Résolution émergente et collective de problèmes par systèmes multi-agents : principes et applications

Collective and emergent problem solving based on multi-agent systems : principles and applications

Jean-Pierre Mano, Marie-Pierre Gleizes et Pierre Glize
Institut de Recherche en Informatique de Toulouse, Université Paul Sabatier,
118 route de Narbonne 31062 Toulouse cedex 4 (France),
{mano,gleizes,glize}@irit.fr www.irit.fr/SMAC

Manuscrit reçu le 1er mars 2005

Résumé et mots clés

Un système multi-agent est constitué d'un grand nombre d'entités, appelées agents, en interaction entre elles au sein d'un même environnement. Cette technologie aborde de nombreux domaines d'applications comme la vision par ordinateur, la robotique, la simulation de systèmes, le commerce électronique. Nous considérons que les questions abordées en traitement du signal sont très pertinentes dans un cadre multi-agent.
Nous présentons d'abord les principaux outils dont disposent les concepteurs de systèmes multi-agents à savoir : des modèles, des plates-formes et des méthodes de développement. Puis, le projet SCALA de simulation de résolution de problèmes par des patrouilles aériennes et un projet de simulation de système de transports illustrent la résolution de problèmes à l'aide de systèmes multi-agents.
Ensuite, nous nous intéressons plus particulièrement aux capacités d'adaptation de tels systèmes que nous abordons comme une question de résolution émergente de problèmes. Dans ce cadre nous décrivons en détail la théorie AMAS (Adaptive Multi-Agent System) qui permet de concevoir des systèmes dont la fonction globale émerge à partir d'un processus d'auto-organisation coopérative de ses parties. Une application en prévision de crues donne une indication plus précise des capacités de telles approches.

Système multi-agent, coopération, auto-organisation, emergence, néo-computation.

Abstract and key words

A multi-agent system is composed of numerous entities, called agents, interacting in various ways between them and their common environment. This technology is applied in many domains like computer vision, robotics, system simulation or electronic commerce. We consider that problems occuring in signal processing could also be tackled by this technology.
We present first the basic tools available for multi-agent systems designers : models, platforms and methodologies. Two projects illustrate our purpose : SCALA in the management of aerospace fighter patrol, and goods routing.
We focus then on the adaptation ability of these systems considered as an emergent problem solving question. We detailed in this field the AMAS (Adaptive Multi-Agent System) theory allowing a MAS design where the global fonction is derived from the cooperative self-organisation of its components. An example on flood forecast gives implementation information of this theory.

Multi-agent systems, cooperation, self-organisation, emergence, neo-computation.

1. Introduction

Historiquement, les systèmes multi-agents sont issus de l'Intelligence Artificielle Distribuée. Dans les années 70, la première génération de ces systèmes avait pour objectif de pallier les problèmes d'explosion combinatoire dus à la centralisation dans un même système des connaissances expertes, en les distribuant au sein de plusieurs modules. Le principal modèle de résolution distribuée de problèmes était basé sur l'architecture de tableau noir [Engelmore, 1988]. La deuxième génération de systèmes s'est intéressée à la distribution du contrôle et à la réutilisation des architectures. Dans ces systèmes, il existe une gradation dans l'intelligence d'un agent que l'on classe habituellement en trois catégories [Wooldridge, 1995] :

• *réactivité* : l'agent est uniquement dirigé par les événements perçus dans l'environnement. Il réagit de manière opportuniste à ces changements.

• *pro-activité* : l'agent n'agit pas simplement en réponse à des changements de l'environnement, mais est aussi capable de s'assigner des buts et de prendre des initiatives pour les atteindre ;

• *socialité* : l'agent interagit avec les autres agents et les humains.

Les travaux sur la troisième génération de systèmes concernent les interactions, l'interopérabilité, l'ouverture de ces systèmes et l'émergence.

Les systèmes multi-agents sont utilisés dans trois grandes familles de logiciels ayant des finalités différentes : la simulation, la résolution, l'intégration [Boissier, 2004] :

• L'ensemble des applications multi-agents pour la simulation modélisent et reproduisent des phénomènes du monde réel, afin de comprendre ou d'expliquer leur comportement. Ce sont par exemple, des applications de simulation de phénomènes sociaux, environnementaux, éthologiques...

• La résolution regroupe l'ensemble des applications issues de l'Intelligence Artificielle Distribuée [Bond, 1988]. Ces applications s'intéressent essentiellement à la problématique de la résolution de problèmes telle qu'elle avait été définie en Intelligence Artificielle étendue à un contexte distribué. Dans ce cadre, l'objectif de l'application est de mettre en œuvre un ensemble de techniques pour que des agents, résolvant une partie ou l'ensemble du problème, participent de manière efficace et cohérente à la résolution du problème global pour lequel l'application a été élaborée. On trouve un ensemble d'applications liées à des problèmes d'optimisation pour lesquels une solution centralisée est inadaptée [Ferber, 1995], mais aussi différentes applications de supervision de réseaux de télécommunications ou de réseaux de transport.

• Les applications d'intégration sont de plus en plus nombreuses et impliquent une hybridation de l'informatique avec les êtres humains et les automatismes. Leur principal objectif est d'intégrer des logiciels existants, des êtres humains et des systèmes mécaniques et de faire interopérer l'ensemble de manière cohérente et flexible. Ces applications visent à répondre à des exigences de plus en plus complexes, du fait de la multiplicité des services, des acteurs et des ressources. Ces applications ont trait notamment au commerce électronique, au collecticiel ainsi qu'à l'informatique diffuse.

L'objectif premier de cet article est de montrer l'intérêt intrinsèque des techniques multi-agents pour des applications qui sont constituées d'éléments actifs et en partie autonomes. À cette fin nous présentons de manière synthétique dans la deuxième partie les principaux outils dont disposent les concepteurs de systèmes multi-agents à savoir : des modèles, des plates-formes et des méthodes de développement. Deux projets de simulation illustrent la résolution de problèmes à l'aide de systèmes multi-agents : SCALA pour la résolution de problèmes par des patrouilles aériennes, et la gestion d'une flotte de transport routier.

Le second objectif porte sur une approche très spécifique dans les multi-agents : la capacité à résoudre des problèmes de manière émergente. Nous l'abordons dans la troisième partie car elle est certainement appropriée pour la coopération intelligente de moyens d'observation lorsque la situation courante à laquelle un système doit faire face est réellement imprévue. Peu de travaux au niveau international abordent cette voie et c'est pourquoi nous la présentons principalement à partir de la théorie et la technologie AMAS fondées sur l'auto-organisation coopérative. Nous illustrons la réalisation de tels systèmes à fonctionnalité émergente à partir d'une application en prévision de crues.

2. Principes et architectures des systèmes multi-agents

Les logiciels fondés sur la notion d'agent ou de multi-agent permettent de résoudre des problèmes complexes dans lesquels l'environnement est généralement contraint. Le prochain défi est de construire des logiciels plus robustes que ceux existant actuellement, dans des environnements imprévisibles tels que l'Internet ou les robots mobiles se déplaçant dans le monde réel. Il devient alors nécessaire de développer de nouveaux modèles, de nouvelles méthodes et de nouveaux outils. Comme nous allons le voir, les multi-agents peuvent être une de ces nouvelles approches. Après avoir donné les définitions des principaux concepts clés du domaine, les principales architectures de systèmes multi-agents sont présentées. Puis, les outils permettant de concevoir ces architectures sont décrits, à savoir : les plates-formes et les méthodes de conception orientées agent.

2.1. Définitions

Un agent est une entité (logicielle ou matérielle) douée d'autonomie (notamment de décision) qui est capable d'agir dans un

environnement. Cette entité est assujettie à un objectif individuel ou à une fonction à satisfaire ou à optimiser ; elle possède des ressources propres, elle est capable de percevoir son environnement mais n'en n'a, de manière générale, qu'une perception limitée. Afin d'atteindre son objectif, un agent met en oeuvre un comportement basé sur ce qu'il sait faire (ses compétences), ses perceptions, sa représentation du monde et les communications qu'il possède avec les autres ou avec son environnement.

Un système multi-agent peut se définir comme un macro-système composé d'agents autonomes qui interagissent dans un environnement commun afin de réaliser une activité collective cohérente [Sierra, 2002]. Le résultat de l'organisation de ces agents, les liens les reliant, définit l'identité du système multi-agent. Le point essentiel d'un système multi-agent réside dans le collectif des agents qui le constituent. Il faut évidemment qu'un système soit composé de plusieurs agents mais ce sont leurs comportements, plus leurs interactions, qui permettent au système d'avoir un comportement global cohérent.

L'interaction entre les agents passe par une communication et conditionne l'organisation du système. L'interaction entre agents a été étudiée dans [Ferber, 1995] selon deux axes : le partage de ressources et le partage de compétences. Les systèmes multi-agents appartenant à la famille « résoudre » sont constitués, en général, d'agents qui cherchent à être en interactions coopératives pour partager des compétences et/ou des ressources.

Les agents logiciels mettent en œuvre leurs communications de deux manières. D'une part, les agents peuvent communiquer de manière indirecte en laissant des traces dans leur environnement comme les fourmis laissent de la phéromone pour indiquer aux autres fourmis un chemin vers une source de nourriture. D'autre part, les agents peuvent communiquer par envoi de messages. De nombreux travaux, issus de la théorie des actes du langage de Searle, ont été réalisés dont KQML (Knowledge Query Manipulation Language) [Finin, 1994] et ACL (Agent Communication Language) de FIPA (http://www.fipa.org). Ces langages fournissent les principaux actes de langages que les agents peuvent utiliser au sein de protocoles tels que : affirmer, infirmer, refuser, ne pas comprendre .... La normalisation des ACL a pour objectif de faire interopérer différentes plates-formes agents.

Dans le domaine militaire la norme de spécification d'architecture HLA (High Level Architecture) [HLA1] [HLA2] est bien connue et employée pour les simulations où le temps réel est essentiel. Le fonctionnement est régi par dix règles portant à la fois sur la fédération et le fédéré. L'intérêt de HLA pour les systèmes multi-agents est bien analysé dans l'article de Pierre, Bourdon et Négret [Pierre, 2004].

2.2. Quelques architectures

Pour développer un système multi-agent, il faut concevoir d'une part, les agents et d'autre part, les interactions entre eux. Cette partie dresse un rapide panorama des principales architectures et modèles existants. Certains se focalisent sur l'aspect agent, par exemple dans les architectures délibératives et la subsumption ; d'autres mettent principalement l'accent sur les interactions comme le réseau de contrats, l'éco-résolution et la théorie AMAS (Adaptive Multi-Agents Systems).

2.2.1. Modèle du réseau de contrats

Dans le modèle du réseau de contrats [Davis, 1980], les agents peuvent jouer le rôle de contractant ou de manager. Ces rôles sont attribués aux agents de manière dynamique. Les agents communiquent par envoi de messages en suivant, en général, le protocole suivant, constitué de quatre phases : 1) un agent manager fait une annonce de tâche, 2) des agents contractants susceptibles d'être intéressés pour résoudre la tâche répondent à l'appel d'offre, 3) le manager choisit le contractant et lui envoie un message de décernement, 4) les deux agents établissent un contrat. Ce modèle ne donne pas de solution pour définir le temps d'attente d'une réponse par un agent, ni de stratégie de choix du bon contractant pour que le système fonctionne bien dans sa globalité, ni de stratégie de décomposition d'une tâche en sous-tâches.

Le langage de communication FIPA (Foundation for Intelligent Physical Agent) a défini à la fin des années 90 une série de primitives de communication avec une sémantique associée et un ensemble de protocoles d'interaction. Parmi ces protocoles d'interaction nous retrouvons le réseau de contrats qui est désormais très courant dans la communauté agent [Fipa, 2000].

2.2.2. Modèles d'architectures délibératives

Les architectures de systèmes multi-agents dans lesquels les agents sont cognitifs sont souvent appelées architectures cognitives ou délibératives. Les agents sont caractérisés essentiellement par leurs capacités de raisonnement complexe et leurs capacités de mémorisation.

Les premiers systèmes de cette classe dérivent des systèmes experts de l'Intelligence Artificielle classique dans lesquels la résolution de problèmes est distribuée dans les agents. On parlait de systèmes multi-experts (parmi lesquels : Archon [Wittig, 1992], MACE [Gasser, 1987], Synergic [Gleizes, 1990]....). Chaque agent de ce type de systèmes est capable de résoudre des problèmes pointus et de coopérer avec les autres pour pallier soit le manque de compétences, soit le manque de ressources. Ces agents communiquent par envoi de messages.

Actuellement, les systèmes délibératifs sont des systèmes pour lesquels les agents suivent le modèle BDI (Belief, Desire, Intention) [Rao, 1991]. Un agent est constitué de croyances, de désirs et d'intentions, c'est-à-dire respectivement de connaissances sur son environnement, de buts à réaliser, et de plans qu'il souhaite mettre en œuvre.

2.2.3. Modèle d'architecture avec des agents réactifs

Un agent réactif est un agent dont le comportement est du type réflexe : stimulus-réponse et dont les capacités de mémorisation sont très faibles. En général, un système conçu avec ce type d'agent est constitué d'un grand nombre d'agents dont le comportement collectif est qualifié « d'intelligent » en regard de leurs compétences individuelles réduites.

L'architecture de subsumption [Brooks, 1986] est une architecture possible pour réaliser un agent réactif. Pour cela, le concepteur définit une liste ordonnée de préconditions-actions. Les préconditions représentent des événements ou des résultats fournis par l'environnement du système. L'action effectuée par l'agent à un instant donné, est la première de la liste dont les préconditions sont vérifiées.

Dans l'éco-résolution [Bura, 1991], un agent a un but à résoudre et pour cela il dispose des actions suivantes :

- satisfaire son but ;
- chercher à satisfaire son but ;
- agresser[1] un agent qui le gêne dans sa quête de la satisfaction de son but ;
- fuir en réponse à une demande d'agression.

Le système a terminé sa résolution quand son organisation est stabilisée : les agents ne peuvent plus effectuer d'action. Cette technique impose de résoudre deux principales difficultés : d'éviter le bouclage d'actions entre un pool d'agents et de conduire à une solution sous-optimale.

2.2.4. Architectures basées sur l'auto-organisation

Un AMAS (Adaptive Multi-Agent System) est un système multi-agent qui s'adapte en permanence à son environnement. Cette adaptation est réalisée par auto-organisation basée sur la coopération. Un système en interaction coopérative avec son environnement est dit fonctionnellement adéquat[2]. Pour cela, les agents à l'intérieur du système doivent également être coopératifs. La coopération idéale est définie selon trois points :

- Perception. Un agent devrait idéalement être apte, à tout instant, à « comprendre » sans ambiguïté le sens des signaux perçus, s'il est situé correctement dans l'environnement et si ses capacités perceptives et cognitives sont corrélées.
- Raisonnement. Les signaux reçus étant supposés « compris », ils deviennent une information interprétable que l'agent doit pouvoir exploiter compte tenu de ses compétences
- Action. Du point de vue de l'action, les conclusions (résultats de la fonction de l'agent) doivent être utiles à autrui ou à l'environnement.

Le concepteur dote donc les agents de critères locaux pour juger de situations coopératives ou non coopératives. Le moteur de l'auto-organisation est constitué par la détection puis l'élimination des situations non coopératives entre les agents. Dans la théorie AMAS, l'architecture d'un agent rassemble des compétences, des croyances, un langage d'interaction, des aptitudes et une attitude sociale qui est la coopération. Tous ces points seront développés dans la troisième partie.

2.3. Les plates-formes

Les plates-formes multi-agents fournissent des services qui simplifient le développement d'une application multi-agent.

Une plate-forme peut prendre en charge la recherche d'un agent ayant une compétence particulière, ce qui est le cas dans la plate-forme Magique [Mathieu, 2000]. Magique se base sur une communauté de hiérarchie d'agents. Un agent est une coquille vide dans laquelle le concepteur code les compétences de l'agent. Cette plate-forme se présente sous la forme d'une API Java, elle gère automatiquement le multi-threading et la communication entre les agents. Elle est développée à l'université de Lille (http://www.lifl.fr/SMAC/projects/magique/) et est en *open source*.

Certaines plates-formes sont dédiées à une architecture d'agent, comme Zeus qui permet la conception d'agents BDI (http://more.btexact.com/projects/agents/zeus/).

D'autres sont basées sur un méta-modèle, comme la plate-forme Mask basée sur le modèle AEIO [Demazeau, 2001] ou la plate-forme Madkit (http://www.madkit.org) qui est basée sur le modèle Agent-Groupe-Role. Madkit [Gutknecht, 1998] fournit un noyau minimal qui assure les fonctions de communications locales, la gestion du cycle de vie d'un agent et le maintien des groupes et rôles locaux. Les services tels que la communication distante, la gestion de groupes distribués, la migration d'agents... sont fournis par des méta-agents. Madkit n'est pas dédiée à une architecture d'agent en particulier mais requiert du développeur de connaître au préalable les rôles des agents et des groupes auxquels ils appartiennent.

En général, les plates-formes ne sont pas dédiées à une architecture particulière d'agent. C'est le cas de Dima développée, à l'université de Paris VI, en Smalltalk et en Java [Guessoum, 1996]. La brique de base est le composant qui est autonome et proactif. Le noyau de base de Dima est composé de plusieurs classes et de leurs méthodes, notamment de la classe ProactiveComponent. Une instance de cette classe décrit les compétences de base du composant, son but et son comportement c'est-à-dire la manière dont les compétences sont sélectionnées pour être activées.

Jade (Java Agent DEvelopment Framework) (http://jade.tilab.com) [Bellifeme, 2001] est une plate-forme *open source* développée en Java par Telecom Italia Lab. Elle est

1. Les SMA emploient largement une terminologie d'inspiration anthropomorphique ou sociologique qui était à l'origine plutôt métaphorique et sont devenues des concepts. C'est déjà le cas avec les termes d'agents ou de société et l'on retrouvera souvent dans le texte d'autres notions comme la croyance, l'intention, l'engagement, la coopération, le conflit ou la concurrence. L'éco-résolution utilise les notions de satisfaction, agression, fuite.

2. Comme les agents ont une compétence limitée et une connaissance réduite de l'environnement, aucun ne peut décider si la fonction globale est correcte. À cette fin, nous employons la notion d'adéquation fonctionnelle qui est une attribution faite par un « observateur hypothétique » extérieur au système et qui connaîtrait la fonction globale correcte à réaliser dans l'environnement.

utilisée pour le développement d'applications multi-agents distribuées basées sur une communication pair à pair. Elle est conforme aux spécifications de la FIPA (Foundation of Intelligent Physical Agents) relatives au nommage des services, des services pages jaunes, du transport des messages et de la librairie des protocoles d'interaction FIPA. Toutes les communications entre les agents sont des envois de messages écrits dans le langage ACL (Agent Communication Language). Les agents peuvent être distribués sur plusieurs machines hôtes. Une seule application Java et, par conséquent, une seule machine virtuelle Java est exécutée sur chaque hôte. En général, un agent est implémenté par un *thread* mais l'architecture d'un agent est à définir par le concepteur.

2.4. Méthodologies de conception orientées multi-agent

Les méthodes de conception orientées agent [Bergenti, 2004] visent à construire des systèmes dans lesquels les compétences sont attribuées à des entités logicielles autonomes, qui interagissant dans un environnement commun, ont la faculté de naviguer dans les réseaux pour accomplir des tâches relevant de leurs compétences définies au moment de la conception du système. Une méthode de développement de systèmes multi-agents doit être constituée d'un processus, d'une notation et d'outils pour supporter ce processus et ces notations et/ou pour aider le développeur. Il est actuellement largement admis que les phases principales d'une méthode sont l'analyse des besoins, l'analyse, la conception, le développement (appelé aussi implémentation) et le déploiement [Jacobson, 1999]. Les travaux sur les méthodes ont essentiellement porté sur les trois premières phases qui vont donc être présentées ci-dessous.

2.4.1. L'analyse des besoins

L'analyse des besoins, appelée aussi spécifications dans certaines méthodes orientées agent, est l'étape au cours de laquelle la fonction du système est exprimée du point de vue des utilisateurs. Cette étape peut à son tour être divisée en deux sous-étapes : les besoins préliminaires et les besoins finals.

• Les besoins préliminaires représentent un travail d'intercompréhension et/ou une description consensuelle du problème du cahier des charges entre clients, utilisateurs et concepteurs sur ce que doit être et ce que doit faire le système, ses limites et ses contraintes. Cette phase doit permettre de transformer les besoins exprimés par le client en un cahier des charges consensuel entre client et fournisseur.

• Les besoins finals fournissent un modèle de l'environnement. Les objectifs sont de définir une vue du système, d'organiser et de gérer les besoins (fonctionnels ou non) et leurs priorités. De manière pratique, à ce stade, le concepteur doit définir le système étudié et modéliser l'environnement du système.

Pour certaines méthodes orientées agent telles que Gaia [Wooldridge, 1999] Message[3] [Caire, 2001], cette phase du processus de développement est la même que l'analyse des besoins dans les méthodes classiques. Dans d'autres méthodes, par contre, elle a une place prépondérante, comme dans Adelfe[4] [Bernon, 2002], MaSE[5] [Deloach, 2001] Passi[6] [Burrafato, 2002] et Tropos [Castro, 2001].

Les deux notions importantes de cette phase sont la prise en compte des rôles dans le modèle et la notion d'environnement. Le rôle représente une description de haut niveau d'abstraction de l'agent. Les rôles dans AAII[7] [Kinny, 1996] permettent l'élaboration d'une hiérarchie de classes d'agents. Le rôle est un ensemble de responsabilités qui sont décrites par des services. Dans Gaia, le rôle est défini par quatre attributs : responsabilités, permissions, activités et protocoles. À partir des besoins, le modèle de rôle identifie les rôles-clés que les agents devront remplir et nécessaires à l'accomplissement de la tâche du système. Dans MaSE, la définition des rôles est une des sept étapes de la méthode et consiste à associer des buts à un rôle. Massive[8] [Lind, 2001] propose une vue selon les rôles qui permet de regrouper de manière fonctionnelle les différentes compétences nécessaires pour la résolution de problèmes. Dans Message, les rôles et les agents sont les concepts principaux entrant en jeu dans le modèle d'agent. Les rôles sont définis dans Passi au sein du modèle de la société d'agents. Les acteurs, produits lors de l'analyse des besoins, peuvent être des agents, des positions ou des rôles dans Tropos.

Bien que l'environnement soit déjà présent dans la définition du contexte d'un logiciel classique, l'étude des interactions du système avec son environnement nécessite d'être réalisé de manière approfondie lors de l'étude du système à construire. Dans AAII, Adelfe, Massive, Message Prometheus et Tropos, le modèle de l'environnement est élaboré au cours des besoins.

2.4.2. L'analyse

Cette étape, appelée aussi conception de l'architecture dans certaines méthodes orientées agent, doit permettre de fournir la description des besoins de l'utilisateur relativement au domaine et à la fonction du système. Elle décrit le problème à résoudre par le système (s'il est clairement identifié) et définit l'architecture du système.

En effet, les applications dédiées aux systèmes multi-agents ne possèdent pas toutes une tâche clairement définie. Si la tâche est bien définie, les produits de l'analyse sont utilisés pour établir les fonctions du système. Par fonction, on entend les comportements visibles et testables de l'extérieur du système. Si la tâche n'est pas bien définie, la phase d'analyse doit permettre une description des entités réelles intervenant dans l'application.

3. MESSAGE : Methodology for Engineering Systems of Software AGents

4. ADELFE : Atelier de DEveloppement de Logiciels à Fonctionnalité Emergente.

5. MaSE : Multiagent Systems Engineering.

6. PASSI : Process for Agent Societies Specification and Implementation.

7. AAII : methodology of the Australian Artificial Intelligence Institute.

8. MASSIVE : Multi-Agent SystemS Iterative View Engineering.

L'analyse est basée sur des méthodes formelles ou semi-formelles, comme les diagrammes UML, ou sur des modèles. Les méthodes formelles correspondent, par exemple, à l'utilisation de machines d'états comme dans MaSE, de formules logiques comme dans Gaia. Les diagrammes UML[9] et leur extension aux protocoles d'interactions entre agents (AUML[10] [Odell, 2000]) sont utilisés dans Adelfe, Message ou Passi. Les méta-modèles, comme le modèle Agent-Groupe-Rôle [Gutknecht, 1998], permettent de donner une architecture du système à construire en définissant l'organisation dans le système. Le modèle Vowels [Demazeau, 2001] permet de construire les briques du système multi-agent en termes d'Agent, d'Environnement, d'Organisation, et d'Interactions.

Les deux étapes essentielles de cette phase liées aux systèmes multi-agents sont l'identification des agents qui interviennent dans le domaine d'application et la définition des interactions entre ces agents.

L'identification des agents est plus ou moins simple à effectuer. Dans certaines applications, comme les simulations ou la modélisation de sociétés humaines, les concepteurs n'ont aucune difficulté pour cerner les entités que sont les agents dans le système. Par contre, dans des applications de type résolution de problèmes, comme la gestion d'emplois du temps, le processus de gestion d'équipes ou de résolution de systèmes d'équations, il est *a priori* difficile de savoir ce qui sera un agent dans le système. Pour cette raison, il est nécessaire d'aider le concepteur à identifier les agents dans une méthodologie orientée agent. Dans les méthodes comme Gaia, MaSE, Massive, Passi, Prometheus [Padgham, 2002] le rôle des entités du système est d'abord étudié et correspond à la fonction qu'elles exercent. Ensuite les rôles sont liés à des agents, par exemple, Prometheus intègre dans un agent plusieurs rôles. Tropos ne fournit aucune démarche à suivre, les agents se trouvent dans l'ensemble des acteurs (terme employé dans Tropos pour signifier les entités actives ou passives). Dans AAII, l'élaboration et le raffinage des modèles d'agents et d'interactions aident le concepteur à définir les classes d'agents et à donner la multiplicité et la durée de vie des instances de classes d'agents. Un modèle d'agencement à fine granularité est défini à partir des rôles, des responsabilités et des services. La définition d'agent donnée dans Message et Adelfe, décrit les caractéristiques auxquelles un agent doit souscrire pour que le concepteur puisse choisir parmi les entités lesquelles sont effectivement des agents.

Au cœur des systèmes multi-agents, les interactions entre agents sont étudiées dans toutes les méthodes au travers de scénarios. Les travaux les plus avancés concernent le langage UML étendu aux agents : AUML qui prend en compte l'écriture de ces interactions. Les concepteurs de Message considèrent que la définition des protocoles d'interaction entre les agents se trouve à la frontière entre l'analyse et la conception.

9. UML : Unified Modelling Language.

10. AUML : Agent UML.

2.4.3. La conception

La conception, appelée aussi conception détaillée, a pour objectif de définir les entités et les mécanismes qui donnent le comportement établi lors de la phase d'analyse et l'architecture détaillée du système. Elle permet de régler les problèmes liés à l'implémentation. La conception consiste à dire comment le système doit être construit à partir de l'architecture instaurée pendant la phase d'analyse.

Cette phase peut démarrer dès qu'un modèle de ce que doit faire le système a été établi. Ce modèle va être enrichi progressivement au cours de la phase de conception. À l'issue de cette phase, un modèle de l'architecture logicielle du système doit être réalisé et la conception détaillée fournit les composants logiciels à coder. La conception doit définir une architecture détaillée du le système en termes de paquetages, de sous-systèmes, d'objets et d'agents. L'architecture détaillée est obtenue par raffinage des diagrammes produits dans l'analyse comme dans MaSE. Depuis plusieurs années, différents modèles et architectures d'agents sont apparus. Cela a conduit les chercheurs à développer des méthodologies pour leurs théories ou leurs modèles d'agents. Dans certaines méthodes, une architecture agent donnée guide donc le concepteur pour doter les agents de capacités perceptives, d'un comportement et de capacités d'action. Message, par exemple, propose les deux possibilités.

Pour la méthodologie AAII, le modèle interne s'intéresse aux composants internes de l'agent et se base sur l'architecture d'agent BDI. Tropos et Prometheus permettent de mettre en oeuvre des architectures de type BDI, mais la seconde étend les possibilités aux familles d'agents utilisant la planification. Pour Adelfe, lors de la conception au niveau des agents, le modèle d'agent proposé au concepteur est l'architecture d'agent coopératif [Georgé, 2003]. Cette architecture d'agent est liée à la théorie AMAS qui est à la base de la conception de systèmes à fonctionnalité émergente. Dans Desire, un modèle d'agent générique est utilisé. Il est composé de six modules : le contrôle, la gestion des interactions entre agents et avec le monde extérieur, la gestion des informations sur les agents, sur le monde et sur les tâches spécifiques à l'agent. Dans d'autres méthodologies, comme Gaia, Message ou Massive, l'architecture des agents n'est pas définie et reste ainsi ouverte.

2.5. Le projet SCALA

Le projet SCALA (Systèmes Coopératifs d'Agents Logiciels Autonomes) (Degirmencyan, 2002) développé au sein de DASSAULT Aviation est un système multi-agent appliqué à la simulation de missions aériennes dans un environnement dynamique et imprévisible. Un avion est modélisé par un agent. Une patrouille composée de plusieurs agents a une mission à accomplir et doit réagir aux événements multiples qui peuvent se produire dans l'environnement.

2.5.1. Les objectifs du projet

Les objectifs de SCALA sont de :
• fournir un outil de prototypage rapide de systèmes multi-agents ;
• faciliter la modélisation et la conception grâce à la description d'un graphe de dépendances fonctionnelles à un haut niveau d'abstraction ;
• simuler différents types d'organisation et de protocoles de communication entre agents ;
• constituer des bibliothèques de comportements et de protocoles ;
• proposer des outils d'analyse du comportement global et local.

2.5.2. Les aspects techniques

Le comportement global du système est décrit par le concepteur sous la forme d'un graphe de dépendances fonctionnelles. Dans ce graphe apparaissent des tâches ou des comportements élémentaires qui seront exécutés par un agent. Le concepteur spécifie le nombre d'agents nécessaires pour exécuter la tâche globale. Le concepteur isole alors des graphes partiels ou sous-graphes dans le graphe global. Chaque but est ensuite associé à un sous-graphe ainsi que des événements déclencheurs. À un événement est associé une priorité qui permet à l'agent de choisir le bon sous-graphe. Les agents doivent parfois se réorganiser pour atteindre leurs buts. En cours de fonctionnement, un nouvel événement perçu par un agent correspond, pour cet agent, à l'exécution d'un nouveau sous-graphe.

2.5.3. Résultats

Cette plate-forme a été testée sur un scénario d'interception faisant intervenir quatre avions en division. Cette mission requiert une forte coopération entre les avions. Les ordres et le contrôle sont réalisés par un avion radar et des contrôleurs aériens au sol. Les avions quittent le sol et se dirigent vers un point de rendez-vous pour voler en formation. Puis, une fois en formation, le leader vérifie les éventuels changements avec les différents contrôleurs. Il demande la permission d'entrer dans la zone de combat et de bombarder. Puis la division retourne à la base.
L'arrivée d'une nouvelle menace demande à la division pendant qu'elle vole en formation, de prendre en compte ce nouveau but. Si les agents ont des ressources suffisantes ils peuvent essayer d'atteindre plusieurs buts en parallèle en scindant la division en deux, par exemple. La difficulté est de trouver les paramètres déterminants pour assurer que chaque nouveau groupe a assez de ressources pour atteindre son nouveau but, sinon c'est la priorité des éléments déclencheurs qui indique à l'agent ce qu'il doit faire.

2.6. La simulation d'un système de transport

2.6.1. Objectif du projet

L'objectif de ce projet [Sassi, 1997] est d'étudier la planification et la négociation dans les systèmes de transports vus comme des systèmes multi-agents.
Les compagnies de transports disposent d'une flotte de camions et doivent répondre à un certain nombre de demandes de livraisons, arrivant de manière asynchrone et dynamique. Les camions doivent donc négocier pour minimiser le coût global du plan global de livraison. La compagnie envoie la commande suivante au camion : « transporter u1 unités de la marchandises m1 de la localité l1 à la localité l2 ». Ce problème est une généralisation du problème du voyageur de commerce réputé pour être NP complet.

2.6.2. Présentation

Dans ce projet, les compagnies de transports et les camions sont des agents. Les commandes des clients sont allouées aux compagnies selon le modèle du réseau de contrats (décrits dans la partie 2.2.1.)
Le partitionnement des commandes dans le plan des agents camion est réalisé selon le commerce simulé qui permet à des camions d'échanger des commandes. Cette stratégie est basée sur la théorie de l'utilité. Les échanges de commandes définissent un graphe bi-partie à niveaux. Ses nœuds correspondent à une insertion ou à une suppression d'une commande dans le plan. Les arêtes représentent les échanges possibles, elles possèdent un poids positif ou négatif correspondant au gain obtenu par l'action. Si la somme de tous les poids des arêtes est positive alors une amélioration du plan global est possible. La procédure du commerce simulé effectue des opérations d'insertions et de suppressions de tâches ainsi qu'une recherche de relations d'appariement entre une demande de tâche et une proposition pour effectuer la tâche. Pour réaliser cet appariement plusieurs stratégies ont été implémentées dans les agents telles que l'algorithme A• qui est un algorithme optimal, gourmand en temps de calcul et en espace mémoire, l'algorithme Tabou, le recuit simulé et un algorithme génétique.

2.6.3. Résultats

Le problème testé comporte 100 localités et 100 commandes qui sont proposées de manière aléatoire aux agents camions. Aucune localité n'apparaît plus d'une fois. Les paramètres pris en compte sont : la distance parcourue par les camions, le nombre de camions utilisés, le temps d'exécution de l'algorithme.
L'algorithme A• donne les meilleurs résultats quant à la distance parcourue quand le nombre de commande est inférieur à 90 et quand le problème est fermé. En général, les résultats montrent que le commerce simulé est une bonne stratégie. L'application des heuristiques : algorithmes génétiques, tabou et

recuit simulé à la stratégie du commerce simulé améliore le coût de respectivement 7%, 13% et 18%. Le recuit simulé, la métaheuristique de recherche tabou et les algorithmes génétiques sont plus rapides en temps de calcul que l'algorithme A•.

2.7. Analyse

Le panorama de cette partie montre que les agents sont des éléments actifs car ils intègrent systématiquement des senseurs (pour observer d'autres agents ou l'extérieur du système) et sont capables d'agir dans leur environnement (par leurs effecteurs ou la communication avec autrui). Les théories et techniques des systèmes multi-agents ont pour objet de mettre en synergie les aptitudes de chacun de ses agents. Les deux applications en simulation multi-agent montrent qu'ils peuvent apporter beaucoup dans les situations d'engagement[11] coopératif, dans lesquelles les acteurs humains et artificiels interagissent.
Cette technologie se rapproche intuitivement de la fusion de données où il faut combiner des éléments d'information issues de sources multiples, dans le sens où le processus de fusion d'un SMA y serait le résultat de l'activité collective. Lorsque les agents qui encapsulent les senseurs sont de granularité assez importante, cela rejoint les préoccupations qui sont aujourd'hui celles de l'exploitation intelligente de senseurs dans les systèmes complexes [REE, 2004].

3. Émergence et auto-organisation dans les systèmes-multi-agents

Lorsque les agents sont situés dans un environnement évolutif ou que le système est ouvert (ajout ou suppression d'agents), ils doivent apprendre pour faire face au mieux à ces changements. C'est le contexte qui correspond à celui exprimé dans la conférence Cogis [Appriou, 2003] : « il s'agit en définitive de passer du mode séculaire de l'avion piloté indépendant (ou de la patrouille quasi-autonome) à l'opération multi-agent conduite à distance et en temps quasi-réel, à travers les différentes boucles de réaction nécessaires – courtes et longues, humaines et automatiques. Arriver à mettre en place, dans ce contexte de coopération entre robots et opérateurs humains, un fonctionnement suffisamment réactif et adaptatif pour réaliser les opérations précises nécessaires constitue un enjeu majeur. La gestion des différents canaux de transmission et des boucles, humaines ou automatiques, d'information et de décision devient de ce fait un élément clé dans la définition des systèmes aéroportés futurs ».

Des techniques dérivées d'apprentissages standard en intelligence artificielle (renforcement, Q-Learning, classifieurs...) sont souvent employées pour ces situations dans les systèmes multi-agents. Nous pouvons en avoir des aperçus assez complets dans les travaux de Kaelbling [Kaelbling, 1996], Weiss [Weiss, 1997] ou Stone [Stone, 2000]. Mais plutôt que de développer des théories bien étudiées hors du cadre multi-agent et dont les limitations sont déjà bien connues notamment avec le « No free lunch theorem » [Wolpert, 1997], nous préférons focaliser cette troisième partie sur l'adaptation émergente par auto-organisation qui est beaucoup plus spécifique aux systèmes multi-agents. Nous introduisons tout d'abord le concept d'émergence dans les systèmes artificiels, puis une résolution émergente de problèmes par auto-organisation coopérative que nous montrons ensuite dans une application en prévision des crues.

3.1. Le concept d'émergence

« *Emergent Computation* » est le terme très général pour désigner toute utilisation de l'émergence par des informaticiens. L'idée n'est pas nouvelle comme le montrent les deux paragraphes suivants basés sur la volumineuse introduction de Stéphanie Forrest pour les « *Proceedings of the ninth annual CNLS conference* » [Forrest, 1990], conférence montrant bien toute la diversité de la computation émergente mais également à quel point elle était balbutiante.
La computation émergente s'efforce de produire un comportement global intéressant à partir de nombreuses petites interactions. Cela inclut les modèles connexionnistes, les systèmes classifieurs, les automates cellulaires, les modèles biologiques, les modèles de vie artificielle, et l'étude de la coopération dans les systèmes sociaux sans autorité centrale.
Le principe de base ressemble à celui de la computation parallèle dans le sens où elle s'efforce de décomposer des systèmes complexes en sous-unités indépendantes qui interagissent. Mais là où la computation parallèle requiert une supervision contraignante pour contrôler le bon fonctionnement du système avec des performances considérablement inférieures à une fonction linéaire du nombre de processeurs, la computation émergente va exploiter ces interactions pour accroître l'efficacité, la flexibilité et obtenir une représentation plus naturelle.
Près de quinze ans après, l'utilisation de l'émergence s'est amplement répandue dans le domaine de l'informatique. De nombreuses disciplines ou champs d'étude manipulent le terme « émergence », tels que la Vie Artificielle, l'*Evolutionary Computation* (Algorithme Génétiques et *Genetic Programming*), les Réseaux Neuronaux, les *Ants Algorithms*, les Automates Cellulaires, les Systèmes Multi-Agents, la Simulation (de systèmes physiques, chimiques, biologiques, sociaux), la Robotique, l'Intelligence Artificielle Distribuée,... Cette synergie interdisciplinaire autour de ce concept encore inexploité peut sembler prometteuse et aboutir rapidement à une

11. L'agent s'emploie à mettre ses compétences au service d'un objectif collectif précédemment négocié.

utilisation répandue et concrète de l'émergence dans les travaux informatiques.

Cependant, lorsque l'on étudie plus en profondeur un panel de travaux tirés de tous ces champs, on s'aperçoit que la notion d'émergence est fortement sujette à interprétations, et ce même au sein d'une discipline. En fait, souvent, l'émergence est utilisée seulement au sens d'une apparition de propriétés à un certain niveau qui n'existent pas au niveau en dessous, sans prendre en compte les caractères d'irréductibilité et d'imprédictibilité [Brunner, 2002]. Parfois ce n'est même qu'une façon abusive de parler pour désigner l'apparition d'évènements pourtant entièrement descriptibles en terme de causalité.

Une raison relativement consensuelle dans la communauté agent et multi-agent pour laquelle les SMA seraient concernés par l'émergence est donnée par Pfeifer : « *One of the main motivations to employ autonomous agents is the idea of emergence. Autonomous agents, by definition, behave in the real world without human intervention. One of the fascinating features of autonomous agents is that they exhibit so-called emergent behaviours, that is, behaviours not programmed into the agents by the designers* » [Pfeifer, 2001]. Doter les agents d'un comportement autonome (et sans coder la fonction du collectif dans l'agent) peut effectivement entraîner un comportement collectif émergent. Mais l'autonomie des agents seule ne garantit pas un fonctionnement acceptable du système dans son ensemble car il est peu probable (compte tenu de l'espace gigantesque des états du système) que le comportement émergeant du collectif converge vers une activité adéquate (Voir note 2). Or, ce que nous cherchons à faire en tant que concepteur de systèmes ce sont des systèmes qui réalisent une fonction ou une activité donnée.

Une « maîtrise » des phénomènes émergents dans les systèmes artificiels nécessite l'élaboration de théories qui posent à la fois un cadre scientifique rigoureux quand à la définition de l'émergence, et fournissent les moyens techniques d'une construction effective de systèmes cohérente avec la définition. Particulièrement, le système ne doit pas et ne peut pas contenir explicitement ni la connaissance du but du système (*i.e.* de la fonction émergente), ni les moyens d'y arriver. C'est ainsi qu'une définition « computationnelle » de l'émergence comprend deux parties, l'une définit ce que l'on veut faire émerger et l'autre la condition dans laquelle il y a émergence :

Objet - Un système informatique a pour finalité de réaliser une fonction adéquate avec ce que l'on attend du système. C'est cette fonction, pouvant évoluer au cours du temps, que nous voulons faire émerger.

Condition - Cette fonction est émergente si le codage du système ne dépend aucunement de la connaissance de cette fonction. Ce codage doit contenir des mécanismes permettant l'adaptation du système au cours de ses échanges avec l'environnement afin de tendre, à tout instant, vers la fonction adéquate (Voir note 2).

Nous nous intéressons plus particulièrement aux systèmes multi-agents adaptatifs. Au sein de ces systèmes, l'organisation des agents n'est pas fixe mais se transforme suivant les besoins du système et sa confrontation à l'environnement. Ces modifications non supervisées des interactions entre agents est appelée auto-organisation. Tout l'intérêt de ces systèmes se trouve dans l'émergence de structures organisationnelles adéquates à la tâche du système, ceci étant produit par les interactions entre les agents. Nous allons donc présenter une façon de construire et manipuler les systèmes multi-agents adaptatifs : la théorie AMAS.

3.2. La théorie AMAS (Adaptive Multi-Agent Systems)

Par principe, la finalité émergeante d'un système n'est pas connaissable du système lui-même, son seul critère devra être de nature strictement locale (relativement à l'activité des entités qui le composent). Dans cette partie, nous apportons en premier lieu l'explication théorique de l'interdépendance qui peut exister entre un comportement local coopératif et l'adéquation fonctionnelle de la fonction globale collective. Nous montrons ensuite comment cette théorie peut s'appliquer aux systèmes multi-agents adaptatifs.

3.2.1. Théorème de l'adéquation fonctionnelle

Théorème. *Pour tout système fonctionnellement adéquat, il existe au moins un système à milieu intérieur coopératif qui réalise une fonction équivalente dans le même environnement.*

Le milieu intérieur d'un système est constitué de toutes ses parties et des supports nécessaires à leurs échanges. La démonstration de ce théorème [Camps, 1998] se déduit de l'application de l'axiome[12] et des quatre lemmes présentés dans le tableau 1 par des opérations de surjection et d'inclusion d'ensembles.

Ce théorème dit qu'un système particulier (à milieu intérieur coopératif) possède une propriété uniquement observable et inconnue de lui (son adéquation fonctionnelle dans un environnement) est garanti par des critères strictement locaux et connaissables par ses parties (leur activité coopérative). Il possède plusieurs propriétés permettant de situer cette théorie dans les théories de l'émergence :

• Un système coopératif dans l'environnement est fonctionnellement adéquat, ce qui lui évite de connaître la fonction globale qu'il doit réaliser pour s'adapter.

• Même si un système n'a pas de but explicite, il peut agir pertinemment dans son milieu. En fonction de ses perceptions de l'environnement, des représentations qu'il en possède et de ses compétences, il agira au mieux pour que son comportement soit coopératif.

12. Cet axiome joue un rôle similaire, pour la classe des systèmes fonctionnellement adéquats, à la thèse de Church pour les fonctions effectivement calculables.

Tableau 1. Récapitulatif du théorème de l'adéquation fonctionnelle.

Axiome et Lemmes	Explications
Un système fonctionnellement adéquat n'a aucune activité antinomique sur son environnement.	La véracité de cette assertion ne peut pas être prouvée, car il faudrait un observateur extérieur à l'activité de tous les systèmes évoluant dans un certain univers physique, tout en n'interagissant aucunement avec celui-ci afin de ne pas le perturber.
Tout système coopératif est fonctionnellement adéquat.	La démonstration s'appuie sur l'axiome précédent, car par définition un système coopératif n'a pas d'activité antinomique.
Pour tout système S fonctionnellement adéquat, il existe au moins un système coopératif S• qui soit fonctionnellement adéquat dans le même environnement.	La démonstration consiste en une expérience de pensée de déconstruction du système S pour en construire un nouveau S•. Elle est réalisée en quatre étapes : définir un algorithme de construction d'un système coopératif, montrer que cet algorithme se termine, montrer que le système coopératif obtenu est équivalent au système initial pour l'environnement, montrer que le nouveau système est fonctionnellement adéquat.
Tout système à milieu intérieur coopératif est un système coopératif.	Le milieu intérieur correspond aux parties du système ainsi qu'aux supports physiques nécessaires à leurs échanges. Un système à milieu intérieur coopératif possède des échanges coopératifs avec son environnement, car ceux-ci sont un sous-ensemble des échanges que réalisent ses parties.
Pour tout système coopératif, il existe au moins un système à milieu intérieur coopératif avec une fonction équivalente dans le même environnement.	La méthode est identique à celle du lemme 2. La particularité réside dans l'objet de la construction qui est maintenant chaque partie du système.

• La notion de rétroaction n'est pas contraignante dans cette théorie car le système doit seulement juger si les changements s'opérant dans le milieu sont coopératifs de son point de vue sans savoir si ces changements sont dépendants de ses propres actions passées.

• À la définition « computationnelle » de l'émergence posée dans le 3.1, nous pouvons aussi ajouter un troisième volet qui précise la technologie AMAS et dont nous montrons dans la suite du chapitre qu'elle se situe bien dans le champ des agents et multi-agents :

Méthode – Dans la technologie AMAS, pour changer la fonction il suffit de changer l'organisation des composants du système. Ces mécanismes sont spécifiés par des règles locales régissant l'auto-organisation entre les composants et ne dépendant pas de la connaissance de la fonction collective.

3.2.2. Adapter le système par ses parties

Spécifier un modèle *a priori* pour un système qui aura à faire face à des imprévus, c'est contraindre (peut être inopportunément) l'espace des possibles. Depuis Bertalanffy, de nombreux auteurs ont étudié des « *systèmes de divers ordres qui ne peuvent s'appréhender que par l'étude de leurs parties prises isolément.* » [Bertalanffy, 1993].

Un moyen d'apprendre pour un système S consiste à transformer sa fonction actuelle fs de manière autonome (donc par auto-organisation) afin de s'adapter à l'environnement, considéré comme une contrainte qui lui est donnée. Chaque partie Pi d'un système S réalise une fonction partielle fpi de la fonction globale fs. fs est le résultat de la combinaison des fonctions partielles fpi, notée par l'opérateur « ⊖ ». La combinaison étant déterminée par l'organisation courante des parties, il s'ensuit que fs = fp1 ⊖ fp2 ⊖ ... ⊖ fpn. Comme généralement fp1 ⊖ fp2 ≠ fp2 ⊖ fp1, transformer l'organisation conduit à changer la combinaison des fonctions partielles et donc à modifier la fonction globale fs, devenant par là même un moyen d'adapter le système à l'environnement (*cf.* figure 1).

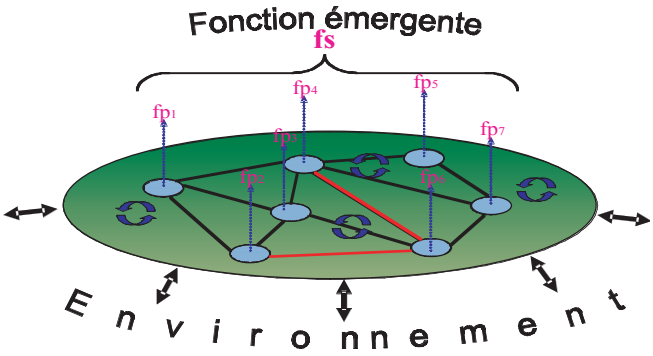

Figure 1. Parties d'un système en interaction dynamique.

Pour un système multi-agent, la mise en œuvre de cette adaptation implique que le concepteur ne s'intéresse qu'à l'agent et lui donne les moyens de décider de manière autonome de changer ses liens avec les autres agents pour tendre vers une organisa-

tion coopérative. Ainsi, en fonction des interactions que développe le système multi-agent avec son environnement, l'organisation entre ses agents émerge et constitue une réponse pour faire face aux imprévus.

3.2.3. Les composants d'un agent AMAS

En elle-même, l'organisation qui émerge est une organisation observable non préétablie par le concepteur du système. Mais ce qui nous intéresse le plus c'est l'émergence de la fonction du système qui est produite par l'organisation entre les agents à un instant donné. Pour réaliser cela, les agents sont programmés pour rechercher en permanence une situation coopérative avec les autres agents du système. Un agent considère ainsi de son point de vue local qu'il doit recevoir les informations appropriées de l'environnement ou d'autres agents pour réaliser sa fonction, et qu'il transmet des informations utiles à d'autres. Si tous les agents du système sont dans cette situation idéale, alors le théorème précédent indique que le système, en tant que collectif, est fonctionnellement adéquat dans son environnement sans qu'aucun agent particulier n'en ait le contrôle. La technologie agent décrite ci-dessous leur confère la possibilité de raisonner et agir sur l'organisation du système à partir de ce qu'ils peuvent considérer localement et à tout instant comme une situation non coopérative.

Un agent doit pouvoir localement décider s'il est en situation coopérative. En effet, par rapport aux croyances qu'il a sur lui-même, il peut localement déterminer si ce qu'il reçoit est compréhensible et si ce qu'il reçoit lui permet de réaliser une action. De la même manière, en fonction de ses perceptions, il pourra juger localement si ses actions ont été utiles. D'une manière générale, cinq parties sont indispensables à un agent coopératif pour qu'un comportement collectif cohérent puisse être observé à partir de l'agrégation de comportements individuels.

Les compétences sont des connaissances d'un domaine particulier qui permettent à l'agent de réaliser la fonction partielle qui lui est assignée. Aucune contrainte technique n'est imposée pour le développement (système de production, méthode objet,...).

La représentation de lui-même, des autres agents et de l'environnement confère à l'agent une croyance sur ce que l'agent sait de lui-même, des autres et de son environnement. Les croyances peuvent être implicites ou explicites.

L'attitude sociale appelée la coopération permet de définir des critères locaux qui vont permettre à l'agent de décider de son comportement et de se réorganiser avec les autres agents c'est-à-dire modifier ses liens avec les autres agents. Elle est au cœur de la théorie.

Les agents ont un langage d'interaction qui leur permet de communiquer soit directement par envoi de messages soit indirectement par l'environnement (*cf.* figure 2).

Les aptitudes sont les capacités qu'un agent possède pour raisonner sur ses représentations et sa connaissance.

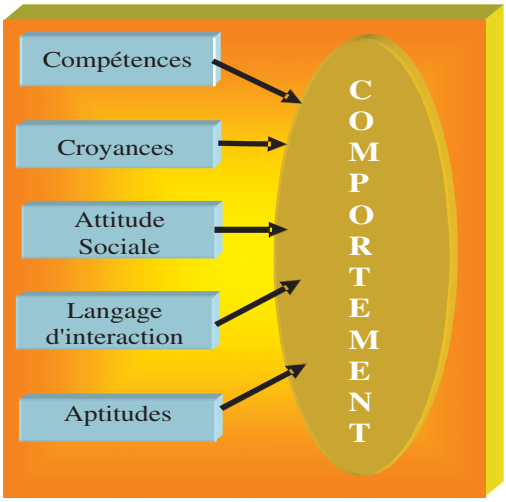

Figure 2. Composants d'un agent.

3.2.4. Les situations de non coopération

L'organisation d'un système est décrite par les liens d'interaction entre agents. Dans la technologie AMAS, un agent autonome considère qu'il a trouvé la bonne place au sein de l'organisation s'il interagit coopérativement avec autrui ; dans le cas contraire, il agira pour chercher une place mieux adaptée. À chaque instant son comportement coopératif (de son point de vue qui est donc toujours subjectif) est guidé par ses compétences et croyances courantes. Les conditions de non coopération conduisant au processus de réorganisation dérivent immédiatement de la définition de la coopération idéale :

$$\text{Non coopération} = \neg\mathbf{C1} \vee \neg\mathbf{C2} \vee \neg\mathbf{C3}$$

¬C1 : Un signal perçu est incompris ou possède de multiples interprétations (ambiguïté). Dans ce cas, un agent coopératif ne va pas ignorer le signal car il le considère nécessaire à l'activité du système. Il va donc tenter de le transmettre à d'autres agents qu'il estime plus compétents ou bien se faire aider par autrui afin de lever les ambiguïtés.

¬C2 : L'information reçue est déjà connue ou n'a aucune conséquence logique. L'agent coopératif ne peut pas tirer profit de cette information pour transformer le monde, il va donc chercher d'autres agents qui pourraient en bénéficier.

¬C3 : Compte tenu de ses croyances courantes, l'agent considère que la transformation de l'environnement qu'il peut opérer n'est pas bénéfique à autrui (il croit qu'il va perturber l'activité d'autres agents). Cette situation englobe les notions de conflit et de concurrence qui sont fréquemment étudiées dans le domaine. Par exemple, un conflit de résultat peut survenir si l'agent aboutit à une conclusion opposée à celle d'un autre. Une concurrence est détectée localement si l'agent aboutit à des conclusions identiques à celles d'un autre.

L'agent réalise en permanence sa fonction partielle, mais il agit simultanément et localement sur l'organisation interne du système s'il détecte des situations non coopératives. La conception

d'un système multi-agent coopératif consiste ainsi en la définition pour chaque composant – les agents – pris isolément de tous les états non coopératifs et les activités associées pour les supprimer. Quand le système est plongé dans un environnement dynamique, l'observateur peut analyser un processus interne au système conduisant à la modification des relations entre les agents. Ainsi, la recomposition des fonctions partielles réalisées par chaque agent amène une transformation de la fonction globale du système et les états non coopératifs dus aux situations imprévues sont progressivement supprimés.

3.3. Le projet STAFF

3.3.1. Objectif du projet

La réalisation de nombreux ouvrages (barrages, digues, ponts,...) comme l'organisation de la vie le long des cours d'eau, impose de connaître les risques naturels liés à la montée des eaux. Néanmoins, le problème de la prévision des crues est complexe et ne peut être réalisé qu'à court terme car il dépend d'un environnement dynamique qui comporte de nombreux paramètres, telles les précipitations qui sont imprévisibles à long terme. D'ailleurs, les systèmes actuellement en place utilisent des modèles mathématiques qui sont différents suivant le lieu auquel on veut effectuer la prévision, car ils sont fondés sur des connaissances hydrologiques spécifiques.

Afin de répondre à ces contraintes de rapidité, d'efficacité et de simplicité d'utilisation, des outils sont indispensables pour l'observation, l'analyse et la prévision. L'objet du projet STAFF (Software Tool for Adaptive Flood Forecast) [Sontheimer, 1999] est focalisé sur le dernier point car il propose de fournir un outil de prévision auto-adaptatif dans tout environnement qui dispose de capteurs pour l'observation des phénomènes naturels. Ce type d'outil générique, permettra d'accroître les capacités de prévision et donc de diminuer les risques encourus par les populations.

Actuellement, STAFF est intégré dans l'environnement d'annonce de crues déjà opérationnel du bassin supérieur de la Garonne. La DIREN[13] dispose d'un environnement de prévision nommé SOPHIE. D'autres sites en France pourraient bénéficier rapidement de cet outil, dans la mesure où SOPHIE va être installé dans de nombreuses régions dans les années à venir.

3.3.2. Descriptif technique de STAFF

À chaque station où doit s'effectuer une prévision est associé un agent (élément du logiciel STAFF) chargé de la réaliser. Il interagit avec tous les autres agents-stations réalisant le même type de travail sur d'autres sites. Le modèle de calcul utilisé est le même pour tous les agents. Il s'agit d'une somme pondérée de toutes les données d'entrée de l'agent (indifféremment, les hauteurs d'eau, les pluies ou les mesures radar) [Sontheimer, 2001].

Figure 3. Architecture de STAFF à deux niveaux de SMA.

Chaque agent-station est lui-même un système complexe à part entière composé d'agents-prévision qui ont pour tâche de déterminer l'évolution de la station considérée pour un intervalle de temps élémentaire (Figure 3). Par exemple, s'il faut réaliser une prévision sur quatre heures, un agent-station sera constitué de quatre agents-prévision. Un feed-back exogène au système est effectué chaque heure lorsque l'agent-prévision à une heure à produit un résultat erroné concernant l'heure précédente. Mais l'application de la théorie AMAS à ce niveau d'agent est essentielle pour obtenir de bons résultats. En effet, l'agent-prévision à quatre heures ne peut recevoir de feed-back que quatre heures après son résultat, ce qui produit comme comportement (vérifié expérimentalement) un retard sur tous les changements de courbure dans une prévision [Georgé, 2004]. Mais l'exploitation de la coopération au sens AMAS le force à obtenir un résultat voisin de celui que l'agent-prévision à trois heures fournira une heure plus tard. Cette coopération endogène au système entre tous les agents horaires est génératrice d'un nombre plus important d'ajustements qui améliorent sensiblement les résultats.

Ce modèle n'est pas linéaire car l'ajustement des poids dépend de leur valeur précédente et de la coopération entre les différents agents. Chaque agent utilise ainsi un modèle de calcul spécifique pour faire sa prévision. L'auto-organisation coopérative entre les agents définit la manière d'ajuster continuellement le modèle relativement aux mesures d'entrée et aux résultats des autres agents. Lorsqu'un agent se trouve en situation non coopérative (relativement aux autres agents ou aux données), il doit agir de manière à aboutir à une situation coopérative. Ici, de par la conception du système, le seul moyen pour un agent d'agir sur sa prévision est de modifier les poids appliqués aux valeurs qui lui sont fournies.

Le système est donc un réseau à deux niveaux d'agents et à connectivité totale lors de l'initialisation du système. Au cours des apprentissages, le réseau va se structurer de manière à considérer seulement les paramètres d'entrée corrélés avec les

13. DIrection Régionale de l'EnNvironnement : organisme en charge des annonces de crues dans le bassin supérieur de la Garonne.

résultats de sortie : à ce stade, le système a trouvé le modèle adéquat. Les agents ne connaissent pas la finalité du système global même s'ils y contribuent directement, ils ont un but qui leur est propre. Ici, les objectifs sont clairement déterminés :

• le système doit fournir les meilleures prévisions sur l'ensemble des stations ;

• les agents doivent fournir la meilleure prévision possible pour la station à laquelle ils appartiennent.

Selon l'approche théorique AMAS, ces objectifs sont obtenus lorsque le système auto-organisateur est en situation coopérative permanente. Comme il ne nécessite aucun paramètre prédéfini, c'est un modèle auto-adaptatif dont l'ajustement est effectué la première fois à l'installation du système sur des historiques de crues.

3.3.3. Résultats

Afin d'étudier le caractère réellement auto-adaptatif de STAFF des traitements ont été effectués sur des échantillons de données réelles [Régis, 2002]. Les tests portent à la fois sur des types de données différents (hauteur d'eau, pluies, mesures radar) et sur des situations hydrologiques très diverses (topographie du terrain, hydrologie,...). Ces échantillons sont donnés par les experts de la DIREN.

La figure 4 est une indication sur la capacité de convergence du modèle. L'abscisse indique le nombre d'apprentissages (240 feed-backs sur 10 jours), tandis que l'ordonnée exprime la hauteur d'eau en mètres. La courbe bleue est la hauteur observée du niveau de la rivière, les courbes en bleu clair et vert clair sont celles de deux modèles de prévision physico-hydrologiques classiques, tandis que la courbe rouge est la prévision donnée par STAFF.

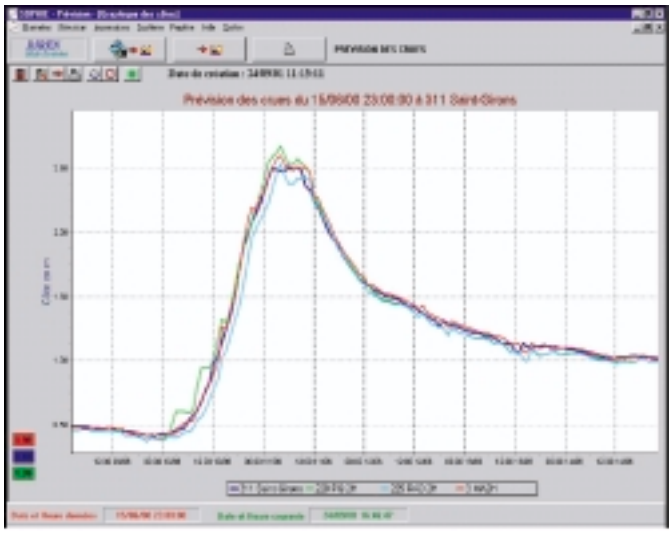

Figure 4. Fonctionnement typique de STAFF.

Le problème habituel d'un système adaptatif est son inaptitude à fournir une réponse optimale par le fait de la convergence vers des attracteurs locaux. Nous observons expérimentalement que la théorie employée ici supprime cet inconvénient, car le système tend vers la fonction souhaitée quel que soit son état initial.

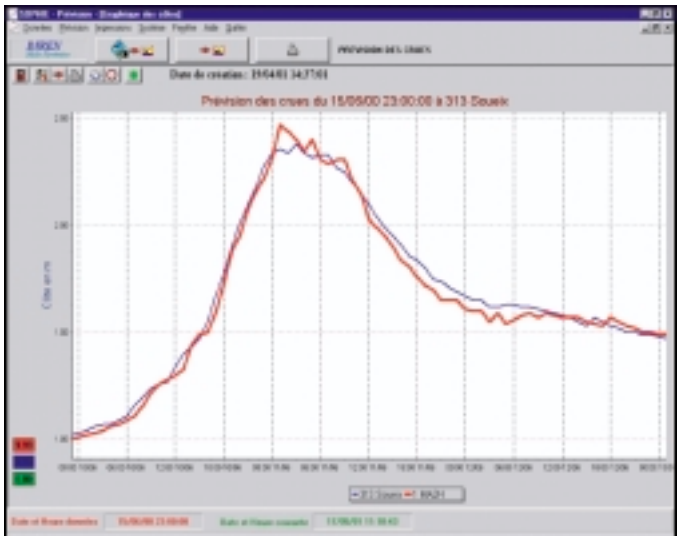

Figure 5. Prévision pour une station amont.

La figure 5 est une prévision sur une rivière très en amont dans les Pyrénées où il ne peut pas exister de modèles physico-hydrologiques, car il ne peut exister aucun capteur sur des rivières amont. STAFF à fait émerger des corrélations avec des bassins adjacents (à l'est et à l'ouest) qui n'ont rien à voir avec de l'hydrologie car les alimentations en eau sont séparées. Toutefois, l'évolution des bassins adjacents est un indicateur d'une évolution similaire pour celui de Soueix. Nous pouvons remarquer que si nous avions intégré des contraintes spatiales (évidentes dans ce domaine) dans la conception de STAFF, ce genre d'émergence de fonction n'aurait jamais pu survenir.

Ce modèle adaptatif de prévision de crues ne nécessite aucun calage préalable à son utilisation. Ainsi, le système sera initialisé au lancement par une seule information : la durée de la prévision demandée (par exemple 3, 6 ou 12 heures) selon les souhaits du prévisionniste et les conditions hydrologiques. Le système recevra ensuite en temps réel (typiquement toutes les heures) les données échantillonnées disponibles, comme nous l'avons indiqué sur les figures 4 et 5. Ces données – issues de côtes ou de pluviomètres – peuvent être traitées isolément ou simultanément par le logiciel STAFF.

3.4. Analyse

Etablir un modèle de prévision hydrologique exige une compétence approfondie du domaine et beaucoup d'expérimentations sur le terrain. Prendre des problèmes par une approche émergente, c'est considérer qu'il est trop difficile pour spécifier par-

faitement l'algorithme de résolution à la conception. Il faut donc laisser de l'autonomie au système pour s'adapter à des imprévus survenant au micro-niveau de ses parties qui sont typiquement en interaction non linéaires. Dans les théories de l'émergence des phénomènes non explicitement traités au micro-niveau doivent apparaître au macro-niveau. Dans le cas de STAFF, des corrélations spatiales et temporelles apparaissent entre des événements, alors que le critère d'adaptation des agents au micro-niveau (qui est la coopération) n'en a aucune connaissance. Le feed-back dans de tels systèmes n'indique pas une relation causale de l'action de l'agent à la perception des changements dans l'environnement, mais simplement l'apparition de nouveaux événements et c'est à l'agent (en coopération avec les autres) à déterminer sa part de causalité. De plus, c'est aussi à l'agent de déterminer la relation temporelle entre ses actions les feed-backs (encore faut-il le doter d'une mémoire appropriée pour avoir une trace de ses actions).

La résolution émergente de problèmes par auto-organisation coopérative rentre dans le cadre de conception de systèmes complexes comme peuvent les modéliser les théories du chaos, des catastrophes, la synergétique... [Goldstein, 1999]. Mais ce sont des questions largement du domaine de la recherche, car il faut encore théoriquement prouver la convergence générale de telles méthodes, qui dépend d'un ensemble d'éléments parmi lesquels :

• La dynamique de l'environnement : si les processus d'auto-organisation du système sont moins rapides que les changements du milieu, le système sera toujours inadéquat ;

• Les compétences des agents : s'il manque des compétences au sein du système aucune auto-organisation ne pourra conduire à la fonction collective qui doit émerger ;

• Les capacités de perception et d'action des agents : par exemple si les agents AMAS ne perçoivent dans leur environnement ce qui serait pour eux une situation non coopérative ou ne pourraient agir avec des effecteurs ou des messages appropriés pour adapter le collectif, le système tendrait toujours vers des solutions sous-optimales.

4. Conclusion

Pour analyser l'intérêt potentiel des systèmes multi-agents dans la maîtrise de systèmes dynamiques constitués de sous-systèmes distribués et autonomes encapsulant une variété de senseurs, nous reprendrons tout d'abord un extrait des conclusions de la conférence COGIS [Appriou, 2003].

« La maîtrise des systèmes complexes, mais également leur architecture et leur gestion est un défi pour l'avenir. Elle demandera le travail en commun de plusieurs communautés : communauté d'informaticiens pour les simulations coopératives et distribuées, communauté de cogniticiens pour les outils de gestion des senseurs et des systèmes et une communauté à créer multidisciplinaire d'architectes des systèmes complexes. La conception des architectures des systèmes complexes devra être établie en considérant conjointement les outils de gestion de ces systèmes ou des senseurs les constituant, à cause des intercorrélations fortes entre la structure du système et le degré d'intelligence de chacun de ses éléments (senseur ou sous-système) ».

D'un point de vue conceptuel, la problématique multi-agent est effectivement celle de systèmes constitués de parties autonomes en interactions, pouvant eux-mêmes être des systèmes multi-agents constitués d'agents de granularité plus fine. Comme nous l'avons montré dans la deuxième partie, les systèmes multi-agents s'intéressent aux trois classes d'application : la simulation permettant d'étudier finement (au niveau de l'agent) la pertinence de différentes stratégies, l'intégration (y compris d'agents humains dans le collectif) et la résolution de problèmes. Les architectures, plates-formes et méthodes intègrent implicitement cette problématique ; même si l'autonomie des agents est parfois contrainte par une forte centralisation. Si la centralisation permet d'augmenter le contrôle du système global, elle peut aussi s'avérer inappropriée (en termes de surcharge des canaux de communication et de résolution sub-optimale) dans des environnements fortement dynamiques.

C'est dans un contexte d'ouverture de ces systèmes que les notions d'auto-organisation et d'émergence – abordées dans la troisième partie – prennent tout leur sens. Il faut être capable de définir des stratégies locales pertinentes, cohérentes avec un objectif global mais dont l'imprévu rend inapproprié les méthodes de résolution précédemment employées. « Quelle que soit l'approche choisie, les agents n'ont pas, pour des raisons soit de complexité soit de structure (distribution), accès à l'état de tous les autres et de tout l'environnement, et donc pas à l'état global qui porte le sens interprétable par l'utilisateur » [Drogoul, 2004]. La troisième partie de l'article s'est focalisée sur une théorie particulière – celle des AMAS – permettant de résoudre des problèmes sans avoir nécessairement à connaître l'objectif global du système car son raisonnement est fondé sur l'auto-observation des activités coopératives de ses parties.

En résumé, l'approche multi-agent est effectivement pertinente pour la thématique abordée dans ce numéro spécial, mais la gestion « intelligente » d'un système d'une réelle complexité exige encore beaucoup de travaux interdisciplinaires tant expérimentaux que théoriques. Sous diverses dénominations : *autonomic computing, pervasive computing, ubiquitous computing, emergent computation, ambient intelligence, amorphous computing*, les industriels s'interrogent sur la capacité à maîtriser les systèmes artificiels du futur. Nous regroupons ces problèmes sous le terme de neo-computation dont les systèmes possèdent les caractéristiques principales suivantes :

• Un grand nombre de composants en interaction,

• Un nombre variable de ces composants durant la vie du système,

• L'impossibilité d'imposer un contrôle centralisé,

• Un environnement évolutif et imprévisible,

• Une tâche collective à effectuer sans pouvoir la spécifier dès la conception du système.

Les questions qui se posent aujourd'hui en traitement du signal rentrent à l'évidence dans le cadre de la néo-computation. Nous considérons que la résolution émergente de problèmes dans un collectif d'entités aux capacités sensori-motrices et cognitives réduites en regard de la fonction globale est une voie intéressante pour les aborder.

5. Références

APPRIOU A. (2003), Conférence COGIS : Commande, Optimisation, Gestion Intelligente & architecture des Senseurs pour les systèmes – Rapport DGA : N° MS 03I018/DSP/SREA/SC/SR – Juin 2003.

ARLABOSSE F., GLEIZES M.-P., OCCELLO M. (2004), Méthodes de conception, Observatoire Français des Techniques Avancées : Systèmes Multi-Agents Série ARAGO 29.

BELLIFEMINE F., POGGI A., RIMASSA G. (2001), Developing multi agent systems with a FIPA-compliant agent framework, Software Practice & Experience, 31:103-128.

BERGENTI F., GLEIZES M.-P., ZAMBONELLI F. (2004), Methodologies and Software Engineering for Agent Systems, Kluwer.

BERNON C., GLEIZES M.-P., PICARD G., and GLIZE P. (2002), The ADELFE Methodology For an Intranet System Design, Fourth International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS-2002), 27-28 May 2002, Toronto (Ontario, Canada) at CAiSE'02.

BERTALANFFY VON L., « Théorie générale des systèmes », Editions Dunod, 1993.

BOISSIER O., GITTON S., GLIZE P. (2004), Caractéristiques des systèmes et des applications Observatoire Français des Techniques Avancées : Systèmes Multi-Agents Série ARAGO 29.

BOND A.H., GASSER L. (1988), Readings in Distributed Artificial Intelligence, Morgan Kaufman.

BROOKS R. A. (1986), A robust control system for a mobile robot. IEEE Journal of Robotics and Automation, 2(1):14-23, 1986.

BRUNNER K. A. (2002), What's Emergent in Emergent Computing?, Proceedings of the EMCSR, E. Trappl editor.

BURA S., DROGOUL A., FERBER J., JACOPIN E. (1991), Eco-résolution : un modèle de résolution de problèmes par interactions RFIA.

BURRAFATO P., COSSENTINO M. (2002), Designing a Multi-Agent Solution for a Bookstore with the PASSI Methodology, AOIS'02 at CAiSE'02, Toronto, May 2002.

CAIRE G., LEAL F., CHAINHO P., EVANS R., GARIJO F., GOMEZ J., PAVON G., KEARNEY P., STARK J. and MASSONET P. (2001), Agent Oriented Analysis using MESSAGE/UML – AOSE 2001.

CAMPS V., GLEIZES M.-P., GLIZE P. (1998), Une théorie des phénomènes globaux fondée sur des interactions locales, Actes des Sixième journées francophones IAD&SMA Pont-à-Mousson, Editions Hermès, Novembre, 1998.

CASTRO J., KOLP M. and MYLOPOULOS J. (2001), À Requirements-driven Development Methodology, In Proceedings of the 13th International Conference on Advanced Information Systems Engineering (CAiSE'01), Stafford, UK – June 2001.

SMITH R.G., (1980) The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. IEEE Transactions on Computers C-29(12).

DEGIRMENCIYAN-CARTAULT I., MARC F., (2002), SCALA : une approche multi-agent pour la conception de systèmes complexes, Journées Francophones sur les Systèmes Multi-Agents, P. Mathieu et J.P. Müller éditeurs, Hermès, 2002.

DELOACH S.A., WOOD M. (2001), Developing Multiagent Systems with agentTool, in Intelligent Agents VII. AgentTheories Architectures and Languages, 7th International Workshop (ATAL 2000, Boston, MA, USA, July 7-9, 2000), C. Castelfranchi, Y. Lesperance (Eds.). LNCS Vol. 1986, Springer Verlag, Berlin, 2001.

DEMAZEAU Y. (2001),VOYELLES, Habilitation à Diriger des recherches, INP Grenoble, 2001.

DROGOUL A., FERRAND N., MÜLLER J.P. (2004), Emergence : de l'articulation du local au global, Observatoire Français des Techniques Avancées : Systèmes Multi-Agents Série ARAGO 29.

ENGELMORE R., MORGAN T. (1988), Blackboard systems. – Addison Wesley 1988.

FERBER J. (1995), Les systèmes multi-agents. Vers une intelligence collective, Editions InterEditions 1995.

FININ T., FRITZSON R., MCKAY D., MCENTIRE R. (1994), « KQML as an Agent Communication Language », Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM'94), 1994.

FIPA (2000), Contract net interaction protocol specification, 2000. Identifier 00029, http://www.pa.org/specs/pa00029/.

FORREST J. (1990), « Emergent computation: self-organizing, collective, and cooperative phenomena in natural and artificial computing networks », Proceedings of the ninth annual CLNS conference, 1990.

GASSER L., BRAGANZA C., HERMAN N. (1987), Implementing Distributed AI Systems Using MACE. Proceeding of the Third IEEE Conference on Artificial Intelligence Applications 1987, (p 315-320).

GEORGÉ J.-P., GLEIZES M.-P., GLIZE P., and RÉGIS C. (2003), Real-time Simulation for Flood Forecast: an Adaptive Multi-Agent System STAFF, Proc. of the AISB'03 symposium on Adaptive Agents and Multi-Agent Systems, Univ. of Wales, Aberystwyth, 2003.

GEORGÉ J.-P., GLEIZES M.-P., GLIZE P. (2003), Conception de systèmes adaptatifs à fonctionnalité émergente : la théorie des AMAS - Revue d'Intelligence Artificielle, 2003.

GEORGÉ J.-P., EDMONDS B., GLIZE P. (2004), Self-organizing adaptivemulti-agent systems work, Chapter 16 in Methodologies and Software Engineering for Agent. The Agent-Oriented Software Engineering handbook, Bergenti F., Gleizes M.-P., Zambonelli F. Editors, Kluwer Publishing.

GLEIZES M.-P., GLIZE P. (1990), Les systèmes multi-experts. Collection technologie de pointe. Editions HERMES 1990.

GOLDSTEIN J. (1999), Emergence as a Construct : History and issues, Emergence Volume 1, Issue 1.

GUESSOUM Z., DOJAT M. (1996), A real-time agent model in an asynchronous object environment, in W Van de Velde et J. Perram (eds) Agent Breaking Away LNAI, Vol 1038, pp. 190-203 Eindhoven Springer Verlag, 1996.

GUTKNECHT O., FERBER J. (1998), À meta-model for the analysis and design of organizations in Multi-Agent Systems, In Demazeau Y., editor, Proceedings of ICMAS'98, Paris, France. IEEE Computer Society, 1998.

HLA 1, High Level Architecture – Livre pédagogique
http://www.mite.org/technology/hla-book

HLA2 Standardisation HLA : DMSO http://dmso.mil SISO
http://siso.sc.ist.ucf.edu/

JACOBSON I., BOOCH G. and RUMBAUGH J. (1999), The Unified Software Development Process, Addison-Wesley, 1999.

KAEBLING L.P., LITTMAN M.L., MOORE A.W. (1996), Reinforcement learning: a Survey, Journal of Artificial Intelligence Research 4, (1996), 237-285.

KINNY D., GEORGEFF M. and RAO A. (1996), À Methodology and Modelling Technique for Systems of BDI Agents, In Van Der Velde, W., Perram, J. (eds.) : Agents Breaking Away: Proc. of the Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World, 1996. MAAMAW'96, (LNAI Vol. 1038). Springer-Verlag: Heidelberg, Germany (1996).

LIND J. (2001), Iterative Software Engineering for Multiagent Systems: the MASSIVE Method, LNCS Vol. 1994, 2001.

MATHIEU P., TAQUET A. (2000), Une forme de négociation pour les systèmes multi-agents Journées francophones IAD-SMA, St Jean de Vêtre, Hermès, 2000.

ODELL J., PARUNAK H.V. and BAUER B. (2000), Extending UML for Agents, In Proceedings of the Agent Oriented Information Systems (AOIS) Workshop at the 17th National Conference on Artificial Intelligence (AAAI), 2000.

PADGHAM L., and WINIKOFF M. (2002), Prometheus : A Pragmatic Methodology for Engineering Intelligent Agents, Workshop on Agent-Oriented Methodologies at OOPSLA 2002.

PIERRE D., BOURDON F., NÉGRET J.-M. (2004), Standards et normes - In journée 29 « Systèmes multi-agents » de l'Observatoire Français des Techniques Avancées - Arago, 2004.

PFEIFER R., SCHEIER C. (2001), Understanding intelligence, The MIT Press, 2001.

RAO A.S., GEORGEFF M. (1991), Modeling Rational Agents within a BDI-Architecture In J. F. Allen, R. Fikes and E. Sandewall (Eds.), Proceedings of the International Conference on Knowledge Representation and Reasoning, Cambridge, MA, PP 473-484 Morgan Kaufmann.

REE (2004), Revue de l'électricité et de l'électronique, Numéro spécial « Exploitation intelligente des senseurs dans les systèmes complexes », ISSN 1265-6534, n° 6/7.

RÉGIS C., SONTHEIMER T., GLEIZES M.-P., GLIZE P. (2002), STAFF : un système multi-agent adaptatif en prévision de crues, 10e Journées Francophones d'Intelligence Artificielle Distribuée et des Systèmes Multi-Agents, Editions Hermès.

SIERRA C. (2002), Autonomous Agents and Multi-Agent Systems, Upgrade : European Online Magazine, Editors: Barber F., Botti V.J., Koehler J. http://www.upgrade-cepis.org, Vol. 3 N°5.

SONTHEIMER T. (1999), Modèle adaptatif de prévision de crues par systèmes multi-agents auto-organisateurs – Rapport de stage Institut Universitaire Professionnalisé – Direction Régionale de l'Environnement Midi-Pyrénées.

SONTHEIMER T., CORNUAU P., VIDAL J.J., GLIZE P. (2001), Application d'un système adaptatif pour la prévision des crues dans le bassin de la Garonne. Conférence SIRNAT'01, Systèmes d'Informations et Risques Naturels.

STONE P., VELOSO M. (2000), Multiagent Systems: A Survey from a Machine Learning Perspective, Autonomous Robots 8, pp345-383, 2000, Kluwer Academic Publishers.

WEISS G., editor (1997), Distributed Artificial Intelligence Meets Machine Learning, Lecture Notes in Artificial Intelligence Volume 1221, Springer-Verlag.

WITTIG T., (1992) ARCHON, an architecture for multi-agent systems-Thies Wittig editor - Ellis Horwood Limited, 1992.

WOLPERT D. H. and MACREADY W. G. (1997), No free lunch theorems for optimization, IEEE Transactions on Evolutionary Computation Vol.1 N°1.

WOOLDRIDGE M., JENNINGS N.R. (1995), Intelligent Agents: Theory and Practice. In Knowledge Engineering Review Vol.10 N°2.

WOOLDRIDGE M., JENNINGS N.R. and KINNY D. (1999), A Methodology for Agent-Oriented Analysis and Design, In Proceedings of the 3rd International Conference on Autonomous Agents (Agents 99), pp 69-76, Seattle, WA, May 1999.

Jean-Pierre **Mano**

Jean-Pierre Mano possède une formation initiale biologique, et plus particulièrement un Master deuxième année en biologie cellulaire et neurobiologie et comportement animal. Il termine actuellement son doctorat sur les systèmes neuronaux ontogéniques artificiels à l'Institut de Recherche en Informatique de Toulouse. Ses centres d'intérêts et d'inspiration sont les systèmes biologiques complexes où l'auto-organisation doit être explicitée. Les théories des systèmes à fonctionnalité émergente qu'il aborde permettent une plasticité neuronale totale et locale : tant au niveau de la fonction de transfert du neurone, que de la régulation de ses entrées, ainsi que de l'apoptose et la prolifération.

Marie-Pierre **Gleizes**

Marie-Pierre Gleizes est maître de conférences à l'Université Paul Sabatier - Toulouse III et effectue ses recherches à l'IRIT (Institut de Recherche en Informatique de Toulouse). Son principal domaine de recherche est la conception de systèmes complexes à fonctionnalité émergente. A cette fin, elle travaille sur les systèmes multi-agents, l'auto-organisation, l'adaptation et l'émergence et s'intéresse au développement d'une méthodologie pour de tels systèmes. Dans ce cadre, elle a participé à plusieurs projets européens. Elle est membre du comité scientifique des groupes de travail d'AgentLink III en 2004 et de plusieurs comités de programme tels que RFIA, JFSMA, MAAMAW, AAMAS, ESAW, SELMAS, COOP, CEEMAS, IAT, MATES... Elle a aussi été membre des comités de pilotage des JFIADSMA de 2000 à 2004, de EUMAS en 2003 et 2004, et d'ESAW en 2005.

Pierre **Glize**

Pierre Glize est ingénieur au CNRS, responsable de l'équipe SMAC (Systèmes Multi-Agents Coopératifs) à l'IRIT (Institut de Recherche en Informatique de Toulouse). Ses travaux de recherches portent sur la conception de systèmes complexes à fonctionnalité émergence. Il s'intéresse à la résolution de problèmes complexes par des systèmes multi-agents auto-organisateurs. Il a une grande expérience des projets avec l'industrie et a réalisé plusieurs transferts de technologie, tels TELEMAC qui est un système multi-expert en médecine et STAFF qui est un système de prévision des crues. Il est le contact du noeud du réseau Agentlink pour l'IRIT et anime le groupe COLLINE (Collectif, Interaction et Emergence) de l'AFIA.