est définie selon ces deux parties essentielles

- Un premier axe : l'histoire du projet Nous souhaitons développer une mémoire de projet pour archiver les connaissances pertinentes utilisées dans un projet qui seront réutilisables lors des projets futurs. Il est donc nécessaire de positionner ces connaissances dans un référentiel qui est le processus de conception. Chaque connaissance doit être placée dans un contexte précis. Ce contexte ne peut être expliqué qu'à l'aide d'une description du déroulement du projet, c'est-à-dire des activités réalisées dans le projet. Nous rejoignons donc l'idée d'un axe de clas-

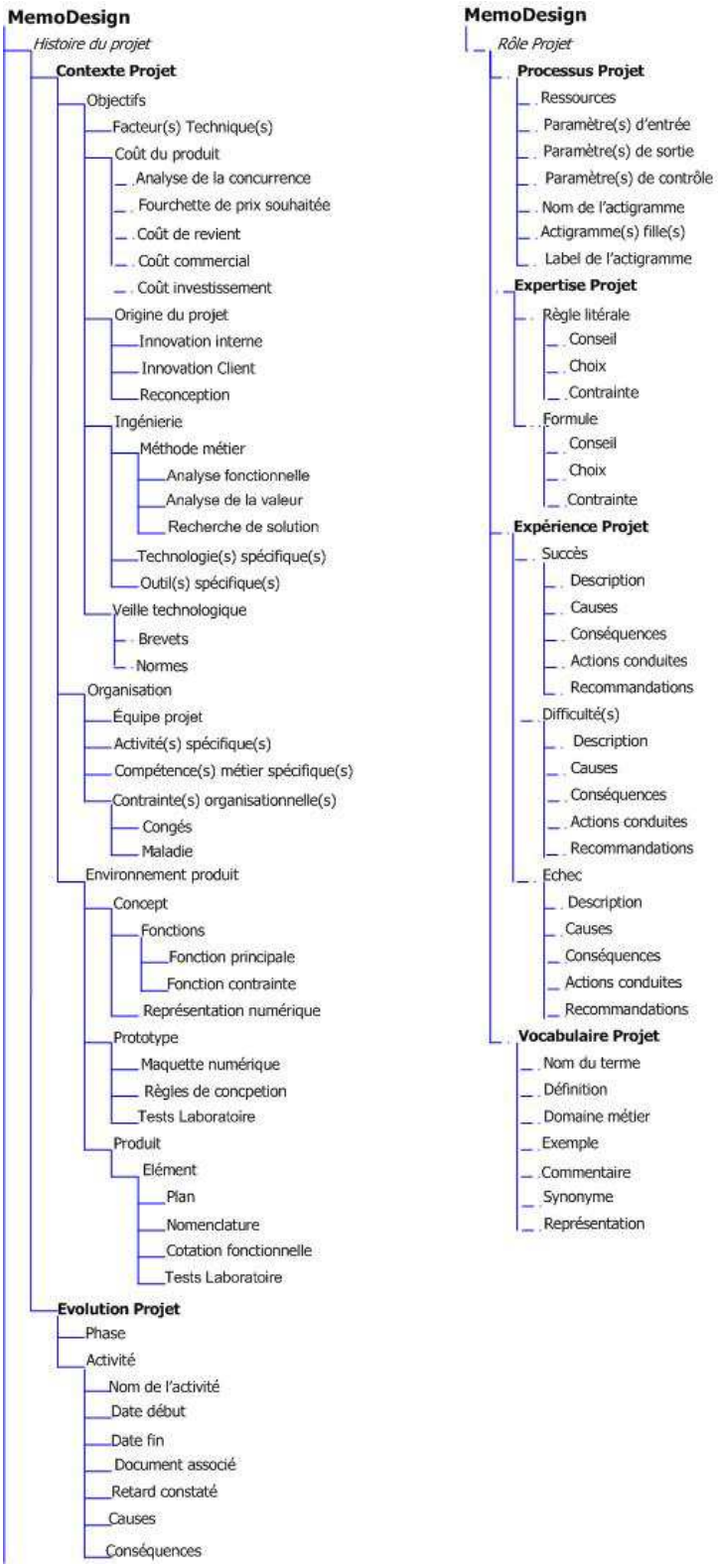


FIG. 4.5 – Taxonomie des connaissances métier

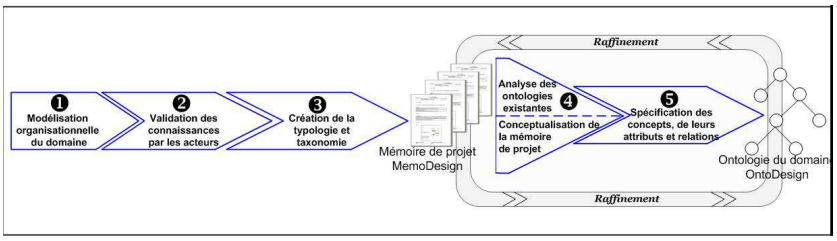


FIG. 4.6 – Processus de construction de l'ontologie OntoDesign

sification dédié à la mémoire de caractéristiques de projets tel que le décrit Matta (Matta et al., 2000b). Les connaissances classées dans cet axe présentent l'historique du projet, son origine et son organisation. Ces connaissances sont vérifiées puisqu'elles sont basées sur des faits produits durant le projet.

– Un second axe : les rôles dans le projet Nous avons observé, lors de l'étape d'identification des connaissances, qu'un grand nombre de connaissances résultent des travaux et des collaborations entre les acteurs métier. D'après la modélisation organisationnelle du processus de conception, nous avons observé que les acteurs réalisent les activités métier en interprétant des rôles spécifiques. Chacun de ces rôles nécessite une ou plusieurs compétences mettant en valeur une ou plusieurs connaissances. Il est donc nécessaire de classer les connaissances selon les rôles des acteurs métier qui les utilisent. Nous définissons ainsi le second axe de la mémoire de projet où sont classées les connaissances utilisées par les acteurs métier en fonction des rôles qu'ils interprètent.

Une taxonomie des connaissances suivant ces deux axes est présentée dans la figure 4.5.

4.3.3 Ontologie

Nous proposons à présent de construire une ontologie (Sowa, 2000) de domaine afin de définir une terminologie et une sémantique pour représenter ces connaissances. L'ontologie représente un vocabulaire consensuel qui définit le sens des concepts et des relations entre ces concepts. Ce vocabulaire peut être associé à un modèle qui décrit le contenu d'une mémoire de projet, ses propriétés, la manière dont elle peut être utilisée ainsi que la syntaxe et les contraintes fournies par le langage de représentation. L'objectif est d'assurer la spécification explicite des connaissances du domaine à l'aide d'un langage offrant une sémantique adaptée. De nombreux travaux définissent des

approches de conception d'ontologie. Nous décrivons, dans la suite de ce document, la démarche que nous avons suivie et nous expliquons quels sont les travaux qui nous ont guidés lors de ce travail.

Cette ontologie du domaine des projets de conception, que nous appelons OntoDesign, a été conçue en cinq étapes (cf. figure 4.6) :

- Une modélisation organisationnelle du domaine en vue de l'identification des connaissances ;
- La validation des connaissances du domaine par les acteurs métier ;
- La création de la typologie et taxonomie des connaissances ;
- Une analyse des ontologies de domaines existantes en vue de leur réutilisation ;
- La spécification des concepts, de leurs attributs et relations.

Ce processus de construction d'ontologie a été appliqué au sein de l'entreprise Zurfluh-Feller (Monticolo et al., 2007b) et possède actuellement 104 concepts et 34 relations.

4.3.4 Capitalisation au fil de l'eau

Grundstein (Grundstein, 2000) présente un processus de gestion des connaissances basé sur quatre étapes : repérer, actualiser, valoriser et préserver. Repérer les connaissances consiste à identifier et cartographier les connaissances lors des projets. À la suite de cette étape, il est nécessaire d'actualiser les connaissances c'est-à-dire de les enrichir, de les mettre à jour, mais également de les évaluer. L'étape suivante, valoriser, comprend la diffusion, le partage des connaissances, mais également leurs exploitations et leurs manipulations pour la réutilisation. La dernière étape préserver permet de modéliser et de formaliser les connaissances dans l'objectif de les archiver pour les réutiliser. Nous nous basons, dans la suite de ce paragraphe, sur le cycle de Grundstein pour déterminer les rôles nécessaires à la gestion des connaissances.

Les connaissances Projet sont les connaissances créées, utilisées et partagées lors d'un projet. Pour assurer le processus de gestion des connaissances Projet, nous avons identifié cinq rôles (figure 4.7) : détecteur de connaissances, médiateur, Créateur de la mémoire de projet, assistant cognitif et utilisateur de connaissances. Ces rôles seront mis en oeuvre par des agents qui assurent à leur tour l'identification, la capitalisation et la réutilisation des connaissances issues du projet. Le Détecteur de connaissances identifie les informations dites candidates c'est-à-dire les connaissances qui semblent être pertinentes et qui pourraient être utilisées pour construire la mémoire de projet. Pour identifier les connaissances, ce rôle les localise parmi les informations créées lors du projet (documents, planning, calculs, ...). Après avoir localisé et identifié ces connaissances, il les cartographie en vue de leur

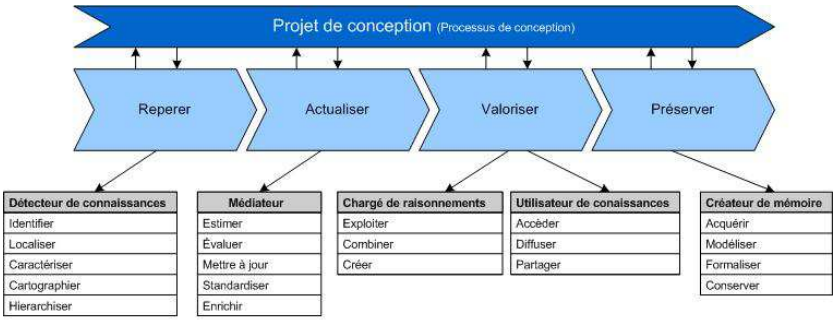


FIG. 4.7 – Rôles des agents KATRAS pour la capitalisation des connaissances projet

donner un contexte qui sera une activité métier c'est-à-dire une étape du processus de conception. La cartographie permet également la caractérisation des connaissances en vue de les annoter.

Le Médiateur présente les connaissances candidates aux membres de l'équipe projet identifiées comme référents Projet. Celles-ci sont ainsi soumises à un processus de validation. Les référents Projet peuvent modifier, supprimer ou accepter les connaissances candidates pour que celles-ci passent à l'état de Connaissances Projet. Dans cet objectif, le rôle du médiateur permet d'estimer, d'évaluer, de standardiser, d'enrichir et de mettre à jour les connaissances capitalisées.

Le Créateur de la mémoire de projet construit la mémoire de projet à partir des connaissances annotées par le détecteur de connaissances et validées par le médiateur. Les connaissances sont modélisées et formalisées selon la structure de la mémoire de projet. Ce rôle gère l'archivage c'est-à-dire la conservation des connaissances dans la mémoire de projet.

Le Chargé de raisonnements exploite les connaissances, il les manipule à l'aide de l'ontologie OntoDesign afin que ces connaissances puissent être réutilisées. Par exemple grâce à des relations telles que la déduction ce rôle pourra associer une règle de conception à une expérience réussie. Quelquefois, la combinaison et le raisonnement sur les connaissances permettent de créer de nouvelles connaissances utiles à l'assistance des acteurs lors des activités.

L'Utilisateur de connaissances assiste les acteurs métier à accéder à la mémoire de projet. Son objectif est de diffuser et partager les connaissances capitalisées lors du projet. Ce rôle transmet les requêtes formulées par les

utilisateurs au rôle Chargé de raisonnement. Il met également en forme les résultats de ces requêtes afin de les présentés aux utilisateurs. Ce rôle interagit avec les acteurs métier à travers les interfaces du module de gestion des connaissances.

Les cinq rôles dédiés à la gestion des connaissances projet, présentés sur la figure 4.7, sont mis en oeuvre pour chaque projet.

Les Connaissances métier sont les connaissances qui ont été créées lors de projets actuellement terminés. Ces connaissances sont issues de l'ensemble des mémoires de projet archivées. Elles constituent une base de connaissances métier que nous nommons Référentiel Métier. Ainsi, le Référentiel Métier contient le patrimoine des connaissances de l'entreprise construit à partir des expériences des projets réalisés.

Les connaissances métier sont managées par un cycle en trois étapes : actualiser, valoriser, préserver. La première étape du cycle de Grundstein Repérer n'a plus lieu d'être puisque les Connaissances Métier sont des connaissances déjà identifiées et capitalisées lors de la gestion des Connaissances Projet.

La gestion des Connaissances Métier est réalisée par quatre rôles (cf. figure 4.8) ; Médiateur Métier, Utilisateur de Connaissances Métier, Chargé de Raisonnements Métier et Créateur du Référentiel Métier.

Dans le même objectif que pour la gestion des Connaissances Projet, le rôle de créateur du référentiel métier intervient à ce niveau pour construire le référentiel métier avec les connaissances capitalisées dans l'ensemble des projets. Les rôles de médiateur, utilisateur de connaissances et chargé de raisonnement ont des fonctions similaires aux rôles de même nom lors de la gestion des connaissances Projet. La différence réside dans le fait que ces rôles gèrent des connaissances Métier provenant de l'ensemble des projets.

La figure 4.8 décrit à la fois les rôles des connaissances Projet pour les projet A et B ainsi que les rôles pour la gestion des connaissances métier et la création du référentiel métier à partir des mémoires des projets A et B. Cette gestion des connaissances Métier prend en compte les connaissances du projet A lorsque celui-ci est terminé.

4.4 Conclusion

Tout au long de ce chapitre nous nous sommes efforcés de mettre en place une approche de gestion des connaissances au fil de l'eau des projets de conception fondée sur :

– Un modèle organisationnel du processus de conception ;

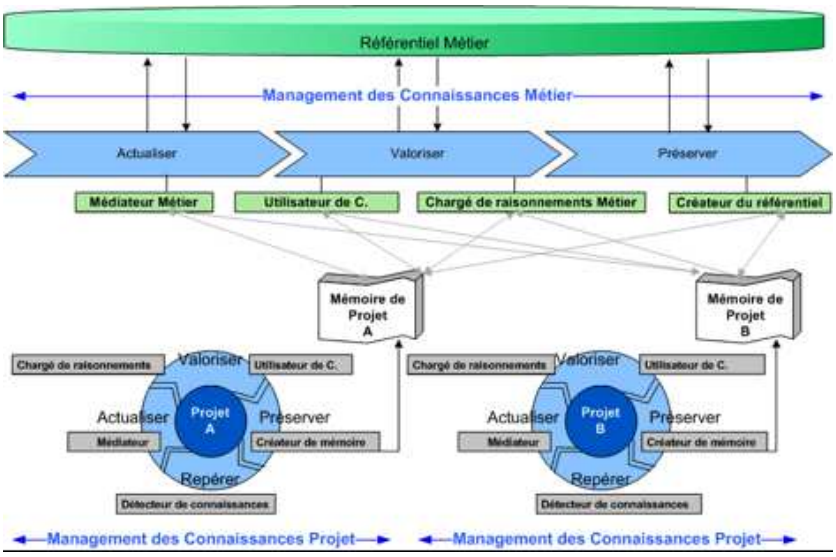


FIG. 4.8 – Rôles des agents KATRAS pour la capitalisation de la connaissance métier

- Une structuration et une représentation conceptuelle des connaissances utilisées par les rôles des acteurs du processus ;
- Un système multi-agents en mesure de mettre en oeuvre cette approche au fil de l'eau des projets de manière transparente aux acteurs métier.

En ce sens notre approche est novatrice car les approches à base d'agents, comme par exemple (Tacla and Barthès, 2003; Gandon et al., 2002b; van Elst and Abecker, 2004; van Diggelen and Dignum, 2006), n'utilisent pas de modèle organisationnel pour identifier, structurer et aider à la capitalisation. Dans (Hassas, 2003) une approche agent est proposée pour construire partir des traces individuelles des concepteurs un sens global pour le collectif. La contribution porte sur la construction d'une sémantique alors que la nôtre porte sur la capitalisation et la réutilisation.
On peut définir deux catégories de SMA dédiés à la gestion des connaissances : les SMAs basés sur la coopération entre les agents afin de traiter les problèmes complexes liés à la nature des connaissances et les SMA proposant des agents assistants qui gèrent les connaissances en fonction des intérêts des acteurs.
Dans la première catégorie de SMA, certains utilisent les agents comme des entités autonomes (à l'image des employés d'une entreprise). Les coopérations entre ces agents génèrent une dynamique complexe. Les plus connus sont les SMAs gérant les réseaux d'échanges Peer-to-Peer (Guizzardi et al., 2003;

Nejdl et al., 2002). Les interactions entre les agents sont alors destinées à l'échange de ressources.

D'autres SMA sont spécialisés dans la recherche et le partage d'informations. Ils utilisent les travaux sur les négociations, la mobilité, et la collaboration entre agents. Ces SMA utilisent des modèles de préférences des utilisateurs pour reconnaître et fournir les informations aux utilisateurs (Preece et al., 2000).

Dans cette catégorie, les agents sont conçus pour exhiber des comportements flexibles, proactifs et réactifs en fonction des besoins de leurs utilisateurs (Tacla and Barthès, 2003; Champin et al., 2004). D'autres travaux complètent cette approche en intégrant la capacité pour les agents de gérer des ressources distribuées et de résoudre des problèmes complexes tels que la coordination de la diffusion des connaissances dans une communauté de pratique . Certains de ces SMAs ont été conçus en complément d'outils de gestion de l'information (workflow, ontologies, système de recherche d'informations, ...) pour donner naissance à des plates-formes telles que FRODO (Abecker et al., 2003), CoMMA (Gandon et al., 2002a) ou encore KRAFT (Preece et al., 2000).

Les travaux présentés dans ce chapitre sont le fruit de la thèse CIFRE de Davy Monticolo (Monticolo, 2008) et ont conduit à des expérimentations conduites dans l'entreprise Zurfluh-Feller qui produit des accessoires de volets roulants et à l'implémentation du système de capitalisation et réutilisation des connaissances dans une plate-forme de travail collaboratif utilisée en entreprise. Par ailleurs, ce prototype fait l'objet d'un transfert technologique par un éditeur de logiciel dans le cadre d'un projet du pôle de compétitivité « Véhicule du futur ».

Une deuxième thèse est en cours sur cette thématique et vise à approfondir l'aspect réutilisation des connaissances. Ce travail est également fondé sur le modèle organisationnel du processus de conception (Miled et al., 2008).

Chapitre 5

Systèmes Multi-Agents Holoniques

5.1 Introduction

Le terme holon a été introduit par le philosophe hongrois Koestler en 1967 pour désigner des structures naturelles ou artificielles qui ne sont ni tout ni parties dans un sens absolu (Koestler, 1967). Selon Koestler un holon doit posséder trois propriétés : (i) être stable (ii) être autonome (iii) être coopératif. La stabilité correspond à la capacité d'auto-organisation. L'autonomie désigne le même concept que l'autonomie des agents. La coopération implique l'habilité à travailler conjointement et d'avoir des buts communs avec d'autres holons. Les organisations holoniques ont été appliquées avec succès à de nombreux problèmes associés à une structure hiérarchique auto-organisée, on peut par exemple citer (Bürckert et al., 2000; Maturana et al., 1999; Brussel et al., 1998; Adam et al., 2002; Ulieru and Geras, 2002). Dans certaines applications à base de SMA, un agent qui apparaît comme une entité unique est en fait composé de plusieurs agents. Cette structure hiérarchique correspond à celle existante dans les systèmes holoniques. Des frameworks, comme PROSA (Brussel et al., 1998) et MetaMorph (Maturana et al., 1999), ont été proposés pour modéliser les concepts holoniques dans certains domaines d'applications spécifiques comme les systèmes manufacturiers. Cependant, ces frameworks sont dépendant d'un domaine d'application et sont difficilement applicables en dehors de ce domaine. Pour remédier à ce problème nous proposons un framework organisationnel pour les Systèmes Multi-Agents Holoniques (SMAH).
Ce chapitre est organisé comme suit : la section 5.2 introduit les principes de modélisation organisationnelle des SMAH et le concept de capacité ajouté au

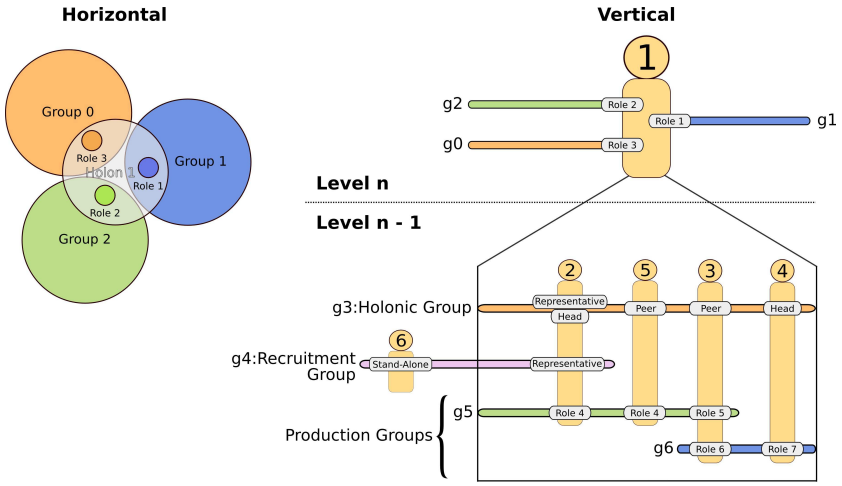


FIG. 5.1 – Décompositions horizontales et verticales d'un holon

modèle RIO. La section 5.3 présente le framework pour SMAH, sa spécification formelle et une architecture pour SMAH inspirée du système immunitaire. La section 5.4 conclut ce chapitre.

5.2 Modélisation organisationnelle des SMAH

5.2.1 Principes

Nous avons fait le choix de décomposer un SMAH selon deux perspectives complémentaires. La première perspective relative aux individus ou holons est la décomposition verticale de la figure 5.1. Chaque holon y est vu comme une entité autonome qui a des buts et qui peut être composé d'autres holons, appelés sous-holons. Un holon composé est appelé super-holon. Un super-holon n'est pas seulement caractérisé par ses membres mais par leurs interactions. Pour décrire ces interactions nous utilisons une décomposition dite horizontale, cf. figure 5.1. Cette décomposition correspond à la relation de mise en oeuvre de rôles par les sous-holons. Chaque sous-holon peut ainsi jouer un ou plusieurs rôles au sein de groupes, instances d'organisations, qui font partis du super-holon. Certains de ces groupes représentent le fonctionnement d'une holarchie (hiérarchie de holons) et sont systématiquement présents quelque-soit l'application. Ces groupes constituent le framework organisationnel que nous proposons pour les SMAH (voir section suivante). Le reste des groupes représentent les aspects dépendants de l'application.

5.2.2 CRIO

Afin d'une part d'accroître la modularité et la réutilisabilité des organisations et des rôles les composant et d'autre part représenter les contributions d'organisations à plusieurs niveaux d'abstractions, nous avons défini la notion de capacité (Rodriguez et al., 2007a). Ce concept s'ajoute aux concepts de rôle, interaction et organisation déjà définis dans le méta-modèle RIO. Une capacité est une abstraction d'un savoir-faire.

Définition 5 *Une capacité est la spécification d'une transformation d'une partie du système en cours de conception. Cette transformation garantit certaines propriétés si le système satisfait certaines contraintes.*

En d'autres termes, une capacité peut être considérée comme la spécification des pre et post conditions propres à l'accomplissement d'un but. Une capacité peut être utilisée de manière duale pour décrire :

- ce qu'un comportement, rôle ou organisation, est capable de faire,
- ce qu'un rôle requiert pour définir son comportement.

La figure 5.2 représente le méta-modèle CRIO. Le niveau abstrait (organisationnel) et au niveau concret (agent), sont représentés. Au niveau organisationnel le concept de capacité est introduit et est lié aux rôles. Un rôle peut en effet externaliser la définition d'un savoir-faire et requérir d'une entité qui souhaite mettre en oeuvre ce rôle de posséder cette capacité. Par exemple, le rôle de manager dans le réseau contractuel peut externaliser la capacité de choisir la meilleure proposition parmi un ensemble d'offres (Gaud et al., 2008b). De manière duale une capacité peut décrire ce qu'un ensemble de rôles en interaction, c'est-à-dire une organisation, est capable de réaliser au niveau système. L'organisation colonie de fourmis est ainsi capable de réaliser la capacité « trouver un plus court chemin » (Rodriguez et al., 2007a).
Au niveau concret le concept de groupe instancie le concept d'organisation. Un groupe est composé de holons. Un holon est une entité auto-similaire qui peut être composée de holons. Un holon « atomique » ou non composé est, pour nous, un agent. Un holon composé est défini par les groupes de holons qui le composent. Les holons possèdent des réalisations de capacités. C'est-à-dire qu'ils peuvent réaliser les transformations décrites par les capacités. Un holon pouvant représenter un collectif, les groupes peuvent également exhiber une implémentation de capacité. Dans ce sens la capacité décrit ce qu'une instance d'organisation est capable de faire.

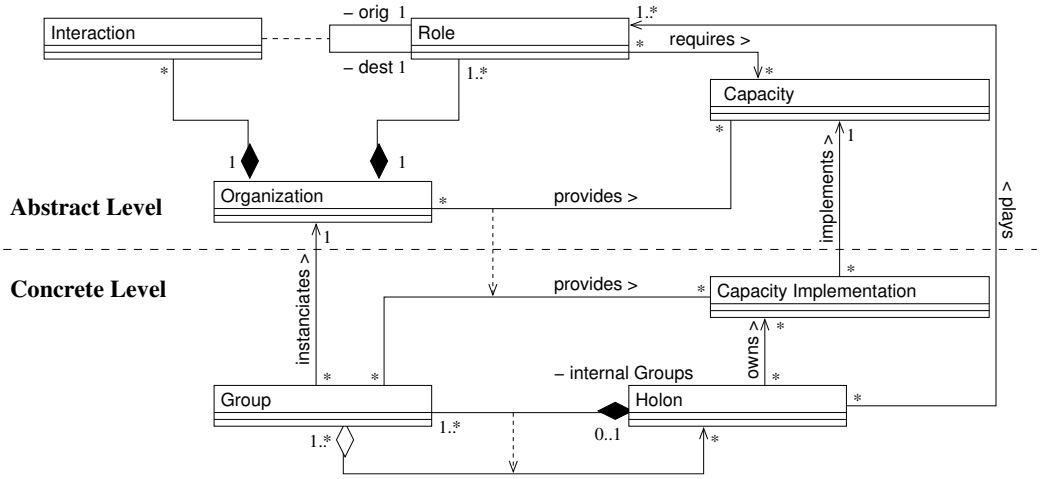


FIG. 5.2 – Méta-modèle CRIO

La classe suivante spécifie formellement le concept de capacité. Elle introduit les ensembles d'entrées et et de sorties qui précisent ce qui va être transformé par la capacité. Les contraintes *requires* et *ensures* spécifient respectivement ce qui doit être vrai avant que la transformation débutent et ce qui est vrai lorsque la transformation est terminée.

Capacity

$$inputs : \mathbb{F}\ Attribute$$
$$outputs : \mathbb{F}\ Attribute$$
$$requires : \mathbb{F}\ Constraint$$
$$ensures : \mathbb{F}\ Constraint$$

$$capacityCall(name) \land (\bigwedge_{r \in requires}) \Rightarrow \Diamond(\bigwedge_{e \in ensures} e)$$

La classe *CapacityImpl* spécifie une implémentation de capacité. Cette classe introduit un attribut qui précise quelle capacité est implémentée et un sous-schéma *behaviour* qui spécifie à l'aide d'un statechart la réalisation de la capacité.

CapacityImpl

$$implements : Capacity$$

behaviour

Le concept de capacité et son pendant, l'implémentation, permettent de modéliser des savoir-faire et rendent de fait les rôles plus modulaires et réutilisables. Une organisation ainsi paramétrée par des capacités peut être utilisée dans des contextes différents avec des implémentations différentes. Cela constitue une sorte de schéma de conception organisationnel (Cossentino et al., 2008b).
Le concept de capacité peut également être utilisé comme relation d'abstraction en vue de faciliter des preuves de propriétés des organisations (Gaud et al., 2008b). En effet, par définition la capacité abstrait une partie de comportement.

5.3 Framework holonique

5.3.1 Les organisations des SMAH

La structure d'un SMAH peut être considérée comme un ensemble de niveaux imbriqués. Cette structure hiérarchique est appelée holarchie. On désigne par super-holon le holon de niveau supérieur et par sous-holon les membres qui le composent. Le comportement des membres d'un super-holon et leurs interactions sont décrits avec le méta-modèle CRIO. La structure interne choisie pour représenter les interactions entre sous-holons est celle de groupe modéré (Gerber et al., 1999). Dans cette structure un ou plusieurs sous-holons représentent le super-holon pour le niveau supérieur. Cette structure permet un grande nombre de configurations possibles. Les représentants peuvent être sélectionnés par différents mécanismes comme un vote ou par des règles pré-définies. Cette sélection demeure du ressort du problème traité et n'est donc pas intégrée dans le framework.
Un groupe modéré est modélisé par l'organisation de la figure 5.3. En tant que représentant, un holon joue le rôle Head. Selon les objectifs du super-holon les responsabilités du Head peuvent varier de tâches administratives à la prise de décisions. Le rôle Head n'est pas nécessairement joué par un unique holon. Les membres qui ne jouent pas le rôle Head jouent soit le rôle Part soit le rôle MultiPart. Ce dernier rôle correspond à des holons qui sont membres de plusieurs super-holon. Une fois un holon accepté au sein d'un super-holon, son autonomie peut être réduite pour satisfaire les obligations du super-holon.
Quand un holon intègre une organisation d'un SMAH, il ne fait partie d'aucun super-holon. Ce statut est représenté par le rôle StandAlone. Le rôle StandAlone représente la façon dont les non-membres sont perçus par les super-holons. Les holons jouant le rôle StandAlone interagissent avec les

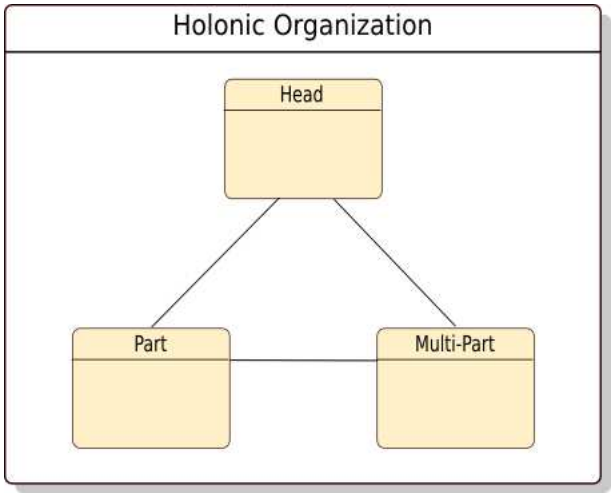


FIG. 5.3 – Organisation groupe modéré

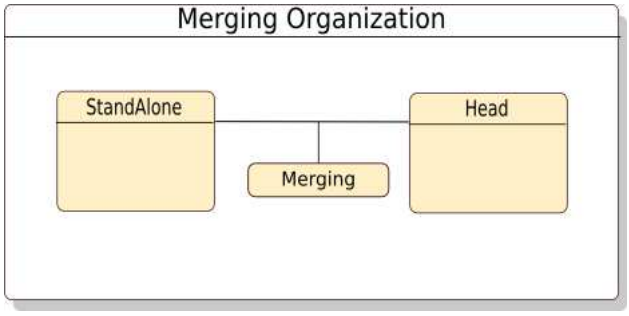


FIG. 5.4 – Organisation de fusion

représentants (heads) des super-holons de façon à coopérer ou rejoindre un super-holon. L'interaction dont le résultat est une intégration d'un StandAlone dans un super-holon est appelée Merging et est représentée dans le diagramme CRIO de la figure 5.4 Si l'intégration est acceptée, le holon devient membre de ce super-holon. A partir de ce moment il peut interagir directement avec les membres du super-holon.

5.3.2 Auto-organisation au sein du framework

Pour permettre aux holons de décider dynamiquement du changement de leurs rôles, nous avons défini des critères de satisfaction. Ces critères sont numériques et basés sur l'évaluation de la satisfaction des buts du holon. Chaque rôle du framework définit des critères particuliers. La satisfaction personnelle ou Self Satisfaction (SS_h) d'un holon h est produite par ses propres actions. La satisfaction collaborative ou Collaborative Satisfaction (CS_{Hh}) est la satisfaction produite pour un holon h par ses collaborations avec les autres membres du super-holon H. La satisfaction accumulée ou Accumulative Satisfaction (AS_h) est la satisfaction produite pour le holon h par ses collaborations avec les membres de plusieurs super-holons.

$$AS_h = \sum_p CS_h^p \quad \forall\, p \in superholon(h) \tag{5.1}$$

où la fonction superholon renvoie les super-holons de h. La satisfaction instantanée d'un holon h ou Instant Satisfaction (IS_h) est la combinaison de ces différents critères et est définie comme

$$\forall\, h \in HMAS \quad IS_h = \begin{cases} CS_h + SS_h & if\ R_h = Part \vee R_h = Head \\ AS_h + SS_h & if\ R_h = MultiPart \\ SS_h & if\ R_h = StandAlone \end{cases} \tag{5.2}$$

Où R_h est le rôle joué par le holon h. A l'aide de ces définitions, nous avons établi les évolutions possibles entre les rôles holoniques. Ces évolutions sont spécifiées par l'automate de la figure 4.5. Necessary Satisfaction (NS) représente le seuil de satisfaction minimum nécessaire pour accomplir un but. Chaque transition est étiquetée par une condition entre crochet. Si la condition est vraie alors la transition est franchie. Les conditions C_{PH}, C_{MH} et C_{MP} est une condition dépendante du problème. Dans la section suivante, nous donnons la spécification formelle des rôles holoniques.

5.3.3 Spécification formelle

La classe de base dont hérite les autres est la classe HolonicRole. Elle spécifie un booléen *satisfied* qui est vrai si et seulement si l'entité qui joue le

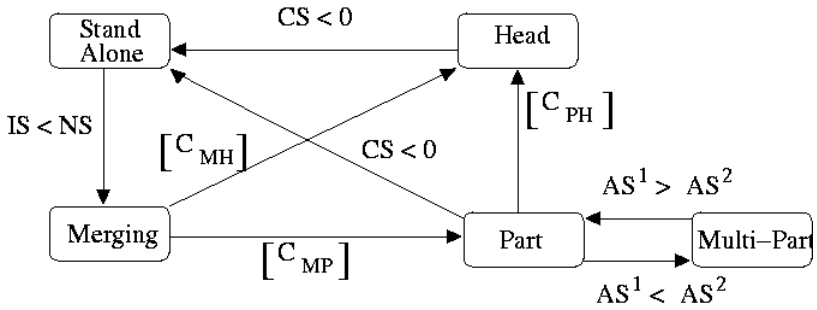


FIG. 5.5 – Transitions entre rôles holoniques

rôle est satisfaite. Un rôle holonique est également caractérisé par les critères de satisfaction SS et IS précédemment définis, une tâche courante et le seuil de satisfaction minimum NS.

HolonicRole

Role

$satisfied : \mathbb{B}$
$SS, IS : \mathbb{R}$
$current : Task$
$NS : Task \mapsto \mathbb{R}$

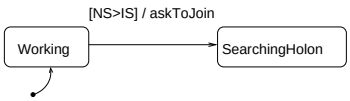
La classe StandAlone hérite de HolonicRole et raffine le critère de satisfaction ainsi que la condition pour quitter ce rôle. Lorsque le rôle cherche à rejoindre un super-holon, il envoit une requête par l'opération *askToJoin* et sélectionne la meilleure proposition a l'aide de l'opération *getBestAnswer*.

StandAlone

HolonicRole

$leaveCondition = \{SS < NS\}$
$IS = SS$

behaviour



askToJoin

$mergeRequest'$

getBestAnswer

$answers? : \mathbb{P}\, Answer$
$bestAnswer! : Answer$

$b : Answer \mid \forall\, c \in answers? \bullet better(b, c)$
$bestAnswer! = b$

L'interaction Merging spécifie l'interaction entre un StandAlone et un Head. Cette interaction est déclenchée par un *askToJoin* de la part d'un StandAlone et est examinée au travers de l'opération *examineFusionRequest* d'un Head.

Merging
Interaction

$REJECTED : \mathbb{B}$
$answers : \mathbb{P}\, Answer$

$orig \subseteq StandAlone$
$dest \subseteq Head$
$aux = \varnothing$
$examineFusionRequest \in requires(dest)$
$E = askToJoin \parallel examineFusionRequest$

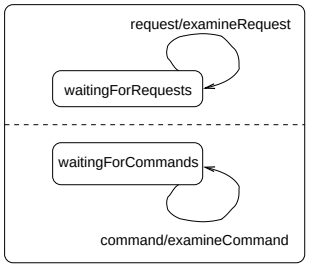
Le rôle Part raffine les critères de satisfaction et est spécifié par un ensemble de Head qui sont les Head du super-holon auxquels il appartient.

Part
HolonicRole

$others : \mathbb{P}\, Holon$
$myHeads : \mathbb{P}\, Head$
$CS : \mathbb{R}$

$IS = CS + SS$
$leaveCondition = \{(IS < NS) \vee (CS < 0)\}$
$myHeads \subseteq others$

behavior



examineRequest

$\Delta engagements$
$request? : \mathbb{P}\ Service$

$engagements' = engagements$
$\lor\ engagements' = engagements$
$\bigoplus \bigcup_{\forall\, r \in request?} (r, true)$

examineCommand

$\Delta engagements$
$command? : \mathbb{P}\ Service$

$\forall\, c \in command? \bullet$
$availableServices(c) = false \land engagements' = engagements$
$\bigoplus \bigcup_{\forall\, c \in command?} (c, true)$

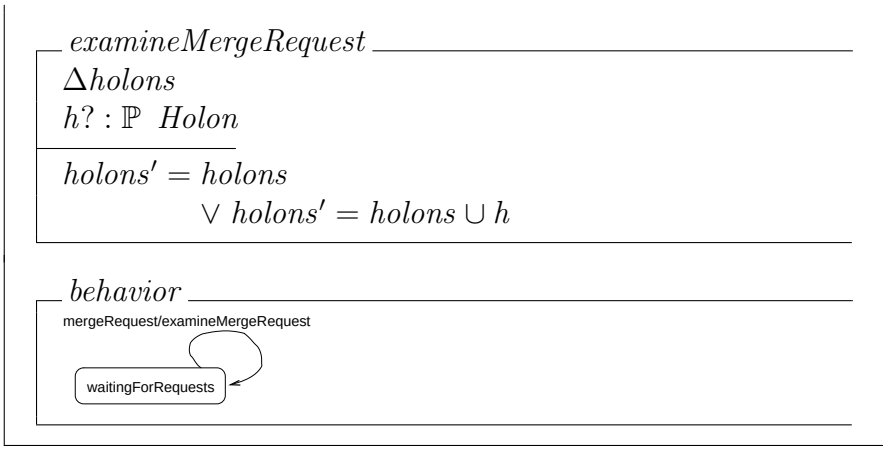
La classe Head spécifie le rôle Head. Les critères de satisfaction sont raffinés et l'ensemble holons spécifie les sous-holons membres de ce super-holon.

Head

HolonicRole

$holons : \mathbb{P}\, Holon$
$CS : \mathbb{R}$

$IS = CS + SS$
$leaveCondition = \{(IS < NS) \lor (CS < 0)\}$



Cette architecture a été appliquée à plusieurs problèmes. Notamment un SMAH pour la résolution de problème (Rodriguez et al., 2003) et un SMAH dédié à la simulation de système complexe (Rodriguez et al., 2007c). Une API JAVA a été écrite pour permettre l'implémentation de SMAH (Rodriguez et al., 2006b). Cette API permet également de distribuer sur un ensemble de machines le déploiement de SMAH.
La spécification formelle de cette architecture a permis de démontrer certaines propriétés (Rodriguez et al., 2007b). Parmi ces propriétés nous avons prouvé qu'un SMAH réalisé selon ce framework exhibe des propriétés d'auto-organisation.

5.3.4 Architecture holonique

Nous avons défini une architecture pour SMAH sur la base de l'architecture immunitaire présentée au chapitre 3.4.2 (Rodriguez et al., 2005b; Hilaire et al., 2008b). L'analogie faite consiste à assimiler un holon à un système immunitaire de type réseau idiotypique dont les anticorps peuvent être décomposés en systèmes immunitaire d'un niveau inférieur. Ce principe, appliqué à l'exemple des robots footballeurs, est décrit dans la figure 5.6.
Celle-ci présente une holarchie composée de quatre niveaux. Chaque holon est décomposé en sous-holons. Le niveau le plus haut représente une équipe de robots footballeurs. Ce niveau a pour responsabilité de choisir une stratégie. Chaque anticorps du deuxième niveau représente une stratégie en terme d'affectation de rôles pour les robots (comme gardien de but, attaquant, ...). Le troisième niveau définit pour chaque rôle un comportement spécifique. Dans l'exemple ce comportement se résume à un point destination pour le robot. Enfin, le dernier niveau associe pour chaque point destination un ensemble de trajectoires aptes à amener le robot au point destination en évitant les

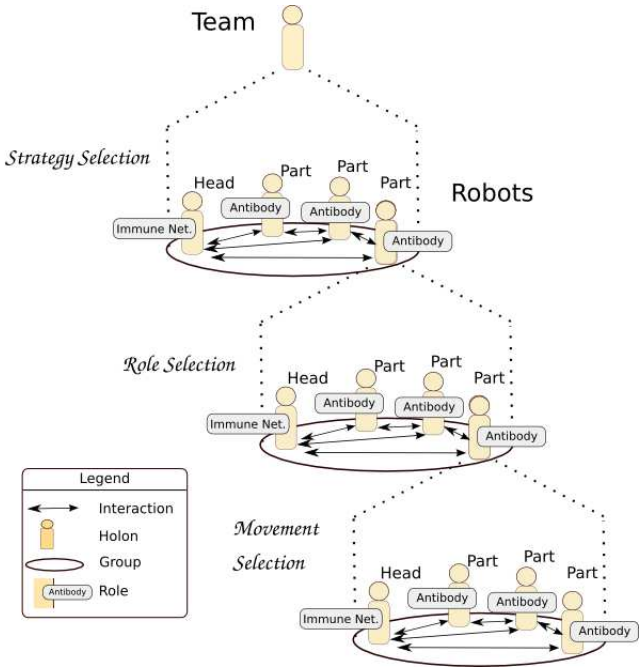


FIG. 5.6 – Holarchie à base de systèmes immunitaires

obstacles et en respectant les règles du jeu.
Le rôle du système immunitaire dans cette architecture est la sélection d'action. Le résultat de cette sélection d'action à un niveau donné de la holarchie est donné en entré au niveau immédiatement inférieur. On a ainsi une coordination des holons à plusieurs niveaux d'abstraction et des caractéristiques d'adaptabilité intrinsèques au système immunitaire et propres à chaque niveau.

5.4 Conclusion

Dans ce chapitre nous avons présenté notre contribution à l'ingénierie de SMAH. Nous avons proposé une approche de décomposition basé sur les concepts du modèle RIO auxquels nous avons ajouté le concept de capacité qui permet de décrire les relations de contribution entre organisations à différents niveaux d'abstraction. Un framework a été défini pour l'analyse et le développement de SMAH. L'ensemble de ces concepts a fait l'objet d'une spécification formelle avec le formalisme OZS.
Les perspectives futures concernant les SMAH peuvent prendre plusieurs directions. J'ai participé à un travail précurseur concernant la simulation multi-

niveaux à base de holons (Gaud et al., 2009). Les SMAH peuvent également être appliqués à de nombreux autres domaines comme la gestion des connaissances ou la résolution de problèmes. Nous sommes par contre conscient de la difficulté d'analyse et de développement qu'engendre ces systèmes. Pour pallier à ce problème le chapitre suivante propose des méthodologies pour SMAH.

Chapitre 6

Méthodologie

6.1 Introduction

Une méthodologie est un plan d'action qui décrit les étapes pour atteindre un (ou plusieurs) objectif(s). Lors des débuts du paradigme des SMA les développeurs utilisaient des méthodes ad-hoc ce qui nuisait à la qualité et à la réutilisabilité des systèmes développés. A l'heure actuelle, les travaux sur les méthodologies ont acquis une certaine maturité même si aucune ne couvre l'ensemble des problèmes que les SMA peuvent traiter (Henderson-Sellers and Giorgini, 2005). Cette absence de consensus vient du fait que, à la différence des méthodologies orientées objets, les méthodologies orienté agent doivent prendre en compte les aspects tels que : l'autonomie et les buts propres des agents, la dynamique de l'environnement, l'auto-organisation, le fait qu'un SMA soit ouvert, ... Ces différents aspects sont difficiles à traiter simultanément au sein d'une méthodologie et peuvent être traités de façons différentes. Nos travaux sont orientés vers des SMA à base d'abstraction organisationnelle. Certaines méthodologies ont pris le partie de décrire des SMA sans concepts organisationnels. Adelfe (Bernon et al., 2005a), par exemple, vise l'analyse et la conception de SMA adaptatifs.
Pour notre part, nous nous sommes concentrés sur les méthodologies propres à l'analyse et au développement de systèmes complexes. Pour cela notre hypothèse issu de (Simon, 1996) est de considérer un système complexe comme une hiérarchie au sens holonique où chaque holon (ou sous-système) est dit presque décomposable c'est-à-dire que les interactions entres les sous-systèmes sont faibles mais non négligeables. En d'autres termes (i) le comportement de chaque sous-système à court terme est indépendant du comportement des autres sous-systèmes (ii) à long terme le comportement d'un sous-système dépend faiblement de la somme du comportement des autres

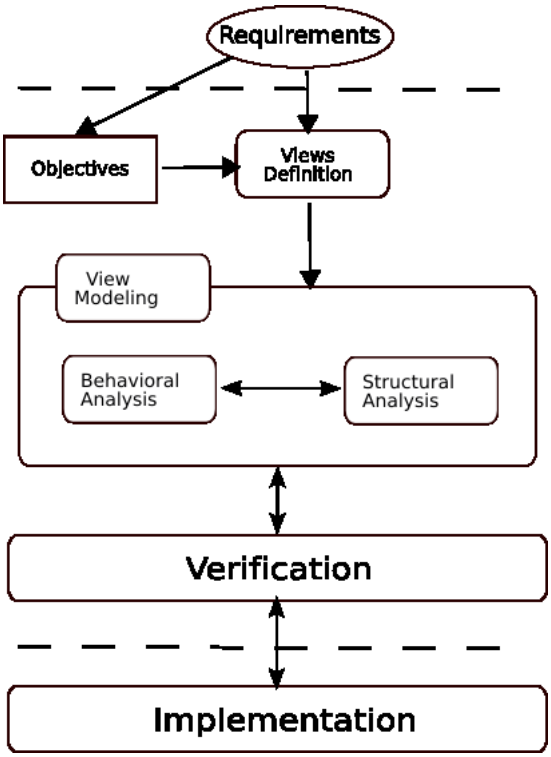


FIG. 6.1 – Approche multi-vues

sous-systèmes.
Nous avons fait deux propositions la première, est une approche orientée multi-vues, est présentée dans la section 2. La deuxième est une méthodologie, appelée ASPECS, issue d'une collaboration avec l'équipe SMA dirigée par Massimo Cossentino[1] est présentée en section 3. La section 4 conclut ce chapitre.

6.2 Approche multi-vues

6.2.1 Principes

La première approche présentée dans ce chapitre est basée sur la notion de vue (Nuseibeh et al., 2003; Koukam et al., 2003). Cette approche repose sur le principe de décomposition d'un système en vues ou perspectives. Chaque vue agit comme un filtre qui permet d'isoler certains aspects du système en

[1]chercheur de l'ICAR/CNR Palerme.

cours d'étude. En effet, les systèmes complexes sont caractérisés par un grand nombre d'entités en interaction. Une façon de traiter la complexité est de diviser l'analyse du système en vues ou perspectives. Ce procédé évite de traiter l'ensemble des vues du système simultanément. De plus, la décomposition en vue permet la séparation des préoccupations (Dijkstra, 1982) et réduit le nombre d'entités étudiées simultanément. Dans notre approche chaque vue sera in fine modélisée par une holarchie telle que définie dans le chapitre 3. L'approche est schématisée par la figure 6.1. Les entrées de cette approche sont les besoins (l'analyse des besoins n'est pas couverte). A partir des besoins, on doit identifier des objectifs globaux pour le système. Chaque objectif est lié à une vue particulière. Une fois les vues identifiées, l'étape suivante consiste en leurs définitions. Cette étape, désignée par View modeling dans la figure 6.1, est composée de deux blocs : analyse comportementale et analyse structurelle. Ces deux blocs décrivent les deux aspects inhérents à la holarchie en charge de l'accomplissement de l'objectif : son comportement et la façon dont sont structurés les holons qui la composent.

Analyse structurelle : Décrit la relation de composition entre les holons et par là définit la holarchie. Par exemple, une *université* est composée de *Départements* et de *Laboratoires*. Un *Département* regroupe des *Etudiants* et des *Enseignants*.

Analyse comportementale : Décrit les comportements d'un ensemble de holons en terme de rôles, interaction et organisations. Par exemple, pour une *université*, les *Enseignants* interagissent avec des *Etudiants* en instanciant une organisation *Cours*.

Ces deux phases, ainsi qu'indiqué sur la figure 6.1 sont liées et nécessitent plusieurs itérations avant d'aboutir à une holarchie complètement définie. L'ordonnancement des blocs comportementaux et structurels est dépendant du problème traité. Par exemple, un simulateur est sensé reproduire un système réel la première étape consiste donc à la définition de la partie structurelle. D'autre part, un système d'aide à la décision est défini par les objectifs en terme d'optimisation ou de résolution de problèmes, dans ce cas l'accent est mis sur la partie comportementale.
Pour illustrer cette approche, nous avons choisi de modéliser un simulateur pour une grande entreprise dédiée à la construction automobile. Le site de cette entreprise est étendu sur plus de 250 hectares. Afin d'assurer l'approvisionnement de la production, 1700 véhicules entrent tous les jours sur le site qui est entouré de trois villes et est traversé par une autoroute et une voie ferrée. Les objectifs de ce simulateur sont d'évaluer le trafic routier au sein de l'usine et d'identifier les groupes de bâtiments liés par des flux de produits finis ou semi-finis. Le modèle correspondant met en avant ces deux

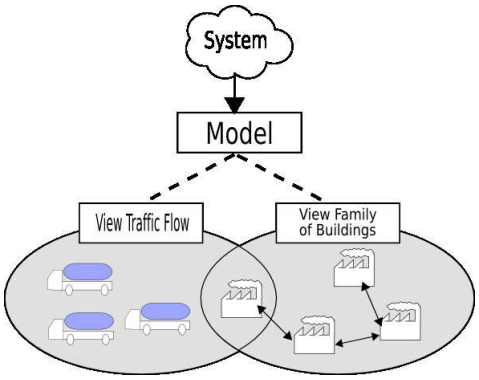


FIG. 6.2 – Vues du simulateur

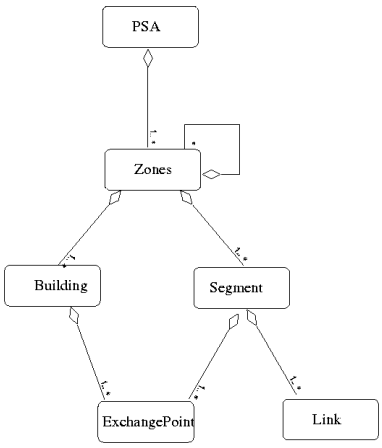


FIG. 6.3 – Relations de composition entre entités de la vue évaluation du trafic

objectifs en associant à chacun une vue particulière. La figure 6.2 illustre ces deux vues qui sont respectivement dédiées à la simulation de trafic et à l'identification de groupes de bâtiments.

6.2.2 Analyse structurelle

Quand cette perspective sur une vue a été choisie, l'objectif est d'obtenir une structure préliminaire de la holarchie correspondant à la vue. Pour obtenir cette structure on considère le système en cours d'étude au travers du filtre de la vue. On ne considère que les entités pertinentes dans ce contexte. Par exemple, pour la vue évaluation du trafic, les entités pertinentes sont : les

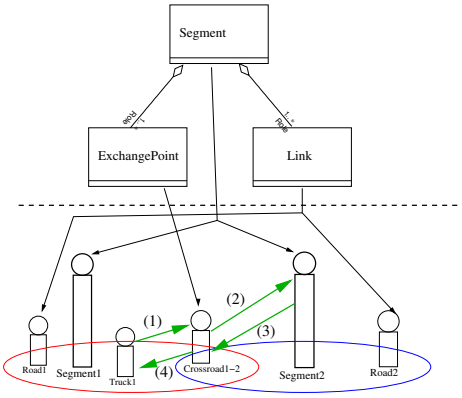
camions en tant que véhicules, les routes avec la signalétique, les croisements et les bâtiments ne seront considérés que comme des zones de passage pour les camions. La figure 6.3 présente les relations structurelles entre ces différentes entités. Le choix de modélisation est de considérer l'usine comme composée de zones qui contiennent des zones de plus faible granularité. Chaque zone est composée de segments routiers et de bâtiments. Le passage d'un segment à un autre ou à un bâtiment se fait par un point d'échange dont des cas particuliers sont les portes et les carrefours. Cette décomposition récursive définit la structure de la holarchie.

6.2.3 Analyse comportementale

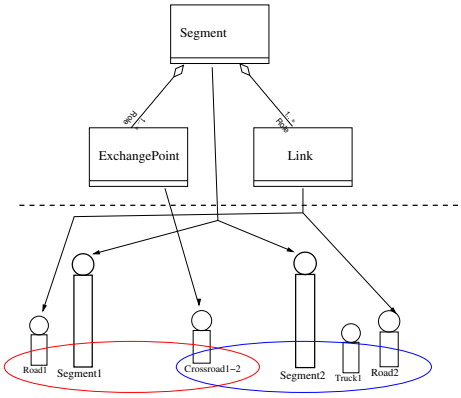
Le principe de l'analyse comportementale est d'associer un comportement pour chaque vue en termes de rôles en interactions. Dans notre exemple il faut associer un comportement à la vue évaluation du trafic et à la vue identification des groupes de bâtiments. Pour la vue évaluation du trafic les entités pertinentes sont les véhicules circulant dans l'usine. Ces véhiculent traversent les zones, utilisent les segments et les points d'échanges définis précédemment dans l'analyse structurelle. Pour définir les comportements des véhicules, il faut d'une part associer à chaque véhicule un modèle de conduite qui lui permet de décider d'une destination et du chemin qui va l'y amener. D'autre part, il faut définir les interactions de chaque véhicule avec son environnement composé des autres véhicules et de la structure définie dans la figure 6.3. Un scénario type de ces interactions est présenté dans la figure 6.4.

6.2.4 Intégration des vues

Une fois l'analyse des vues réalisée il faut intégrer ces différents aspects du système. Pour cela, nous proposons l'utilisation de la relation de mise en oeuvre des rôles. En effet, chaque vue apporte un modèle organisationnel différent qui va être ensuite dynamiquement mis en oeuvre au sein d'holons jouant des rôles dans ces différentes organisations. Par exemple, un holon camion joue le rôle de véhicule qui se déplace sur le site (vue trafic) et joue également le rôle de transporteur de produits (vue famille de bâtiments).
L'utilisation de spécification formelle dans ce cadre permet d'une part de représenter de manière non ambigüe les différentes vues mais aussi de mener des vérifications sur l'intégration de ces vues.



Un camion se déplace sur le segment 1



Le camion demande le passage au point d'échange

FIG. 6.4 – Exemple de séquences de déplacements

6.3 ASPECS

6.3.1 Motivations

La méthodologie ASPECS est fondée sur des concepts de l'ingénierie de méthodes (Henderson-Sellers, 2003). Elle s'appuie sur un processus itératif et incrémental comme c'est le cas pour de nombreuses méthodes de développement de SMA (Kolp et al., 2006; Bernon et al., 2005a; Pavon et al., 2005; Padgham and Winikoff, 2002a). ASPECS s'appuie également sur les concepts de l'Ingénierie Dirigée par les Modèles (Miller and Mukerji, 2003). En particulier, les concepts manipulés au cours du processus sont représentés au sein de méta-modèles. Ces méta-modèles sont au nombre de trois en référence aux trois niveaux proposés par l'OMG (CIM, PIM et PSM). Des mécanismes de transformation et de raffinement entre ces modèles sont également proposés. L'objectif de cette méthodologie est d'analyser et de concevoir des SMA Holoniques pour des systèmes complexes, ouverts et exhibant des caractéristiques dynamiques. Les concepts à la base de cette méthodologie sont ceux déjà abordés aux chapitres 3 : les agents, les holons et le méta-modèle CRIO. Le processus d'ASPECS couvre l'intégralité du processus d'analyse et de développement de logiciel depuis l'analyse des besoins jusqu'à l'implémentation, le déploiement grâce à la plateforme JANUS (Gaud et al., 2008a).
La notation employée est basée sur UML en tant que langage de modélisation. Pour satisfaire les besoins spécifiques aux agents et à l'approche organisationnelle, la sémantique et la notation UML ont été étendues en utilisant des profils.

La structure du processus d'ASPECS est basée sur le standard SPEM (Software Process Engineering Metamodel). La spécification de l'OMG (SPEM, 2007) propose l'idée qu'un processus de développement logiciel est basée sur la collaboration entre entités abstraites, appelées *Rôles*, qui effectuent des *Activités*, sur des entités concrète appelées *Produits*. Selon cette approche, le processus d'ASPECS s'appuie sur une hiérarchie à trois niveaux : *Phases*, *Activités* et *Tâches*. Une *Phase* construit un produit composite, composé d'un ou plusieurs composants qui peuvent être de types différents. Une *Activité* construit un produit principal, comme un diagramme ou un document texte, et est composée de *Tâches*. Une *Tâche* contribue à la réalisation d'un produit. Cette contribution peut prendre la forme d'une réalisation de partie de produit et peut par la-même instancier, mettre en relation ou raffiner des éléments du méta-modèle.

6.3.2 Processus

Le cycle de vie d'ASPECS est composé de trois phases décrites ci-dessous. Ces phases sont également représentées par la figure 6.5 où le détail des activités est également précisé.

1. La phase **Besoin du Système** a pour but l'identification d'une hiérarchie d'organisations dont le comportement global tend a la satisfaction des besoins du système. Cette phase débute par une activité de description des besoins au cours de laquelle les besoins sont identifiés au travers de techniques usuelles, comme , par exemple, les cas d'utilisations. La connaissance du domaine et le vocabulaire spécifique sont ensuite collectés et définis lors de l'activité Ontologie du Problème. Les besoins sont alors associés à des organisations en charge de les réaliser (activité Identification des Organisations). La hiérarchie d'organisations ainsi produite est raffinée et étendue par la suite lors d'itérations successives. Le comportement de chaque organisation est réalisé par un ensemble de rôles en interactions dont les buts contribuent en partie à la réalisation des besoins du système. Afin de concevoir des organisations modulaires et réutilisables, les rôles sont définis sans faire d'hypothèses sur la structure des agents qui vont les mettre en oeuvre. Pour atteindre cet objectif, le concept de capacité a été introduit (Rodriguez et al., 2007a). Une capacité est une description abstraite d'un savoir-faire ou d'une compétence d'un rôle. Chaque rôle peut nécessiter un certain nombre de capacités pour être mis en oeuvre. En effet, une entité doit posséder ces capacités pour le mettre en oeuvre. La dernière activité de cette phase consiste à identifier ces capacités et à les lier

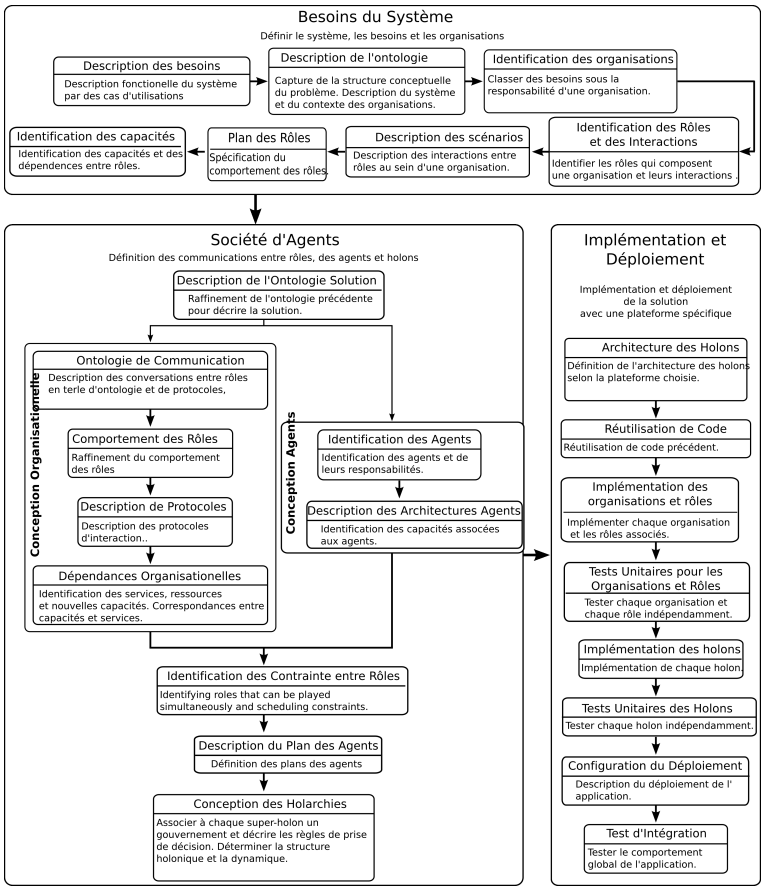


FIG. 6.5 – Aperçu du cycle de vie d'ASPECS

aux rôles.

2. La phase **Société d'Agents** a pour but de concevoir une société d'agents dont le comportement global est capable d'apporter une solution aux problèmes identifiés et décrits dans la phase précédente. L'objectif est de produire un modèle de la solution en termes d'interactions sociales et de dépendances entre entités (holons et agents). Les éléments précédemment identifiés comme l'ontologie, les rôles et les interactions sont raffinés. A la fin de la conception, la hiérarchie organisationnelle est associée à une holarchie (hiérarchie de holons) en charge de réaliser les comportements attendus. Chaque organisation précédemment identifiée est instanciée sous forme de groupes. Les rôles correspondants sont associés à des holons ou agents. Cette dernière activité doit également décrire les règles qui gouvernent la prise de décision au sein d'un superholon et la dynamique du système.
3. La phase d'**Implémentation et Déploiement** a pour but d'implémenter la solution de la phase précédente en utilisant une plateforme spécifique et de configurer le déploiement de cette solution. Pour notre cas, une plateforme a été développée : JANUS (Gaud et al., 2008a). Cette phase utilise les concepts de cette plateforme pour implémenter les éléments du modèle issu de la phase précédente. La génération des tests et la réutilisation de code sont également prises en compte.

6.3.3 Activités

Chaque activité est décrite selon la structure donnée dans la table 6.1. La table 6.2 décrit l'activité description de scénario de la phase Besoins du Système. Pour une description des autres activités et d'autres exemples le lecteur peut consulter le site web d'ASPECS [2].

6.4 Conclusion

Dans ce chapitre nous avons présenté deux méthodologies dédiées à l'ingénierie de SMAH. La première propose une approche de décomposition par vues du problème et une intégration de ces vues par le biais d'holarchies différentes. La deuxième méthodologie, ASPECS , couvre les phases de l'analyse à l'implémentation et au déploiement de SMAH. Cette dernière méthodologie est plus détaillée du point de vue du processus et des activités. De plus nous avons défini des profils UML et identifié des règles méthodologiques pour

[2] http ://www.aspecs.org

TAB. 6.1 – Structure d'une activité

But	Les objectifs de l'activité .
Entrées	Les éléments nécessaires à la réalisation de l'activité.
Sorties	Ce qui est produit par l'activité.
Eléments du méta-modèle	Chaque activité peut définir ou associer des éléments du méta-modèle.
Ce qui doit être réalisé	Explications décrivant le contenu de l'activité.
Règles méthodologiques	Heuristiques, règles et éléments pouvant aider à la réalisation de l'activité.
Exemple	Quelques exemples de réalisation de l'activité.

aider le concepteur.
A l'heure actuelle nous travaillons à la définition d'un environnement d'aide à l'analyse et à la conception de SMAH. Cet environnement est basé sur l'IDM (Miller and Mukerji, 2003). L'idée est d'utiliser les méta-modèles définis au sein d'ASPECS pour générer des éditeurs graphiques aptes à manipuler les modèles et de définir des transformations pour automatiser certaines activités d'ASPECS et permettre la production automatique de spécification formelles, documents textes ou squelettes de code à partir des sorties d'activités.

TAB. 6.2 – Activité Description de Scénario

But	Décrire la séquence d'interactions entre rôles impliqués dans chaque scénario. Un scénario décrit comment les rôles interagissent et se cordonnent pour satisfaire les buts affectés à l'organisation. La description de scénario intervient après les activités d'identification d'organisations et d'identification d'interactions. A ce stade il est possible d'affecter à chaque besoin une organisation et un ensemble de comportements en interactions.
Entrées	Les scénarios sont déduits des scénarios d'utilisation du système et des sorties des activités d'identification des organisations et des interactions.
Sorties	Les scénarios sont décrits en utilisant une version étendue des diagrammes de séquence UML. Chaque objet dans le diagramme représente un rôle avec la syntaxe `:< role name >:< organisation name >`. Chaque message dans le diagramme représente soit une influence extérieure générée par l'environnement soit une communication entre rôles.
Eléments du méta-modèle	Define(Scenario), Quote(Interaction), Quote(AbstractRole), Quote(Relate(AbstractRole, Interaction)).
Ce qui doit être réalisé	Un scénario explore les différentes possibilités pour des rôles en interaction de réaliser les buts de l'organisation. Pour ce faire les interactions déjà identifiées sont assemblées pour composer un ensemble d'interaction entre les rôles de l'organisation. Explications décrivant le contenu de l'activité.
Règles méthodologiques	Les règles méthodologiques appliquées lors de l'analyse orienté objet peuvent être appliquées ici car les principales différences résident dans la sémantique des objets manipulés et la notation légèrement étendue.

Chapitre 7

Conclusion

7.1 Bilan

Ce mémoire est le résultat de mon parcours de recherche durant lequel nous avons abordé la thématique générale de l'ingénierie de SMA avec une perspective organisationnelle et selon deux perspectives : semi-formelle et formelle. Pour cela nous avons défini un langage multi-formalismes, OZS, et des concepts organisationnels que nous avons appliqués à différentes classes de SMA. Nos contributions peuvent être résumées selon les points suivants.

Formalisme OZS

Le langage multi-formalismes, OZS, est défini en intégrant les formalismes Object-Z (Duke et al., 1991) et statecharts (Harel, 1988). Cette intégration consiste en la définition d'une syntaxe et d'une sémantique pour OZS. OZS permet de spécifier formellement les aspects réactifs et transformationnels des SMA. La sémantique, définie avec des systèmes de transitions permet d'utiliser des outils de validation et de vérification.

Concepts organisationnels

Du point de vue organisationnel nous avons identifié un ensemble de concepts apte à la décomposition d'un SMA en termes de comportements en interaction. Cette décomposition permet notamment une analyse plus fine, la modularité et la réutilisation. Pour chaque concept identifié une classe exprimée avec le langage OZS donne une sémantique formelle. L'ensemble de ces classes constitue un framework pour la spécification organisationnelle et formelle de SMA. En effet, un concepteur peut réutiliser ce cadre pour sa propre étude. Nous avons appliqué ce framework pour la décomposition d'architectures d'agents existantes selon un processus de rétro-ingénierie afin d'obtenir pour chacune un modèle organisationnel et d'étudier leurs propriétés. Ces études nous ont permis de dégager les propriétés principales de ces architectures et

d'isoler des composants réutilisables.

Approche de capitalisation et de réutilisation des connaissances

A partir du constat que le processus de conception de produits industriel peut être vu comme un SMA où les agent sont hétérogènes et distribués, nous avons modélisé ce processus selon un modèle organisationnel. Ce modèle est un référentiel pour la cartographie des connaissances. Il nous a permis d'identifier les connaissances à capitaliser et de définir la structure et l'ontologie sous-jacente d'une mémoire de projet qui stocke et indexe les connaissances capitalisées. Un SMA gère de manière semi-automatique la capitalisation en s'appuyant sur le modèle organisationnel du processus de conception.

Systèmes Multi-Agents Holoniques

Nous avons abordé le thème des systèmes holoniques en proposant de voir un holon comme un agent particulier qui peut contenir d'autres holons. Pour analyser et concevoir de tels systèmes nous avons proposé une décomposition selon la relation d'appartenance d'un holon au sein d'un super-holon et selon les groupes d'interactions, instances d'organisations. Un framework complet a été ainsi conçu pour l'analyse et la conception de SMAH. Une architecture adaptative pour ce type d'agent a été proposée.

Méthodologies

En terme de méthodologies pour Systèmes Multi-Agents Holoniques nous avons proposé une première approche basée sur le concept de multiples vues. Chaque vue considère une perspective particulière du système et la composition de ces différentes vues se fait au travers de la holarchie finale. Nous avons également proposé une méthodologie, ASPECS, basée sur l'Ingénierie Dirigée par les Modèles et un processus plus complet qui couvre les phases d'analyse, de conception et d'implémentation.

7.2 Perspectives

Nos travaux ont été animés par la problématique de l'ingénierie de SMA. Cette problématique a été continuellement alimentée par des problèmes réels pour lesquels nous avons proposée des SMA comme solutions. Ces problèmes nous ont permis de fonder et de vérifier nos propositions. Par le traitement de nouvelles classes de problèmes, comme les réseaux de capteurs sans fils ou la simulation d'évolution de populations, nous envisageons de faire évoluer les méta-modèles organisationnels et les outils que nous avons définis.

Au delà des applications qui nous permettent de valider notre approche, nous avons comme perspective de faire évoluer le langage formel OZS présenté au chapitre 2 ainsi que les méthodologies présentées au chapitre 6.

Concernant le langage OZS, la définition de mécanismes de raffinement per-

mettrait la transformation de spécifications et éventuellement la génération de code ou de squelette de code. Ces mécanismes existent déjà pour des langages comme B (Abrial, 1996) et dans une moindre mesure Z (Spivey, 1992). L'adaptation de ces techniques semblent prometteuse. La possibilité d'intégration d'autres formalismes en ajout et/ou remplacement d'Object-Z et des statecharts mérite également d'être étudiée.
L'approche de capitalisation et réutilisation des connaissances que nous avons proposé mérite d'être testée au sein de contextes différents pour pouvoir être pleinement validée. L'aspect réutilisation des connaissances capitalisées est encore à un stade préliminaire mais constitue le sujet de la thèse d'Achraf Ben Miled.
Concernant les SMAH certains points comme les différents mode de gouvernement, les critères d'intégration de holons et l'application à d'autres classes d'applications nous semblent intéressants.
Nous avons conscience que les méthodologies que nous avons proposées manque de guide méthodologique et d'heuristiques pour pouvoir être pleinement utilisées. Sur ce point, de nouvelles expériences pourront permettre d'enrichir l'existant. Un outil graphique d'édition et de manipulation de modèles est indispensable aujourd'hui. Un tel outil doit permettre la sauvegarde et la manipulation de modèles. Associé à l'Ingénierie Dirigée par les Modèles et à des règles de transformation, il doit également permettre de rendre les modèles productifs et pérenne. C'est dans cet esprit que nous envisageons d'utiliser les méta-modèles définis et d'autres à venir comme celui du langage OZS pour permettre l'intégration complète de tous les concepts développés dans ce mémoire. Pour enrichir ASPECS, nous avons entamé des travaux concernant l'ingénierie des méthodes et en particulier les approches par fragments de méthodologies (Cossentino et al., 2008a).

Chapitre 8

Curriculum Vitae

Diplômes

- **1990**, Baccalauréat C, mention AB, lycée de Pézenas (34), option informatique
- **1992**, DUT informatique, Université Montpellier II
- **1993**, DEUG Mathématiques Appliquées et Sciences Sociales, Université Montpellier III
- IUP Génie Mathématiques et Informatique, Université Montpellier II :
  - **1994**, Licence mention B,
  - **1995**, Maîtrise mention AB.
- **1996**, DEA Informatique, "Génération efficace des concepts d'une relation binaire : exemples et contre-exemples", mention B, Laboratoire d'Informatique de Robotique et de Micro-électronique de Montpellier,
- **2000**, Doctorat en Informatique de l'université de Franche-Comté, spécialité Automatique et Informatique. "Vers une approche de spécification, de prototypage et de vérification de Systèmes Multi-Agents", soutenue le 26/01/2000, mention très honorable, Laboratoire de Recherche en Informatique de Sévenans (LaRIS).

Expériences professionnelles

- Enseignant vacataire à l'IPSé 1996-1999.
- $\frac{1}{2}$ ATER à l'UTBM année universitaire 1999-2000.
- Maître de Conférences à l'UTBM à partir du 1/09/2000.

Activités de recherche

- 2 chapitres dans des ouvrages collectifs
- 12 articles dans des revues internationales avec comités de lecture et actes
- 45 conférences internationales avec comités de lecture et actes
- 1 conférence nationale avec comité de lecture et actes
- Encadrement de 5 Master recherche, 5 Master étranger, et 1 ingénieur CNAM
- Co-encadrement de 4 thèses dont 2 déjà soutenues
- Titulaire de la PEDR depuis septembre 2007.
- Membre du comité de programme de 11 conférences
- Reviewer pour 2 revues

8.1 Thématique de la thèse

Le travail de thèse s'inscrit dans le cadre d'une démarche méthodique pour la construction de spécifications formelles de Systèmes Multi-Agents. L'idée de base est de définir de tels systèmes comme un ensemble d'entités mettant en œuvre des comportements génériques, des rôles, entre lesquels des interactions sont spécifiées. Ces rôles et ces interactions sont structurés en unités appelées organisations. Pour concevoir les spécifications déduites de cette démarche, un langage d'expression d'objets actifs et réactifs est proposé. Ce langage est basé sur l'utilisation de deux formalismes : Object-Z et statecharts. La composition de ces formalismes consiste en un ensemble de règles permettant leur intégration syntaxique et sémantique (Gruer et al., 2004). Ce langage multi-formalismes est caractérisé par :

- un pouvoir d'expression suffisant pour spécifier les différents aspects des SMA,
- des outils pour analyser une spécification,
- la possibilité de raffinements pour aboutir à une spécification proche d'une implémentation.

Notre démarche de spécification de Systèmes Multi-Agents s'inscrit dans un processus de prototypage et de vérification. Le prototypage réalisé grâce à l'animation des statecharts permet de valider la spécification par rapport au comportement attendu du système (Hilaire et al., 2000a). La vérification est rendu possible par l'expression de la sémantique du langage multi-formalismes avec des systèmes de transitions et à l'utilisation de ces systèmes de transi-

tions par des logiciels adéquats.
La spécification d'un Système Multi-Agents est basée sur un framework exprimé avec le langage multi-formalismes qui spécifie les concepts de rôle, interaction et d'organisation. Cette étude est concrétisée par la spécification de deux Systèmes Multi-Agents sur lesquelles nous avons mis en œuvre le prototypage et la vérification.

8.2 Activités de recherche après la thèse

Nos activités de recherche après la thèse s'articulent autour de quatres axes. Les deux premiers prolongent le travail initié lors de la thèse quant à la spécification des SMA et l'extension du méta-modèle RIO. Les deux derniers axes de recherche, entièrement nouveau, portent sur les systèmes holoniques et la capitalisation des connaissances.

8.2.1 Langage OZS

Le premier axe porte sur la notation formelle OZS. Nous avons approfondi la syntaxe et la sémantique de la notation et proposé des outils et techniques pour faciliter la vérification par évaluation sur un modèle et la preuve automatique de théorème. Parmi ces techniques, nous avons proposé l'utilisation du concept « Capacity » afin de définir une relation d'abstraction entre spécifications ce qui permet de réduire la complexité algorithmique de la vérification. Nous avons également étudié la combinaison des techniques de vérification par évaluation sur un modèle et de preuve par induction.

8.2.2 Dynamique des rôles

La relation entre les agents et les rôles mis en œuvre a été défini dans un premier temps de manière statique. Nous avons approfondi la sémantique de cette relation et définit des mécanismes basés sur le raffinement qui permettent d'obtenir une sémantique pour une relation dynamique (Hilaire et al., 2002c). Ces mécanismes permettent de valider et de vérifier des SMA dont les agents peuvent changer de rôles au cours de leurs existences (Hilaire et al., 2007).

8.2.3 Modélisation des systèmes holoniques

Le concept « Capacity » introduit dans ce but permet de modéliser le savoir-faire d'un rôle ou d'une organisation (Rodriguez et al., 2007a). Ce concept permet d'introduire une indirection entre une description abstraite d'un savoir faire et ses possibles réalisations concrètes. Ce méta-modèle est au coeur de notre approche de modélisation des organisations complexes basée sur le paradigme holonique. Ce paradigme introduit la notion de holon, entité qui peut être composée de holons comme sous-structures. La structure hiérarchique ainsi définie s'appelle holarchie. Nous avons défini un framework organisationnel pour l'ingénierie de systèmes holoniques en modélisant un holon comme un agent pouvant contenir d'autres agents (Rodriguez et al., 2003; Rodriguez et al., 2006a; Rodriguez et al., 2007b; Rodriguez et al., 2007c).
Concernant l'aspect méthodologique nous avons défini une méthodologie. Cette méthodologie est dédiée à l'ingénierie de SMA Holoniques. Elle aborde la modélisation en décomposant le système en cours d'étude selon plusieurs vues qui sont recomposées par la suite (Rodriguez et al., 2007c). Chaque vue est associée à une holarchie. Le différentes activités de cette méthodologie sont clairement définies et ce travail est issu d'une collaboration avec M Cossentino (chercheur à l'université de Palerme, Italie). Le processus est complet et part des besoins initiaux et aboutit à l'implémentation du système (Cossentino et al., 2007b; Cossentino et al., 2007a).

8.2.4 Capitalisation des connaissances

La capitalisation des connaissances vise à sauvegarder les connaissances acquises et détenues par les collaborateurs dans la pratique quotidienne de leur activité, principalement les savoir-faire et les retours d'expérience. Cette problématique prend une importance capitale dans un contexte de plus en plus concurrentiel pour les entreprises. Nous avons défini un méta-modèle pour modéliser un processus de conception de produits (Monticolo et al., 2006b). Ce méta-modèle permet de définir un contexte aux acteurs humains qui participent au processus de conception et ce faisant permet la capitalisation des connaissances issues des interactions entre acteurs humains. Ce méta-modèle prend en compte les connaissances et les compétences des acteurs métiers. Une cartographie des connaissances peut être générée à partir de ce méta-modèle . Pour faciliter la capitalisation dite « au fil de l'eau » un SMA a été conçu (Monticolo et al., 2007a). Intégré dans une plateforme de travail collaboratif, ce système permet de monitorer les actions des acteurs humains et de capitaliser les connaissances selon le méta-modèle défini. Ces connaissances sont structurées dans une mémoire de projet à l'aide d'une on-

tologie et peuvent être ainsi exploitées a posteriori (Monticolo et al., 2007c).

8.3 Perspectives de recherche

Les perspectives de recherche concernent la poursuite des travaux menés en ingénierie des connaissances afin de proposer une approche de réutilisation pro-active des connaissances. L'idée est de proposer de l'aide sous forme de connaissances stockées dans la mémoire de projet aux acteurs humains impliqués dans le processus de conception.
Afin de valider nos modèles et méthodologies nous menons des études dans différents domaines d'application avec des problématiques différentes. Par exemple, la simulation multi-niveaux ou les réseaux de capteurs sans fils. Ces domaines d'applications permettent la validation et/ou l'enrichissement de nos modèles et méthodologies.
Nos recherches se basent sur des concepts exprimés sous forme de modèles et méta-modèles. L'ingénierie Dirigé par les Modèles est un domaine de recherche récent qui prône l'utilisation de la méta-modélisation et des transformations de modèles afin de rendre les modèles pérennes et productifs. Nous comptons utiliser ces théories afin d'automatiser les transformations entre concepts semi-formels et formels ainsi que pour aider au raffinement entre les différentes étapes des méthodologies.
Sur le plan de la coopération internationale, nous souhaitons consolider nos travaux de recherche avec Massimo Cossentino, chargé de recherche au CNR (Centre National de la Recherche) de Palerme et Sebastian Rodriguez, chercheur à l'Université Nationale de Tucuman.

8.4 Insertion dans l'équipe de recherche

8.4.1 Animation d'équipe

Au sein de l'équipe informatique du SeT, j'assume depuis 2002, la responsabilité et l'animation de l'axe **Méthodologies de spécification et de vérification de Systèmes Multi-Agents**. Cet axe regroupe depuis septembre 2002 trois maîtres de conférences et un professeur des universités. C'est dans ce cadre que j'ai assumé le rôle de correspondant du réseau européen d'excellence AgentLink (cf point 1 section 8.2). L'activité de recherche de l'axe consiste à proposer une approche de : spécification formelle, de validation et de vérification pour l'analyse et la conception de Systèmes Multi-Agents et de Systèmes Holoniques. Pour cela nous introduisons un cadre

méthodologique fondé sur les concepts de capacité, de rôle, d'interaction et d'organisation. La modélisation s'effectue en considérant deux niveaux d'abstraction. Le premier (niveau organisationnel) considère le système comme un ensemble d'organisations, chacune étant composée d'entités abstraites appelées rôles et de leurs interactions. Une organisation représente le système selon un point de vue donné, relativement à un objectif de modélisation.
Le second niveau, qualifié d'agentification, introduit les agents comme des entités pouvant encapsuler des rôles. Les interactions entre les rôles deviennent alors des interactions entre les agents qui mettent en œuvre ces rôles.
D'un point de vue spécification formelle, nous proposons une approche multi-formalismes fondée sur la composition de spécifications exprimées en Object-Z et statecharts. Chaque concept du modèle est spécifié par une classe Object-Z pouvant intégrer un statechart. On a ainsi la possibilité, d'une part, de définir de nouveaux concepts au niveau modèle, d'autre part, d'hériter de ces classes pour définir ses propres capacités, rôles, interactions et organisations.
La validation repose sur le prototypage et la simulation à l'aide de l'environnement Statemate. Cet environnement permet d'exécuter et d'animer les spécifications statecharts aussi bien au niveau organisation qu'au niveau agentification. Concernant la vérification, nous avons proposé une approche fondée sur la définition d'une sémantique commune aux deux formalismes de spécification. Cette approche permet la vérification par évaluation sur un modèle (model checking) et la preuve de théorèmes grâce à la transformation des spécifications partielles (exprimées en Object-Z et statecharts) en systèmes de transition.
Cette approche nous a permis d'analyser et de concevoir des Systèmes Multi-Agents pour des domaines d'applications variés : optimisation et simulation de problèmes liés au transport, architecture de contrôle intelligent de robots, animation de personnages virtuels et optimisation de problèmes liés aux réseaux radiomobiles.
Depuis 2003 nous travaillons sur l'ingénierie d'architectures d'agents à base de systèmes immunitaires et de holons. Nous travaillons également à la capitalisation des connaissances à base d'agents dans ce cadre, j'assume la responsabilité d'un projet de recherche inter-universités de technologies, soutenu par les conseils scientifiques de l'UTBM et de l'UTT (Université de Technologie de Troyes), cf point 4 section 8.2.

8.4.2 Participation à des projets

1. **Correspondant du réseau d'excellence européen AgentLink**
   Correspondant de AgentLink II et III depuis septembre 2000 jusqu'à 2006. Membre des groupes de travail Self-Organisation in Multi-Agent

Systems (publication d'un article dans la revue Informatica en 2006 avec les membres du groupe (Bernon et al., 2006)) et Agent Oriented Software Engineering.

2. **Coordinateur du groupe de travail "Système Daedalus"**
   Ce groupe a réalisé le système multi-agents Daedalus pour l'animation de personnages virtuels dans la pièce de théâtre Orgia. Cette pièce a fait deux tournée en France 2000-2001 et 2001-2002.
   Publications liées au projet : (Hilaire et al., 2002b)

3. **Coordinateur du projet Robotec**
   Coordinateur, 2001-2002, et membre du projet robotec qui vise la mise en place d'une équipe de robots mobiles. Participation à la coupe du monde de robots footballeurs 2002 organisée par la FIRA en Corée du sud (résultat 15/32). Participation à la coupe d'europe à Vienne en 2003 (résultat 9/20)
   Publications liées au projet : (Hilaire et al., 2002a; Bakhouya et al., 2003; Tolba et al., 2003)

4. **Coordinateur UTBM du projet inter-UT Trace**
   Projet validés par les conseils scientifiques de l'UTT et de l'UTBM pour la période 2007-2010. Ce projet commun aux laboratoires Tech-Cico de l'UTT et SeT de l'UTBM vise à approfondir les collaborations en terme de capitalisation des connaissances et d'annotation de mémoire de projets de conception.
   Publications liées au projet : (Gomes et al., 2007; Djaiz et al., 2006a; Djaiz et al., 2006b)

8.4.3 Organisation de conférence et évaluations d'articles

Co-organisateur

- **Organisation de conférences**
  - de la journée technique Transport urbain et multimodalité en 2001,
  - du workshop ACM/IEEE Agent supported Cooperative Work (ACW) en 2007,
  - du workshop Self-Adaptation for Robustness and Cooperation in Holonic MultiAgent Systems (SARC) en 2008.

Membre des comités de programme
- AgentDay 2002,
- Journée Technique "Ingénierie des Systèmes de Mobilité et de Transport" en 2004,
- workshop Capitalisation et Réutilisation des Connaissances Métier en conception de systèmes mécaniques (C2EI) en 2007,
- workshop AOse Methodologies and Processes en 2008,
- International Conference on Autonomous Robots and Agents 2009.

- **Revue d'articles**

  - Journées Francophones Intelligence Artificielle Distribuée et Système Multi-Agents 2002-2008,
  - International Conference on Information Systems and Engineering (ISE) 2002-2003,
  - IEEE International Symposium on Signal Processing and Information technology (ISSPIT) 2002,
  - Genetic and Evolutionnary Computation COnference 2004.
  - International Journalon Software Engineering and Knowledge Engineering.
  - International Journal on Agent Oriented Software Engineering.
  - ACM Transactions on Autonomous and Adaptive Systems.

8.5 Encadrements et co-encadrements

8.5.1 Thèse

- A Ben Miled « Vers la réutilisation des connaissances en ingénierie de conception », pourcentage d'encadrement 50% Directeur : A. Koukam. 1ère année de thèse.

- Y. Yao « Holonic Models for Wireless Sensor Networks », pourcentage d'encadrement 40% Directeur : A. Koukam. 2ème année de thèse.

- D. Monticolo « Approche multi-agents pour la capitalisation des connaissances du processus de conception de produits », pourcentage d'encadrement : 40 % Directeur : A. Koukam, thèse soutenue le 26/02/2008.

- S. Rodriguez « From Analysis to Design of Holonic Multi-Agent Systems : a Framework, Methodological Guidelines and Applications » pourcentage d'encadrement : 60 %. Directeur : A. Koukam, thèse sou-

tenue le 14/12/05.

8.5.2 Master

- R Basset « Approche Multi-Agents pour l'analyse et la conception d'un collecticiel », Master ISC de l'Université de Haute-Alsace, 2006, pourcentage d'encadrement : 100 % .
- J Jolly « Vers l'apprentissage de comportements adverses en robotique collective ». DEA IAP de Besançon, 2004, pourcentage d'encadrement : 40 % .
- F Tolba, « Architecture réactive pour la supervision d'une équipe de robots footballeurs ». DEA IAP de Besançon, 2003, pourcentage d'encadrement : 50 % .
- S. Rodriguez, "Approches multi-agent en robotique collective, application aux robots footballeurs". DEA IAP de Besançon, 2002, pourcentage d'encadrement : 50 % .
- L. Charara, Spécification des systèmes multi-agents fondée sur le langage Object-Z et les statecharts. DEA IAP de Besançon, 2000, pourcentage d'encadrement : 50 % .

8.5.3 Masters de recherche délivrés par des universités étrangères

Ces stages ont été effectués au sein de notre laboratoire en accord avec l'université mentionnée

- Luis Rosa Soler, An Immune-System based Multiagent System for Solving the Transport on Demand problem. Master de l'université de Tucuman, Argentine, 2004, pourcentage d'encadrement : 100 %.
- Lukasz Nowak, Coordinating behavior-based intelligent mobile robot soccer using an immunized reinforcement adaptive learning mechanism in Multi Agent environment. Master, de l'AGH, Pologne, 2004, pourcentage d'encadrement : 100 %
- Tomacz Rodecky, Développement de réseaux temporels simples pour système multi-robots, Master de l'AGH, Pologne, 2002, pourcentage d'encadrement : 100 %
- R. Campero, Sur la transformation des Statecharts en Systèmes de Transition (en co-encadrement à 50 % avec P. Gruer), 2002. Master de l'Université Nationale de Tucuman (UNT), Argentine.

8.5.4 Ingénieur CNAM

– P. Descamp, Environnement de modélisation et de simulation de réseaux de transport urbain de passagers, travail soutenu le 8 mars 2002

Je suis titulaire d'une **prime d'encadrement doctoral et de recherche** depuis le dernier trimestre 2007.

8.6 Enseignements et tâches administratives

8.6.1 Enseignements

J'ai participé à l'enseignement de plusieurs unités de valeurs certaines sont sous ma responsabilité directe.

– *LO11 : Algorithmique et programmation niveau I*
Cette UV donne les éléments de base en algorithmique et en programmation. Le cours introduit progressivement les différents concepts Pascal : structure générale, types scalaires prédéfinis, instruction d'affectation, entrées/sorties conversationnelles, structure de choix, structure de répétition, types scalaires définis par l'utilisateur, type chaîne, type tableau.
Intervention de 1996 à 1998, 40h TP par an.
– *LO21 : Algorithmique et programmation niveau II*
Le but de cette UV est de présenter les notions fondamentales de la programmation. Après un rappel sur les structures de données statiques, cette UV aborde les structures de données dynamiques, les listes, les fichiers, récursivité et les algorithmes associés.
Intervention de 1997 à 2002, 48h TD et 40h TP par an.
– *LO43 : Bases fondamentales de la programmation orientée objet* Cette UV propose une approche complète des concepts de la programmation orientée objet : notion d'encapsulation, instanciation, héritage. Elle présente également les bases théoriques de ces concepts (types abstraits).
Intervention en 2000 et 2007, 26h TD.
– *LO52 : Systèmes embarqués et informatique mobile* Les objectifs de cette UV sont : présenter les spécificités des systèmes embarqués et les besoins logiciels et de communication qui en résultent et l'étude des différentes solutions technologiques dédiées à l'embarqué.
Intervention de 2005 à 2007, 12h TD par an.

- *IA52 : Systèmes à base de connaissances*
  Le but de cette UV est de donner les éléments nécessaires pour la conception, le développement et l'exploitation des systèmes à base de connaissances.
  Intervention de 1998 à 2002, 6h CM, 8h TD et 36h TP par an.
- *IA54 : Systèmes Multi-Agens et Résolution Distribuée de Problèmes*
  Cette UV présente le paradigme des systèmes multi-agents et les principales techniques existantes de coordination et de coopération entre agents intelligents.
  Responsable depuis 2002, 12h CM, 20h TD par an
- *GL51 : Processus et Qualité du Logiciel* Cette UV a pour objectifs l'étude des principes et des procédures d'assurance qualité, tout en les intégrant dans le suivi et l'organisation des projets informatiques.
  Responsable depuis 2003, 6h CM, 20h TD et 30h TP par an.
- *GL52 : Génie Logiciel*
  Cette UV a pour but l'étude des principes, techniques et méthodes de développement des logiciels. Elle précise les objectifs du génie logiciel, ses principes fondamentaux, et présente les principaux outils de spécification, de conception et de programmation.
  Intervenant depuis 1999, 2h de CM, 48h TD, 32h de TP par an .

8.6.2 Responsabilités administratives

- Responsable de la filière d'ingénieurs : Ingénierie des Logiciels et de la Connaissance depuis février 2003, cycle des trois derniers semestres du parcours d'ingénieur UTBM spécialisés en ingénierie de systèmes d'informations et systèmes intelligents.
- Responsable des unités de valeurs : IA54 « Systèmes Multi-Agens et Résolution Distribuée de Problèmes » et GL51 « Processus et Qualité du Logiciel »
- Membre du bureau de département Génie Informatique de 2000 à 2003,
- Membre nommé du comité d'évaluation des projets « nouvelles technologies éducatives » de l'UTBM,
- Responsable des ressources en informatique pour la bibliothèque de l'Université de Technologie de Belfort Montbéliard,
- Membre élu du conseil de discipline de l'UTBM.

8.7 Liste des publications

8.7.1 Chapitre d'ouvrage

Gruer, P., Hilaire, V., and Koukam, A. (2002). *approche multi-formalismes pour la spécification des systèmes multi-agents*, volume Organisation et applications des SMA, chapter 1. Hermès.

Gruer, P., Hilaire, V., Kozlak, J., and Koukam, A. (2003). *A multi-agent approach to modeling and simulation of transport on demand problem*, volume Artificial intelligence and security in computing systems, pages 119–126. Kluwer Academic Publishers.

8.7.2 Revue

Bernon, C., Chevrier, V., Hilaire, V., and Marrow, P. (2006). Applications of self-organising multi-agent systems : An initial framework for comparison. *Informatica*, 30(1) :73–84.

Gomes, S., Monticolo, D., Hilaire, V., and Eynard, B. (2007). Content management based on multi agent systems for collaborative design. *International Journal of Product Development*, 2(4) :317–336.

Gruer, P., Hilaire, V., and Koukam, A. (2004). Heterogeneous formal specification based on object-z and state charts : semantics and verification. *Journal of Systems and Software*, 70(1-2) :95–105.

Gruer, P., Hilaire, V., Koukam, A., and Cetnarowicz, K. (2002). A formal framework for multi-agent systems analysis and design. *Expert Systems with Applications*, 23.

Hilaire, V., Gruer, P., Koukam, A., and Simonin, O. (2007). Formal specification approach of role dynamics in agent organisations : Application to the satisfaction-altruism model. *International Journal of Software Engineering and Knowledge Engineering*, 17(5) :615–641.

Hilaire, V., Gruer, P., Koukam, A., and Simonin, O. (2008a). Formal driven prototyping approach for multi-agent systems. *International Journal of Agent Oriented Software Engineering*, 2(2) :246–266.

Hilaire, V., Koukam, A., and Rodriguez, S. (2008b). An adaptative agent architecture for holonic multi-agent systems. *ACM Transactions on Autonomous and Adaptive Systems*, 3(1) :1–24.

Koukam, A., Mazigh, B., Gruer, P., and Hilaire, V. (2003). A multiview approach to modeling and analysis of discrete event systems. *System Analysis - Modelling - Simulation*, 43(6) :721–740. International Journal.

Koźlak, J., Créput, J.-C., Hilaire, V., and Koukam, A. (2006). Multi-agent approach to dynamic pick-up and delivery problem with uncertain knowledge about future transport demands. *Fundamenta Informaticae*, 71(1).

Monticolo, D., Hilaire, V., Gomes, S., and Koukam, A. (2008). A multi-agent system for building project memories to facilitate design process. *Integrated Computer-Aided Engineering*, 15(1) :3–20.

Rodriguez, S., Hilaire, V., Gruer, P., and Koukam, A. (2007a). A formal holonic framework with proved self-organizing capabilities. *International Journal of Cooperative Information Systems*, 16(1) :7–25.

Rodriguez, S., Hilaire, V., and Koukam, A. (2007b). Towards a holonic multiple aspect analysis and modeling approach for complex systems : Application to the simulation of industrial plants. *Simulation Modelling Practice and Theory*, 15(5) :521–543.

8.7.3 Communications internationales avec comité de lecture

Bakhouya, M., Rodriguez, S., Hilaire, V., Koukam, A., and Gaber, J. (2003). Intelligent immune-based system for autonomous soccer robots. In *FIRA Robot World Congress Austria*.

Berdai, A., Gruer, P., Hilaire, V., and Koukam, A. (2002). A multi-agent model for the estimation of passenger waiting time in public transportation networks. In *2nd WSEAS Int. Conf. on Simulation, Modelling and Optimization (ICOSMO 2002)*.

Campero, R., Gruer, P., Hilaire, V., and Rovarini, P. (2000). Modeling and simulation of agent-oriented systems : an approach based on object-z and the statecharts. In Urban, C., editor, *Agent Based Simulation*.

Cetnarowicz, K., Gruer, P., Hilaire, V., and Koukam, A. (2001). A formal specification of m-agent architecture. In *2nd International Workshop of Central and Eastern Europe on Multi-Agent Systems*, Lecture Notes in Artificial Intelligence. Springer Verlag.

Cossentino, M., Gaglio, S., Galland, S., Gaud, N., Hilaire, V., abderrafiaa KOUKAM, and Seidita, V. (2008). A mas metamodel-driven approach to process composition. In *AOSE*.

Cossentino, M., Gaud, N., Galland, S., Hilaire, V., and Koukam, A. (2007a). A holonic metamodel for agent-oriented analysis and design. In *HoloMAS'07*.

Cossentino, M., Nicolas Gaud, S. G., Hilaire, V., and Koukam, A. (2007b). An holonic metamodel for agent-oriented analysis and design. In *EUMAS*.

Delclaux, F. and Hilaire, V. (1995). Premiers essais de détermination du coefficient de ruissellement decennal a l'aide de regles floues. In *L'hydrologie tropicale : géoscience et outil pour le developpement*, number 238, pages 413–424. International Association of Hydrological Science Publ.

Gaud, N., Galland, S., Hilaire, V., and Koukam, A. (2008a). An organisational platform for holonic and multiagent systems. In *PROMAS*.

Gaud, N., Hilaire, V., Galland, S., Koukam, A., and Cossentino, M. (2008b). A verification by abstraction framework for organizational multi-agent systems. In *AT2AI*.

Gruer, P., Hilaire, V., and Koukam, A. (2000a). an Approach to the Verification of Multi-Agent Systems. In *International Conference on Multi Agent Systems*. IEEE Computer Society Press.

Gruer, P., Hilaire, V., and Koukam, A. (2000b). Verification of Object-Z Specifications by using Transition Systems. In Maibaum, T. S. E., editor, *Fundamental Aspects of Software Engineering*, number 1783 in Lecture Notes in Computer Science. Springer Verlag.

Gruer, P., Hilaire, V., and Koukam, A. (2001a). Multi-agent approach to modeling and simulation of urban transportation systems. In *IEEE Systems, Man, and Cybernetics Conference*.

Gruer, P., Hilaire, V., Koukam, A., and Cetnarowicz, K. (2001b). A formal framework for multi-agent systems analysis and design. In *Thirteenth International Conference on Software Engineering & Knowledge Engineering*.

Gruer, P., Hilaire, V., Koukam, A., and Hayat, S. (2001c). a methodology based on multiples views for multi-agent systems in simulation, application to the transportation domain. In *13th European Simulation Symposium*.

Gruer, P., Hilaire, V., Kozlak, J., and Koukam, A. (2002). Application of multi-agent simulations to solve a transport on demand problem. In *Ninth International Conference on Advanced Computer Systems*, Sczeczin.

Hilaire, V. (2000). Formal specification and prototyping of organizational based multi-agent systems. In *International Conference on the Design of Cooperative Systems*, Lecture Notes in Computer Science. Springer Verlag.

Hilaire, V., Gruer, P., Koukam, A., and Moudni, A. E. (2002a). Engineering soccer robots behaviours. In *2002 FIRA Robot Congress*, Seoul. KAIST Press.

Hilaire, V., Koukam, A., and Gruer, P. (2002b). a multi-agent system featuring virtual character in play for theatre. In *Proceedings of the Agent-Day'02 workshop*.

Hilaire, V., Koukam, A., and Gruer, P. (2002c). A mechanism for dynamic role playing. In *Agent Technologies, Infrastructures, Tools and Applications for E-Services*, number 2592 in Lecture Notes in Artificial Intelligence. Springer Verlag.

Hilaire, V., Koukam, A., Gruer, P., and Müller, J.-P. (2001). Formal specification and prototyping of multi-agent systems. In Omicini, A., Tolksdorf, R., and Zambonelli, F., editors, *Engineering Societies in the Agents' World*, number 1972 in Lecture Notes in Artificial Intelligence. Springer Verlag.

Hilaire, V., Lissajoux, T., and Koukam, A. (1998). AgentCharts : An Operational Model For Multi-Agent Systems. In *International Conference on Advanced Computer Systems ACS'98*.

Hilaire, V., Lissajoux, T., and Koukam, A. (1999). Towards an executable specification of multi-agent systems. In Filipe, J. and Cordeiro, J., editors, *International Conference on Enterprise Information Systems'99*. Kluwer Academic Publisher.

Hilaire, V., Lissajoux, T., Koukam, A., and Creput, J. (2000). A multi-agent approach to adaptive mesh generation. In *Agent Based Simulation*.

Hilaire, V., Simonin, O., Koukam, A., and Ferber, J. (2005). A formal framework to design and reuse agent and multiagent models. In Odell, J., Giorgini, P., and Muller, J., editors, *Agent Oriented Software Engineering*, number 3382 in LNCS 3382. Springer.

Kozlak, J., Creput, J.-C., Hilaire, V., and Koukam, A. (2004). Multi-agent environment for dynamic transport planning and scheduling. In Bubak, M., van Albada, G. D., and Peter M. A. Sloot, e. a., editors, *Computational Science*, number 3038 in Lecture Notes in Computer Science, pages 638 – 645. Springer Verlag.

Lissajoux, T., Hilaire, V., Koukam, A., and Caminada, A. (1998). Genetic Algorithms as Prototyping Tools for Multi-Agent Systems : Application to the Antenna Parameter Setting Problem. In Albayrak, S. and Garijo, F. J., editors, *Lecture Notes in Artificial Intelligence*, number 1437 in LNAI. Springer Verlag.

Mazigh, B., Hilaire, V., Gruer, P., and Koukam, A. (1999a). Agent oriented approach for modeling and simulation of productive organizations. In *10th DAAAM International Symposium ¡¡International Manufacturing & Automation : Past-Present-Future¿¿*. DAAAM.

Mazigh, B., Koukam, A., Hilaire, V., and Gruer, P. (1999b). Combining multi-agents approach with high level petri nets for simulation of productive organizations. In *IFAC Workshop on Multi Agent Systems in Production*. IFAC.

Monticolo, D., Gomes, S., Hilaire, V., and Serrafero, P. (2006a). A multi-agent architecture to synthesize industrial knowledge from a plm system. In *PLEDM'06*.

Monticolo, D., Hilaire, V., Koukam, A., and Gomes, S. (2007a). An e-groupware based on multi agents systems for knowledge management. In *DEST'07*.

Monticolo, D., Hilaire, V., Koukam, A., and Gomes, S. (2007b). Ontodesign ; a domain ontology for building and exploiting project memories in mechanical design projects. In *Knowledge Management in Organizations*.

Monticolo, D., Hilaire, V., Koukam, A., and Meunier, S. (2006b). An approach for building project memories to facilitate design process in a concurrent engineering context. In *proceedings of Concurrent Engineering'06*.

Monticolo, D., Hilaire, V., Serrafero, P., and Gomes, S. (2007c). Knowledge capitalization process linked to the design process. In *KMOM'07*.

Rodriguez, S., Gaud, N., Hilaire, V., Galland, S., and Koukam, A. (2007). An analysis and design concept for self-organization in holonic mas. In Brueckner, S., Hassas, S., Jelasity, M., and Yamins, D., editors, *Engineering Self-Organising Systems*, number 4335 in LNAI, pages 15–27.

Rodriguez, S., Gaud, N., Hilaire, V., and Koukam, A. (2006a). Modeling holonic systems with an organizational approach. In *proceedings of EUMAS'06*.

Rodriguez, S. and Hilaire, V. (2002). An architecture for multi-agent systems in the robot-soccer field. In *Proceedings of the AgentDay'02 workshop*.

Rodriguez, S., Hilaire, V., and Koukam, A. (2003). Towards a methodological framework for holonic multi-agent systems. In *Proceedings of the ESAW'03 workshop*, pages 179–185.

Rodriguez, S., Hilaire, V., and Koukam, A. (2005a). Formal specification of holonic multi-agent systems framework. In Sunderam, V. S., van Albada, G. D., Sllot, P. M. A., and Dongarra, J. J., editors, *Computational Science - ICCS 2005*, number 3516 in Lecture Notes in Computer Science, pages 719 – 726. Springer.

Rodriguez, S., Hilaire, V., and Koukam, A. (2005b). Multi-agent coordination is arbitration from a super-holon point of view. In *proceedings of MA4CS'05*.

Rodriguez, S., Hilaire, V., and Koukam, A. (2006b). An holonic approach to model and deploy large scale simulations. In *proceedings of MABS'06*.

Rodriguez, S., Hilaire, V., and Koukam, A. (2006c). Holonic modeling of environments for situated multi-agent systems. In Weyns, D., Parunak, H. V. D., and Michel, F., editors, *Environments for Multi-Agent Systems II*, number 3830 in Lecture Notes in Artificial Intelligence, pages 18–31. Springer.

Tolba, F., Simonin, O., and Hilaire, V. (2003). A reactive macro control for mirosot robot soccers. In *FIRA Robot World Congress Austria*.

Ye, Y., Wandong, C., Hilaire, V., and Koukam, A. (2008a). A holonic model in wireless sensor networks. In *The Fourth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*.

Ye, Y., Wandong, C., Hilaire, V., and Koukam, A. (2008b). Location scheme in wireless sensor networks. In *The Fourth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*.

8.7.4 Communications nationales avec comité de lecture

Hilaire, V., Koukam, A., Gruer, P., and Müller, J.-P. (2000). Vers une méthodologie formelle de spécification de systèmes multi-agents. In *Journées Francophones IAD & SMA*. Hermes.

Bibliographie

Abecker, A., Bernardi, A., and van Elst, L. (2003). Agent technology for distributed organizational memories : The frodo project. In *ICEIS (2)*, pages 3–10.

Abrial, J.-R. (1996). *The B Book - Assigning Programs to Meanings*. Cambridge University Press.

Adam, E., Mandiau, R., and Kolski, C. (2002). *Une Méthode de modelisation et de conception d'organizations Multi-Agents holoniques*, chapter 2, pages 41–75. Hermes.

Ainsworth, M., Cruickshank, A. H., Wallis, P. J. L., and Groves, L. J. (1994). Viewpoint specification and Z. *Information and Software Technology*, 36(1) :43–51.

Bakhouya, M., Rodriguez, S., Hilaire, V., Koukam, A., and Gaber, J. (2003). Intelligent immune-based system for autonomous soccer robots. In *FIRA Robot World Congress Austria*.

Bergenti, F., Gleizes, M.-P., and Zambonelli, F., editors (2004). *Methodologies and Software Engineering for Agent Sytems* . Kluwer Academic Press.

Bergenti, F. and Poggi, A. (2000). Exploiting uml in the design of multi-agent systems. In Omicini, A., Tolksdorf, R., and Zambonelli, F., editors, *Engineering Societies in the Agents' World*, Lecture Notes in Artificial Intelligence. Springer Verlag.

Bernon, C., Camps, V., Gleizes, M., and Picard, G. (2005a). Engineering adaptive multi-agent systems : The adelfe methodology. In Henderson-Sellers, B. and Giorgini, P., editors, *Agent-Oriented Methodologies*, chapter VII, pages 172–202. Idea Group publishing.

Bernon, C., Chevrier, V., Hilaire, V., and Marrow, P. (2006). Applications of self-organising multi-agent systems : An initial framework for comparison. *Informatica*, 30(1) :73–84.

Brazier, F., Keplicz, B. D., Jennings, N., and Treur, J. (1997). Desire : Modelling multi-agent systems in a compositional formal framework. *International Journal of Cooperative Information Systems*, 6 :67–94.

Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J., and Perini, A. (2004a). Tropos : An agent-oriented software development methodology. *Journal of Autonomous Agents and Multi-Agent Systems*, 8(3) :203–236.

Brooks, R. and Connell, J. H. (1986). Asynchronous distributed control systems for a mobile robot. *SPIE*, 727. Mobile Robots.

Brussel, H. V., Wyns, J., Valckenaers, P., Bongaerts, L., and Peeters, P. (1998). Reference architecture for holonic manufacturing systems : Prosa.

Bürckert, H.-J., Fischer, K., and Vierke, G. (2000). Holonic transport scheduling with teletruck. *Applied Artificial Intelligence*, 14(7) :697–725.

Campero, R., Gruer, P., Hilaire, V., and Rovarini, P. (2000). Modeling and simulation of agent-oriented systems : an approach based on object-z and the statecharts. In Urban, C., editor, *Agent Based Simulation*.

Champin, P.-A., Prié, Y., and Mille, A. (2004). MUSETTE : a framework for knowledge capture from experience. In Hébrail, G., Lebart, L., and Petit, J.-M., editors, *EGC*, volume RNTI-E-2 of *Revue des Nouvelles Technologies de l'Information*, pages 129–134.

Chapelle, J., Simonin, O., and Ferber, J. (2002). How situated agents can learn to cooperate by monitoring their neighbors' satisfaction. In *15th European Conference on Artificial Intelligence*.

Clarke, E. M., Grumberg, O., and Long, D. E. (1992). Model checking and abstraction. In *POPL*, pages 342–354.

Conklin, J. and Begeman, M. L. (1988). gIBIS : A hypertext tool for exploratory policy discussion. *ACM Transactions on Office Information Systems*, 6(4) :303–331. Selected Papers from the Conference on Computer-Supported Cooperative Work (CSCW '88).

Cossentino, M. (2005). *From Requirements to Code with the PASSI Methodology*. Idea Group Inc., Hershey, PA, USA.

Cossentino, M., Gaglio, S., Galland, S., Gaud, N., Hilaire, V., abderrafiaa KOUKAM, and Seidita, V. (2008a). A mas metamodel-driven approach to process composition. In *AOSE*.

Cossentino, M., Galland, S., Gaud, N., Hilaire, V., and Koukam, A. (2008b). How to control emergence of behaviours in a holarchy. In *Self-Adaptation for Robustness and Cooperation in Holonic Multi-Agent Systems*.

Cossentino, M., Gaud, N., Galland, S., Hilaire, V., and Koukam, A. (2007a). A holonic metamodel for agent-oriented analysis and design. In *Holo-MAS'07*.

Cossentino, M., Nicolas Gaud, S. G., Hilaire, V., and Koukam, A. (2007b). An holonic metamodel for agent-oriented analysis and design. In *EUMAS*.

Coutinho, L. R., Sichman, J. S., and Boissier, O. (2005). Modeling organization in mas : A comparison of models. In *First Workshop on Software Engineering for Agent-oriented Systems*.

de Moura, L., Owre, S., Rueß, H., Rushby, J., Shankar, N., Sorea, M., and Tiwari, A. (2004). SAL 2. In Alur, R. and Peled, D., editors, *Computer-Aided Verification, CAV 2004*, volume 3114 of *Lecture Notes in Computer Science*, pages 496–500, Boston, MA. Springer-Verlag.

DeLoach, S. (1999). Multiagent systems engineering : a methodology and language for designing agent systems. In *Agent Oriented Information Systems '99*.

Demazeau, Y. (1995). From interactions to collective behaviour in agent-based systems. In *European Conference on Cognitive Science*.

Derrick, J., Bowman, H., and Steen, M. (1995). Viewpoints and objects. In Bowen, J. P. and Hinchey, M. G., editors, *Ninth Annual Z User Workshop*, volume 967 of *Lecture Notes in Computer Science*, pages 449–468, Limerick. Springer-Verlag.

Dieng, R., Corby, O., Giboin, A., and Ribière, M. (1999). Methods and tools for corporate knowledge management. *Int. J. Hum.-Comput. Stud*, 51(3) :567–598.

Dignum, V., Vázquez-Salceda, J., and Dignum, F. (2004). OMNI : Introducing social structure, norms and ontologies into agent organizations. In *PROMAS*, volume 3346. Springer.

Dijkstra, E. W. (1982). On the role of scientific thought. In *Selected Writings on Computing : A Personal Perspective*, pages 60–66.

Djaiz, C., Monticolo, D., and Matta, N. (2006a). Capitalization of knowledge from projects. In *International Conference of Concurrent engineering (CE)*, pages 317–324.

Djaiz, C., Monticolo, D., and Matta, N. (2006b). Project memory decision making. In *International Conference of Knowledge, Information and Creativity Support Systems (KICSS)*.

Duke, R., King, P., Rose, G., and Smith, G. (1991). The Object-Z specification language. Technical report, Software Verification Research Center,

Departement of Computer Science, University of Queensland, AUSTRALIA.

Ermine, J. (2000). La gestion des connaissances, un levier stratégique pour les entreprises. In *IC ?00*.

Farmer, J. D., Packard, N. H., and Perelson, A. S. (1986). The immune system, adaption and machine learning. *Physica D*, 22 :187–204.

Ferber, J. (1997). Les systemes multi-agents : un apercu general. *Technique et science informatiques*, 16(8) :979–1012.

Ferber, J. and Gutknecht, O. (1998). A meta-model for the analysis and design of organizations in multi-agent systems. In Demazeau, Y., Durfee, E., and Jennings, N., editors, *ICMAS'98*.

Ferber, J. and Jacopin, E. (1991). The frameworks of eco-problem-solving. In Elsevier, editor, *Decentralized AI*, volume 2.

Fisher, M. (1994). A survey of Concurrent METATEM — the language and its applications. In Gabbay, D. M. and Ohlbach, H. J., editors, *Temporal Logic — Proceedings of the First International Conference (LNAI Volume 827)*, pages 480–505. Springer-Verlag : Heidelberg, Germany.

Fisher, M. and Wooldridge, M. (1993). Specifying and verifying distributed intelligent systems. In *Progress in AI – Proceedings of Portuguese Conference on Artificial Intelligence (EPIA)*. Springer Verlag.

Gandon, F., Berthelot, L., and Dieng-Kuntz, R. (2002a). A multi-agent platform for a corporate semantic web. In *AAMAS*, pages 1025–1032. ACM.

Gandon, F., Poggi, A., Rimassa, G., and Turci, P. (2002b). Multi-agent corporate memory management system. *Applied Artificial Intelligence*, 16(9-10) :699–720.

Gaud, N., Galland, S., Gechter, F., Hilaire, V., and Koukam, A. (2009). Holonic multilevel simulation of complex systems. application to real-time pedestrians simulation in virtual urban environment. *Simulation Modelling Practice and Theory*.

Gaud, N., Galland, S., Hilaire, V., and Koukam, A. (2008a). An organisational platform for holonic and multiagent systems. In *PROMAS*.

Gaud, N., Hilaire, V., Galland, S., Koukam, A., and Cossentino, M. (2008b). A verification by abstraction framework for organizational multi-agent systems. In *AT2AI*.

Georgé, J.-P., Gleizes, M.-P., and Glize, P. (2003). Conception de systèmes adaptatifs à fonctionnalité émergente : la théorie des AMAS. *Revue d'Intelligence Artificielle*, 17(4/2003) :591–626.

Gerber, C., Siekmann, J., and Vierke, G. (1999). Holonic multi-agent systems. Research Report RR-99-03, DFKI.

Gleizes, M.-P., Camps, V., Georgé, J.-P., and Capera, D. (2008). Engineering systems which generate emergent functionalities. In Weyns, D., Brueckner, S., and Demazeau, Y., editors, *Engineering Environment-Mediated Multiagent Systems - Satellite Conference held at The European Conference on Complex Systems (EEMMAS), Dresden, Germany, 01/10/07-05/10/07*, number 5049 in Lecture Notes in Artificial Intelligence (LNAI), page (on line), http ://www.springerlink.com/. Springer-Verlag.

Gomes, S., Monticolo, D., Hilaire, V., and Eynard, B. (2007). Content management based on multi agent systems for collaborative design. *International Journal of Product Development*, 2(4) :317–336.

Gomes, S., Sagot, J., Koukam, A., and Leroy, N. (1999a). Acsp, an intranet forum supporting a concurrent engineering design life cycle. In *6th European Concurrent Engineering Conference, ECEC'99*.

Gruer, P., Hilaire, V., and Koukam, A. (2000a). an Approach to the Verification of Multi-Agent Systems. In *International Conference on Multi Agent Systems*. IEEE Computer Society Press.

Gruer, P., Hilaire, V., and Koukam, A. (2000b). Verification of Object-Z Specifications by using Transition Systems. In Maibaum, T. S. E., editor, *Fundamental Aspects of Software Engineering*, number 1783 in Lecture Notes in Computer Science. Springer Verlag.

Gruer, P., Hilaire, V., and Koukam, A. (2001a). Multi-agent approach to modeling and simulation of urban transportation systems. In *IEEE Systems, Man, and Cybernetics Conference*.

Gruer, P., Hilaire, V., and Koukam, A. (2004). Heterogeneous formal specification based on object-z and state charts : semantics and verification. *Journal of Systems and Software*, 70(1-2) :95–105.

Gruer, P., Hilaire, V., Koukam, A., and Cetnarowicz, K. (2002b). A formal framework for multi-agent systems analysis and design. *Expert Systems with Applications*, 23.

Grundstein, M. (2000). *From capitalizing on Company Knowledge to Knowledge Management*, chapter 12, pages 261–287. The MIT Press.

Guizzardi, R. S. S., Aroyo, L., and Wagner, G. (2003). Agent-oriented knowledge management in learning environments : A peer-to-peer helpdesk case study. In (van Elst et al., 2004), pages 57–72.

Gutknecht, O. and Ferber, J. (2000a). Madkit : A generic multi-agent platform. In Sierra, C., Gini, M., and Rosenschein, J. S., editors, *Proceedings of the Fourth International Conference on Autonomous Agents*, pages 78–79, Barcelona, Catalonia, Spain. ACM Press.

Harel, D. (1988). On visual formalisms. *Communications of the ACM*, 31(5) :514–530.

Harel, D., Lachover, H., Naamad, A., Pnueli, A., Politi, M., Sherman, R., Shtull-Trauring, A., and Trakhtenbrot, M. B. (1990). Statemate : A working environment for the development of complex reactive systems. *IEEE Transactions on Software Engineering*, 16(4) :403–414.

Hassas, S. (2003). *Systèmes Complexes à base de Multi-Agents Situés*. PhD thesis, Université Claude Bernard-Lyon 1.

Henderson-Sellers, B. (2003). Method engineering for OO systems development. *Commun. ACM*, 46(10) :73–78.

Henderson-Sellers, B. and Giorgini, P., editors (2005). *Agent-Oriented Methodologies*. Idea Group publishing.

Herlea, D. E., Jonker, C. M., Treur, J., and Wijngaards, N. J. E. (1999a). Specification of behavioural requirements within compositional multi-agent system design. *Lecture Notes in Computer Science*, 1647 :8–27.

Hilaire, V. (2000). Formal specification and prototyping of organizational based multi-agent systems. In *International Conference on the Design of Cooperative Systems*, Lecture Notes in Computer Science. Springer Verlag.

Hilaire, V., Gruer, P., Koukam, A., and Moudni, A. E. (2002a). Engineering soccer robots behaviours. In *2002 FIRA Robot Congress*, Seoul. KAIST Press.

Hilaire, V., Gruer, P., Koukam, A., and Simonin, O. (2007). Formal specification approach of role dynamics in agent organisations : Application to the satisfaction-altruism model. *International Journal of Software Engineering and Knowledge Engineering*, 17(5) :615–641.

Hilaire, V., Gruer, P., Koukam, A., and Simonin, O. (2008a). Formal driven prototyping approach for multi-agent systems. *International Journal of Agent Oriented Software Engineering*, 2(2) :246–266.

Hilaire, V., Koukam, A., and Gruer, P. (2002b). a multi-agent system featuring virtual character in play for theatre. In *Proceedings of the Agent-Day'02 workshop*.

Hilaire, V., Koukam, A., and Gruer, P. (2002c). A mechanism for dynamic role playing. In *Agent Technologies, Infrastructures, Tools and Applica-*

tions for E-Services, number 2592 in Lecture Notes in Artificial Intelligence. Springer Verlag.

Hilaire, V., Koukam, A., Gruer, P., and Müller, J.-P. (2000a). Vers une méthodologie formelle de spécification de systèmes multi-agents. In *Journées Francophones IAD & SMA*. Hermes.

Hilaire, V., Koukam, A., Gruer, P., and Müller, J.-P. (2001). Formal specification and prototyping of multi-agent systems. In Omicini, A., Tolksdorf, R., and Zambonelli, F., editors, *Engineering Societies in the Agents' World*, number 1972 in Lecture Notes in Artificial Intelligence. Springer Verlag.

Hilaire, V., Koukam, A., and Rodriguez, S. (2008b). An adaptative agent architecture for holonic multi-agent systems. *ACM Transactions on Autonomous and Adaptive Systems*, 3(1) :1–24.

Hilaire, V., Lissajoux, T., and Koukam, A. (1999b). Towards an executable specification of multi-agent systems. In Filipe, J. and Cordeiro, J., editors, *International Conference on Enterprise Information Systems'99*. Kluwer Academic Publisher.

Hilaire, V., Simonin, O., Koukam, A., and Ferber, J. (2005). A formal framework to design and reuse agent and multiagent models. In Odell, J., Giorgini, P., and Muller, J., editors, *Agent Oriented Software Engineering*, number 3382 in LNCS 3382. Springer.

Hübner, J. F., Sichman, J. S., and Boissier, O. (2007). Developing organised multiagent systems using the MOISE. *IJAOSE*, 1(3/4) :370–395.

Iglesias, C., Garrijo, M., and Gonzalez, J. (1999). A survey of agent-oriented methodologies. In Müller, J., Singh, M. P., and Rao, A. S., editors, *Proceedings of the 5th International Workshop on Intelligent Agents V : Agent Theories, Architectures, and Languages (ATAL-98)*, volume 1555 of *LNAI*, pages 317–330, Berlin. Springer.

Jennings, N. R., Sycara, K., and Wooldridge, M. (1998). A roadmap of agent research and development. *Autonomous Agent and Multi-Agent Systems*, 1.

Jennings, N. R. and Wooldridge, M. J. (1998). Applications of intelligent agents. In Springer-Verlag, editor, *Agent Technology : Foundations, Applications and Markets*.

Jerne, N. K. (1974). Towards a network theory of the immune system. *Ann Immunol (Inst Pasteur)*, 125C :373–389.

Kinny, D., Georgeff, M., and Rao, A. (1996). A methodology and modelling technique for systems of BDI agents. In *Modelling Autonomous Agents*

in a Multi-Agent World'96, number 1038 in Lecture Note of Artificial Intelligence '94.

Koestler, A. (1967). *The Ghost in the Machine*. Hutchinson.

Kolp, M., Giorgini, P., and Mylopoulos, J. (2006). Multi-agent architectures as organizational structures. *Autonomous Agents and Multi-Agent Systems*, 13(1) :3–25.

Koukam, A., Mazigh, B., Gruer, P., and Hilaire, V. (2003). A multiview approach to modeling and analysis of discrete event systems. *System Analysis - Modelling - Simulation*, 43(6) :721–740.

Koźlak, J., Créput, J.-C., Hilaire, V., and Koukam, A. (2006). Multi-agent approach to dynamic pick-up and delivery problem with uncertain knowledge about future transport demands. *Fundamenta Informaticae*, 71(1).

Loiseaux, C., Graf, S., Sifakis, J., Bouajjani, A., and Bensalem, S. (1995). Property preserving abstractions for the verification of concurrent systems. *Formal Methods in System Design : An International Journal*, 6(1) :11–44.

Lopez, F., y Lopez, and Luck, M. (2003). Modelling norms for autonomous agents. In *Proceedings of the Fourth Mexican International Conference on Computer Science*. IEEE Computer Society Press.

Luck, M. and D'Inverno, M. (1995). Structuring a Z specification to provide a formal framework for autonomous agent systems. *Lecture Notes in Computer Science*, 967 :47– ? ?

Luck, M., Griffiths, N., and d'Inverno, M. (1997). From agent theory to agent construction : a case study. In Springer-Verlag, editor, *Proceedings of the third International Workshop on Agent Theories, Architecture and Languages*, Lecture Note in Artificial Intelligence, pages 49–63.

Maes, P. (1989). How to do the right thing. Technical Report AIM-1180, MIT Artificial Intelligence Laboratory.

Mamei, M. and Zambonelli, F. (2004). Self-organization in multi agent systems : A middleware approach. In Serugendo, G. D. M., Karageorgos, A., and Rana, O. F., editors, *Engineering Self-Organising Systems : Nature-Inspired Approaches to Software Engineering*, number 2977 in LNCS.

Manna, Z., Bjoerner, N., Browne, A., and Chang, E. (1995). STeP : The Stanford Temporal Prover. *Lecture Notes in Computer Science*, 915 :793– ? ?

Manna, Z. and Pnueli, A. (1991). *The Temporal Logic of Reactive and Concurrent Systems : Specification*. Springer.

Matta, N., Ribiere, M., Corby, O., Lewkowicz, M., and Zaclad, M. (2000b). *Project Memory in Design, Industrial Knowledge Management - A Micro Level Approach*. Springer-Verlag.

Maturana, F., Shen, W., and Norrie, D. (1999). Metamorph : An adaptive agent-based architecture for intelligent manufacturing.

Mazigh, B., Hilaire, V., Gruer, P., and Koukam, A. (1999a). Agent oriented approach for modeling and simulation of productive organizations. In *10th DAAAM International Symposium ¡¡International Manufacturing & Automation : Past-Present-Future¿¿*. DAAAM.

Mazigh, B., Koukam, A., Hilaire, V., and Gruer, P. (1999b). Combining multi-agents approach with high level petri nets for simulation of productive organizations. In *IFAC Workshop on Multi Agent Systems in Production*. IFAC.

Miled, A. B., Hilaire, V., Monticolo, D., and Koukam, A. (2008). Reusing knowledge by multi agent system and ontology. In *Knowledge Acquisition Reuse and Evaluation*.

Miller, J. and Mukerji, J. (2003). *MDA Guide*. OMG. omg/2003-06-01.

Molesini, A., Omicini, A., Denti, E., and Ricci, A. (2005). SODA : A roadmap to artefacts. In Dikenelli, O., Gleizes, M. P., and Ricci, A., editors, *ESAW*, volume 3963 of *Lecture Notes in Computer Science*, pages 49–62. Springer.

Monticolo, D. (2008). *Une approche organisationnelle pour la conception d ?un système de gestion des connaissances fondé sur le paradigme agent*. PhD thesis, UTBM.

Monticolo, D., Gomes, S., Hilaire, V., and Serrafero, P. (2006a). A multi-agent architecture to synthesize industrial knowledge from a plm system. In *PLEDM'06*.

Monticolo, D., Hilaire, V., Gomes, S., and Koukam, A. (2008). A multi-agent system for building project memories to facilitate design process. *Integrated Computer-Aided Engineering*, 15(1) :3–20.

Monticolo, D., Hilaire, V., Koukam, A., and Gomes, S. (2007a). An e-groupware based on multi agents systems for knowledge management. In *DEST'07*.

Monticolo, D., Hilaire, V., Koukam, A., and Gomes, S. (2007b). Ontodesign ; a domain ontology for building and exploiting project memories in mechanical design projects. In *Knowledge Management in Organizations*.

Monticolo, D., Hilaire, V., Koukam, A., and Meunier, S. (2006b). An approach for building project memories to facilitate design process in a concurrent engineering context. In *proceedings of Concurrent Engineering'06*.

Monticolo, D., Hilaire, V., Serrafero, P., and Gomes, S. (2007c). Knowledge capitalization process linked to the design process. In *KMOM'07*.

Nejdl, W., Wolf, B., Staab, S., and Tane, J. (2002). EDUTELLA : Searching and annotating resources within an RDF-based P2P network. In Frank, M., Noy, N. F., and Staab, S., editors, *Workshop on the Semantic Web*, volume 55 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Nonaka, I. and Takeuchi, H. (1995). *The Knowledge-Creating Company*. Oxford University Press.

Nuseibeh, B., Kramer, J., and Finkelstein, A. (2003). Viewpoints : meaningful relationships are difficult ! In *ICSE*, pages 676–683. IEEE Computer Society.

Padgham, L. and Winikoff, M. (2002a). Prometheus : A methodology for developing intelligent agents. In *AOSE*.

Pavon, J., Gomez-Sanz, J., and Fuentes, R. (2005). The INGENIAS methodology and tools. In *Agent-Oriented Methodologies*, pages 236–276. Idea Group Publishing, NY, USA.

Preece, A. D., ying Hui, K., Gray, W. A., Marti, P., Bench-Capon, T. J. M., Jones, D. M., and Cui, Z. (2000). The KRAFT architecture for knowledge fusion and transformation. *Knowl.-Based Syst*, 13(2-3) :113–120.

Rao, A. S. and Georgeff, M. P. (1995b). BDI agents : from theory to practice. In Lesser, V., editor, *Proceedings of the First International Conference on Multi–Agent Systems*, pages 312–319, San Francisco, CA. MIT Press.

Rodriguez, S., Gaud, N., Hilaire, V., Galland, S., and Koukam, A. (2007a). An analysis and design concept for self-organization in holonic mas. In Brueckner, S., Hassas, S., Jelasity, M., and Yamins, D., editors, *Engineering Self-Organising Systems*, number 4335 in LNAI, pages 15–27.

Rodriguez, S., Gaud, N., Hilaire, V., and Koukam, A. (2006a). Modeling holonic systems with an organizational approach. In *proceedings of EUMAS'06*.

Rodriguez, S., Hilaire, V., Gruer, P., and Koukam, A. (2007b). A formal holonic framework with proved self-organizing capabilities. *International Journal of Cooperative Information Systems*, 16(1) :7–25.

Rodriguez, S., Hilaire, V., and Koukam, A. (2003). Towards a methodological framework for holonic multi-agent systems. In *Proceedings of the ESAW'03 workshop*, pages 179–185.

Rodriguez, S., Hilaire, V., and Koukam, A. (2005b). Multi-agent coordination is arbitration from a super-holon point of view. In *proceedings of MA4CS'05*.

Rodriguez, S., Hilaire, V., and Koukam, A. (2006b). An holonic approach to model and deploy large scale simulations. In *proceedings of MABS'06*.

Rodriguez, S., Hilaire, V., and Koukam, A. (2007c). Towards a holonic multiple aspect analysis and modeling approach for complex systems : Application to the simulation of industrial plants. *Simulation Modelling Practice and Theory*, 15(5) :521–543.

Rushby, J. (2003). Tutorial on mechanized formal methods. NICTA tutorial.

Sathyanath, S. and Sahin, F. (2002). AISIMAM - an artificial immune system based intelligent multi-agent model and its application to a mine detection problem. In Timmis, J. and Bentley, P. J., editors, *Proceedings of the 1st International Conference on Artificial Immune Systems (ICARIS)*, pages 22–31, University of Kent at Canterbury. University of Kent at Canterbury Printing Unit.

Simon, H. A. (1996). *The Science of Artificial*. MIT Press, Cambridge, Massachusetts, 3rd edition.

Simonin, O. and Ferber, J. (2000). Modeling self satisfaction and altruism to handle action selection and reactive cooperation. In *The Sixth International Conference on the Simulation of Adaptative Behavior FROM ANIMALS TO ANIMATS 6*, pages 314–323.

Sowa, J. F. (2000). Ontology, metadata, and semiotics. In Ganter, B. and Mineau, G. W., editors, *ICCS*, volume 1867 of *Lecture Notes in Computer Science*, pages 55–81. Springer.

SPEM (2007). *Software Process Engineering Metamodel Specification, v2.0, Final Adopted Specification, ptc/07-03-03*. Object Management Group.

Spivey, J. M. (1992). *The Z Notation : A Reference Manual*. Prentice Hall.

Suzuki, J. and Yamamoto, Y. (2000b). A decentralized policy coordination facility in openwebserver. In *¡proceedings of SPA2000*.

Tacla, C. A. and Barthès, J.-P. A. (2003). A multi-agent architecture for evolving memories. In (van Elst et al., 2004), pages 388–404.

Tolba, F., Simonin, O., and Hilaire, V. (2003). A reactive macro control for mirosot robot soccers. In *FIRA Robot World Congress Austria*.

Ulieru, M. and Geras, A. (2002). Emergent holarchies for e-health applications : a case in glaucoma diagnosis. In *IECON 02 [Industrial Electronics Society, IEEE 2002 28th Annual Conference of the]*, volume 4, pages 2957– 2961.

van Diggelen, J. and Dignum, V. (2006). Special issue on agent-mediated knowledge management. *KES Journal*, 10(4) :259–261.

van Elst, L. and Abecker, A. (2004). Agent-based knowledge management. *KI*, 18(2) :11–16.

van Elst, L., Dignum, V., and Abecker, A., editors (2004). *Agent Mediated Knowledge Management, International Symposium AMKM 2003, Stanford, CA, USA, March 24-26, 2003, Revised and Invited Papers*, volume 2926 of *Lecture Notes in Computer Science*. Springer.

Watanabe, Y., Ishiguro, A., and Uchkawa, Y. (1999). Decentralized behavior arbitration mechanism for autonomous mobile robot using immune system. *Books Artificial Immune Systems and Their Applications, Springer-Verlag, p. 186-208, ISBN 3-540-64390-7.*

Wooldridge, M. (1992). *The Logical Modelling of Computational Multi-Agent Systems*. Phd thesis, Manchester Metropolitan University.

Wooldridge, M. (1997). Agent-based software engineering. *IEE Proceedings on Software Engineering*, pages 26–37.

Wooldridge, M. and Jennings, N. R. (1998). Pitfalls of agent-oriented development. In Press, A., editor, *Proceedings of The Second International Conference on Autonomous Agents*.

Zambonelli, F., Jennings, N., and Wooldridge, M. (2003). Developing multiagent systems : the gaia methodology. *ACM Transactions on Software Engineering and Methodology*, 12(3).

Zave, P. and Jackson, M. (1993). Conjunction as composition. *acm Transactions of Software Engineering and Methodology*, 2(4) :379–411.

Résumé

Les Systèmes Multi-Agents (SMA) forment un paradigme prometteur pour la conception de systèmes logiciel complexes. En effet, ce paradigme propose de nouvelles stratégies pour analyser, concevoir et implémenter de tels systèmes. Les systèmes multi-agents sont considérés comme des sociétés composées d'entités autonomes et indépendantes, appelées agents, qui interagissent en vue de résoudre un problème ou de réaliser collectivement une tâche. Nous nous plaçons dans un cadre d'ingénierie logicielle pour ce mémoire. Pour tout nouveau paradigme d'ingénierie logicielle pour pouvoir être pleinement appliqué et déployé il est nécessaire de disposer de nouveaux modèles et d'abstractions nouvelles. Ces abstractions servent de base à l'analyse et à la conception des SMA. De plus, toute méthodologie dédiée aux SMA doit prendre ces abstractions pour pouvoir développer des SMA de manière systématique, sure, robuste et fiable. Les modèles à la base de tous les travaux dans ce mémoire sont composés de concepts organisationnels qui permettent de concevoir les SMA comme des sociétés d'individus, les agents, qui jouent des rôles dans des organisations. Ces concepts permettent également la description de structures organisationnelles complexes telles que les holarchies dans lesquelles les agents, désignés par le terme holons, peuvent être composés d'autres holons et définir une structure hiérarchique. La définition de ces concepts se fait au travers de méta-modèles exprimés avec une notation semi-formelle. Pour définir une sémantique non ambigüe et bénéficier d'outils de validation et de vérification nous proposons de spécifier des concepts au travers d'un langage formel. Aucun langage ne satisfaisant nos besoins nous avons composé deux langages existants : Object-Z et les statecharts. La syntaxe et la sémantique du langage formel OZS sont définies sur cette base. Des architectures d'agents sont étudiées en utilisant le modèle organisationnel et les outils formels. Ces études sont de deux types, définition complète d'une architecture et rétro-ingénierie afin de comprendre et analyser les éléments fondateurs d'architectures existantes pour pouvoir les réutiliser dans d'autres contextes. Deux méthodologies sont exploitant les concepts et la notation formelle sont proposées. La première est issue d'un travail initial concernant l'analyse et la conception de SMA Holoniques utilisant un cadre particulier. La deuxième est issue d'une collaboration avec Massimo Cossentino et est plus générale.