Des systèmes multi-agents Anytime pour la conception de systèmes d'aide à la décision

Claude Duvallet, Bruno Sadeg

Des systèmes multiagents *anytime* pour la conception de systèmes d'aide à la décision

Claude Duvallet — **Bruno Sadeg**

Université du Havre
Laboratoire d'Informatique du Havre
25 rue Philippe Lebon, BP 540
F-76058 Le Havre cedex
{Claude.Duvallet,Bruno.Sadeg}@univ-lehavre.fr

RÉSUMÉ. *Dans un système d'aide à la décision (S.A.D.), un système informatique doit permettre au décideur de prendre la meilleure décision possible, souvent avant une échéance donnée. Nous considérons dans ce papier les S.A.D. qui reposent sur une modélisation multiagent. L'objectif de notre travail est d'étendre la notion de système multiagent à un environnement contraint par le temps, c'est-à-dire de prendre en compte les aspects temps réel des applications d'aide à la décision. La voie que nous avons choisie pour concevoir de tels systèmes est l'utilisation de techniques anytime. Les techniques anytime sont des techniques dont l'objectif est de doter les systèmes informatiques de possibilités de fournir des résultats dont la qualité et la précision évoluent en fonction du temps qui leur est imparti. Dans cet article, nous proposons ANYMAS (ANYtime Multi-Agents System), un modèle de conception de systèmes multiagents temps réel fondé sur des techniques anytime. Nous validons notre travail par la conception d'une application d'aide à la gestion d'un marché électronique.*

ABSTRACT. *In a decision support system, a computer system must allow the decision-maker to take the best decision. Very often, the decision must be taken before a deadline. Among decision-support systems, we consider in this paper, the systems that are based on multiagent paradigm. Therefore, our subject aims to spread the multiagent system notion by taking into account the real-time aspect of decision-support systems. To this purpose, we have choosen the use of anytime algorithms, that permit to get results whose quality and precision evolves according to the allocated time. In this paper, we present ANYMAS (ANYtime Multi-agents System), a design model of real-time multiagents systems based on anytime techniques. We validate the model by the conception of an application for decision support system applied to an electronic market management.*

MOTS-CLÉS : *algorithmes anytime, systèmes multiagents, aide à la décision.*

KEYWORDS: *anytime algorithms, multiagents systems, decision support.*

RSTI - TSI. Volume 23 – n° 8/2004, pages 997 à 1025

1. Introduction

Les systèmes d'information, souvent des Systèmes de Gestion de Bases de Données (SGBD), sont à la base de beaucoup de systèmes d'aide à la décision. Dans chaque système d'information sont stockées toutes les informations liées à un domaine, et l'exploitation judicieuse de l'ensemble des informations contenues dans ces systèmes permet d'aider les utilisateurs dans leur prise de décision. Dans ce travail, nous nous plaçons dans le cadre de systèmes devant fonctionner en temps contraint, c'est-à-dire permettant d'obtenir une réponse à une interrogation avant une échéance donnée, afin de permettre aux décideurs d'agir dans les meilleurs délais et de la façon la plus appropriée possible.

Dans les applications nécessitant des prises de décision, le respect des contraintes de temps est primordial : ce sont des systèmes temps réel. Dans les systèmes temps réel, un résultat correct obtenu hors délai est considéré comme faux [ABB 88]. On dit que la correction des systèmes temps réel dépend non seulement des résultats qu'ils fournissent qui doivent être corrects, mais également de l'instant où ces résultats sont fournis.

Dans cet article, nous avons choisi de fonder nos travaux sur des techniques qui permettent d'obtenir des solutions dégradées ou incomplètes dans les temps plutôt que d'obtenir des solutions plus précises ou complètes mais qui arriveraient trop tard, et qui seraient inexploitables par les décideurs. Pour atteindre cet objectif, les techniques et algorithmes *anytime* nous ont semblé les plus appropriés [ZIL 96, GRA 95], car ils permettent la construction progressive du résultat dont la qualité s'améliore avec l'augmentation du temps alloué.

Nos travaux sont appropriés lors de la conception de systèmes critiques tels que la surveillance de sites industriels classés à très haut risque. Cependant, ils peuvent également s'appliquer dans des domaines où les risques sont essentiellement de nature économique. Pour illustrer et valider nos travaux, nous avons choisi de concevoir une application appartenant à cette deuxième catégorie : la gestion de marché électronique. Nos travaux rejoignent ceux effectués pour la conception d'applications gérant le commerce électronique sur internet. Il s'agit de concevoir un système informatique permettant à des utilisateurs (acheteurs ou vendeurs) répartis sur des sites distants, d'effectuer des transactions commerciales dans les meilleures conditions possibles, c'est-à-dire en ayant la meilleure connaissance possible de l'état du marché (hausses, tendances, stocks, etc.). Cette connaissance est extraite depuis de multiples systèmes d'information, qui gardent en mémoire les transactions effectuées.

La complexité de tels systèmes qui doivent gérer des informations en provenance de différentes sources pouvant se contredire ou se confirmer, ainsi que l'utilisation de systèmes à bases de connaissances, rend adéquate l'utilisation du paradigme *multiagent* [BRI 01, FER 95, WOO 94, WOO 95]. Nous nous fondons pour cela en partie sur les travaux d'Alain Cardon concernant les systèmes multiagents dynamiques [CAR 99]. Dans ce travail, nous nous sommes intéressés à l'étude de l'aspect temps réel dans les systèmes multiagents (SMA). Nous proposons un modèle, appelé ANY-

MAS (ANYtime MultiAgents System), pour la conception d'un système multiagent temps réel (SMATR) qui repose sur l'utilisation des algorithmes *anytime*. Ce modèle permet donc de concevoir des systèmes multiagents ayant des caractéristiques *anytime*.

Dans la section 2, nous présentons les travaux relatifs à la prise en compte de l'aspect temps réel dans les systèmes multiagents et l'exploitation d'informations en temps réel dans des systèmes d'information, notamment dans les bases de données. Dans la section 3, nous décrivons notre modèle de système multiagent *anytime* (ANYMAS) pour la prise en compte du temps réel dans les SMA. Dans la section 4, nous présentons l'application de ce modèle à l'extraction d'informations pour une application de gestion de marché électronique. Dans la section 5, nous concluons cet article en donnant quelques perspectives à ce travail.

2. Travaux relatifs

Le domaine de recherche que nous abordons dans cet article fait intervenir plusieurs domaines habituellement étudiés séparément. Il s'agit tout d'abord du domaine des algorithmes *anytime* (cf. sous-section 2.1) puis des systèmes multiagents temps réel (cf. sous-section 2.2) et celui de l'extraction d'informations en temps réel dans des systèmes d'information (cf. sous-section 2.3). Nous décrivons brièvement par la suite chacun de ces aspects.

2.1. *Systèmes temps réel et algorithmes anytime*

Un système temps réel est un système soumis à des contraintes temporelles d'exécution [STA 88, DUV 99]. En effet, de nombreux systèmes informatiques nécessitent que les tâches qui leur sont confiées, soient exécutées avant certaines échéances. Ce sont des tâches *temps réel*. On a l'habitude de classer celles-ci selon le type d'échéances qu'elles doivent respecter : (1) les tâches temps réel à échéances non strictes : lorsque les échéances ne sont pas respectées, les tâches peuvent continuer à s'exécuter sans que cela ait d'autres conséquences qu'une dégradation de la qualité de service offerte par le système, (2) les tâches temps réel à échéances strictes non critiques : lorsqu'une échéance de ce type n'est pas respectée alors la tâche qui lui est associée devient inutile pour le système et est abandonnée, (3) les tâches temps réel à échéances strictes et critiques : si une échéance de ce type n'est pas respectée alors cela peut entraîner de graves conséquences (économiques, humaines, etc.).

Certains systèmes temps réel doivent être capables de réagir face à leur environnement en procédant par une évaluation de la perception qu'ils ont de celui-ci. Il s'agit par exemple de systèmes permettant de contrôler des robots mobiles. Ces robots doivent se déplacer avec une totale autonomie pour éviter les obstacles, mais aussi effectuer les tâches qui leur sont assignées (par exemple distribuer le courrier). Par exemple, si un objet se présente devant le robot et peut donc constituer un obstacle,

alors il faut que le robot puisse évaluer à temps la nature de l'objet afin d'engager une procédure d'évitement si nécessaire. En voulant connaître la nature exacte de l'objet, le robot pourrait être amené à engager la procédure d'évitement trop tard, alors qu'une vision même floue de l'objet (qualité de l'image réduite) aurait permis au robot de réagir à temps. La solution consiste à construire la perception de l'objet en fonction de l'échéance à respecter pour engager la procédure d'évitement. Le temps nécessaire pour construire une solution donnée n'est pas forcément connu à l'avance et une solution construite progressivement et qui fournirait un résultat quel que soit l'instant où l'échéance se produit est plus que souhaitable. Une nouvelle classe d'algorithmes a permis de donner un début de réponse à ce type de problèmes : les algorithmes *anytime* [ZIL 94, RUS 91, MOU 99].

L'apparition des algorithmes *anytime* date de la fin des années 1980 [HOR 87, DEA 88]. La définition de base d'un algorithme *anytime* est la suivante : un algorithme *anytime* est un algorithme qui permet, en échange d'un temps d'exécution plus long, de fournir des résultats de meilleure qualité, c'est-à-dire plus le temps d'exécution laissé pour effectuer une tâche est grand, meilleure sera la qualité du résultat fourni en sortie.

Dans certains domaines où la complexité des tâches à effectuer ne permet pas de garantir le respect des contraintes temporelles et notamment les échéances des tâches, un résultat imparfait, partiel ou imprécis obtenu dans les temps est souvent préférable à un résultat parfait mais qui arrive trop tard. Un des domaines où les algorithmes *anytime* trouvent particulièrement leur place est celui de l'intelligence artificielle [MOU 93]. Dans ce domaine, un raisonnement qui se construirait de façon progressive au cours du temps permettrait de répondre à certains besoins concernant les échéances à respecter. Dans nos travaux, nous appliquons ce principe à l'extraction d'informations depuis des systèmes d'information. Nous effectuons des requêtes sur des systèmes d'information qui fournissent une information dont la qualité (pertinence et quantité) s'accroît au fur à mesure que le temps alloué à l'exécution augmente.

2.2. *Les systèmes multiagents et le temps réel*

Les SMA temps réel sont des SMA où sont prises en compte les contraintes temps réel. Dans un premier temps, nous présentons le domaine des SMA. Ensuite, nous présentons les travaux relatifs aux SMA temps réel.

2.2.1. *Les systèmes multiagents*

Un système multiagent est constitué de multiples entités autonomes qui interagissent afin de résoudre des problèmes complexes. Ces entités sont appelées agents et possèdent habituellement les propriétés d'autonomie du comportement, de proactivité (capacité à agir sans avoir été directement sollicitées) et d'adaptation aux changements de leur environnement [WOO 94, FER 95].

L'emploi d'agents et de SMA est particulièrement bien adapté à la modélisation de problèmes difficilement modélisables de façon classique à cause de la complexité des interactions présentes. Cependant l'absence de déterminisme dans ces systèmes rend leur conception difficile. Nous utilisons alors une modélisation fondée sur le paradigme multiagent et, dans le cadre de nos travaux, nous prenons en compte l'aspect temps réel [DUV 01]. Dans la section suivante, nous présentons les travaux relatifs aux SMA temps réel.

2.2.2. *Les SMA temps réel*

L'approche *anytime* pour la prise en compte de l'aspect temps réel dans les SMA est la plus souvent utilisée [SAL 98, SAL 97], car elle permet de concevoir des applications qui doivent fournir des réponses dans les temps, même si les résultats fournis sont partiels ou approximatifs. Nous nous intéressons aux applications d'aide à la décision dans le domaine de la gestion de marché.

Des travaux sur les agents *anytime* ont été menés par T. Salvant au cours de sa thèse [SAL 98, SAL 97]. Dans ces travaux, la structure d'un agent *anytime* est décomposée de la façon suivante : un contrôleur de tâches, les tâches du domaine, une horloge temps réel et un allocateur de ressources. Le contrôleur de tâches a deux rôles : (1) veiller à ce que les tâches n'excèdent pas le temps qui leur est alloué et créer de nouvelles tâches, et (2) attendre que de nouvelles tâches en provenance d'autres agents lui soient confiées. Lorsque de nouvelles tâches sont allouées, l'agent contractant qui a effectué la demande de tâche doit pouvoir obtenir un résultat à n'importe quel moment (de façon *anytime*) de l'exécution de la tâche. Les deux derniers composants, l'horloge temps réel et l'allocateur de ressources, sont utilisés durant l'exécution des tâches. L'approche *anytime* de T. Salvant est une approche qui se situe essentiellement au niveau agent et ne tient pas réellement compte du niveau multiagent.

Une autre approche pour la conception d'agents temps réel est celle présentée dans [OCC 94]. L'architecture des agents temps réel est implémentée à base de tableaux noirs [ENG 88]. Elle est présentée avec un processus de contrôle ayant un rôle de planification et de décision. Cette architecture dote l'agent de capacités cognitives et réactives. Néanmoins, une architecture d'agent temps réel fondée sur l'utilisation de tableaux noirs est par trop restrictive par rapport à d'autres architectures d'agents [FER 95, WOO 94] car il s'agit d'une architecture d'agents lourds[1]. Dans ces systèmes, le nombre d'agents est souvent très limité. Cette approche ne convient pas pour nos travaux car nous souhaitons pouvoir concevoir des SMA constitués de multiples agents. Par conséquent, il est souhaitable que ces agents aient une architecture légère [WOO 95].

Dans [OCC 94], il est indiqué qu'il est nécessaire de tenir compte du temps réel tant au niveau microscopique (niveau agent) qu'au niveau macroscopique (niveau système multiagent). Beaucoup de travaux concernent la prise en compte de l'aspect

1. Un agent ayant une architecture lourde peut s'apparenter à un processus dans les systèmes d'exploitation alors qu'un agent léger peut s'apparenter à un *thread*.

temps réel au niveau agent uniquement [SAL 98, SAL 97, HOR 02]. Par exemple, dans [WAG 00, HOR 02], une architecture temps réel non strict est proposée afin de construire des agents très sophistiqués, capables de travailler dans un environnement temps réel possédant des interactions complexes et une variété de méthodes pour accomplir n'importe quelle tâche.

Le travail que nous présentons dans cet article est fondé sur certains de ces travaux. Un aspect particulier de notre travail est l'exploration en temps réel des systèmes d'information. Il sera présenté dans la section suivante.

2.3. *Interrogation en temps réel des systèmes d'information*

Il est nécessaire de contrôler les temps d'exploration des systèmes d'information lorsqu'il s'agit d'aider le décideur dans sa prise de décision. Plusieurs solutions sont envisageables dont l'une consiste à utiliser des Systèmes de Gestion de Bases de données (SGBD) temps réel [RAM 93], qui disposent de mécanismes adéquats de gestion du contrôle de concurrence et d'ordonnancement des transactions temps réel [RAM 93]. L'inconvénient de cette approche réside dans la nécessité de disposer de SGBD temps réel. Or, jusqu'à maintenant, ces outils restent à l'état de prototypes universitaires et ne sont pas réellement exploitables, notamment les SGBD temps réel distribués.

Nous avons choisi d'utiliser une autre approche qui consiste à concevoir un modèle logique du système d'information facile à implémenter sur machine. Ce modèle permet de regrouper à la fois les avantages des SMA, des techniques *anytime* et des systèmes distribués. De cette manière, nous pouvons envisager une exploration à différents niveaux de profondeur ou de précision dans des systèmes d'information situés sur différents sites, en fonction du temps qui est alloué pour cette exploration. L'approche *anytime* nous semble la plus adéquate pour ce type d'applications [MOU 93].

Nos travaux se situent dans un contexte d'applications distribuées. Ces interrogations doivent donc être construites de façon à pouvoir extraire de l'information depuis des bases de données distribuées et de fusionner ces informations extraites qui sont souvent hétérogènes et parfois contradictoires. Dans ce domaine, de nombreux travaux ont été menés [MAC 98, WIL 92] et ont servi de base à la conception de notre modèle.

3. Un modèle de système multiagent temps réel : ANYMAS

Un SMA est composé par nature de multiples entités coopérantes, appelées agents. Les agents d'un SMA interagissent et coopèrent pour résoudre des problèmes complexes. Le schéma d'interaction pour la résolution d'un problème particulier est imprévisible : on ne peut pas connaître à l'avance le nombre d'agents qui vont intervenir, quels vont être leurs types, le nombre de fois qu'ils vont communiquer et s'échanger des messages. Par conséquent, il apparaît très difficile de prévoir les temps d'exécution

nécessaires à la résolution d'un problème. Donc, l'approche fondée sur les ordonnancements temps réel « classiques » [DUV 99] où chaque tâche est tenue de respecter son échéance s'avère inexploitable. Nous allons donc présenter une nouvelle approche pour prendre en compte l'aspect temps réel dans un SMA : le modèle ANYMAS, un modèle de système multiagent fondé sur l'approche *anytime*.

Dans un système multiagent *anytime*, il est en général nécessaire de considérer deux niveaux.

– Le niveau agent (dit niveau local) : il s'agit d'introduire des algorithmes *anytime* au sein des agents afin que leur comportement soit régi par ce type d'algorithmes. À cette fin, nous allons présenter un modèle d'agent *anytime* sur lequel les SMA vont reposer pour acquérir des caractéristiques *anytime*.

– Le niveau SMA (dit niveau global) : il s'agit de contraindre le comportement global du SMA de façon à ce qu'il fournisse des résultats intermédiaires. Nous allons introduire des agents qui seront chargés de coordonner les agents *anytime* suivant des groupes fortement liés. Ces agents que nous appellerons *agents de coordination temporelle* devront être aussi légers que possible afin d'intervenir au minimum dans la charge globale du SMA. Nous touchons là aux propriétés d'émergence présentes généralement dans les systèmes multiagents [DRO 93, FER 95]. Les agents de coordination temporelle sont conçus pour permettre une émergence de propriétés *anytime* au sein du SMA.

La principale différence qu'il y a entre nos travaux (modèle ANYMAS) et ceux déjà effectués sur les SMA temps réel est que, dans le modèle ANYMAS, l'aspect temps réel est intégré dans les deux niveaux précédents (agent et SMA).

Notre objectif avec la conception du modèle ANYMAS est d'enrichir les SMA classiques suffisamment pour qu'ils puissent fournir des résultats dans des temps « contrôlés ». Le modèle ANYMAS est fondé sur les systèmes multiagents classiques [FER 95, WOO 94] auxquels sont intégrés des algorithmes *anytime*. Pour la réalisation d'un prototype implantant ce modèle, nous nous sommes servis d'une plate-forme de développement de SMA minimale à laquelle nous avons intégré l'aspect *anytime*.

3.1. *Fonctionnement général du système*

Nous introduisons dans le modèle ANYMAS un fonctionnement en deux temps :

1) Un mode d'apprentissage : durant cette première phase, l'objectif du système est de se calibrer en mesurant les temps d'exécution et en mettant en place les différentes organisations d'agents nécessaires au bon fonctionnement du système par la suite. Cette phase est surtout nécessaire à l'autoévaluation du système.

2) Un mode de fonctionnement courant ou « normal » : cette seconde phase permet d'exploiter pleinement le système qui aura été calibré durant la première phase. On exploite notamment les mesures des temps d'exécution afin de faire des prédictions sur le temps nécessaire pour obtenir un meilleur résultat à partir du résultat précédent.

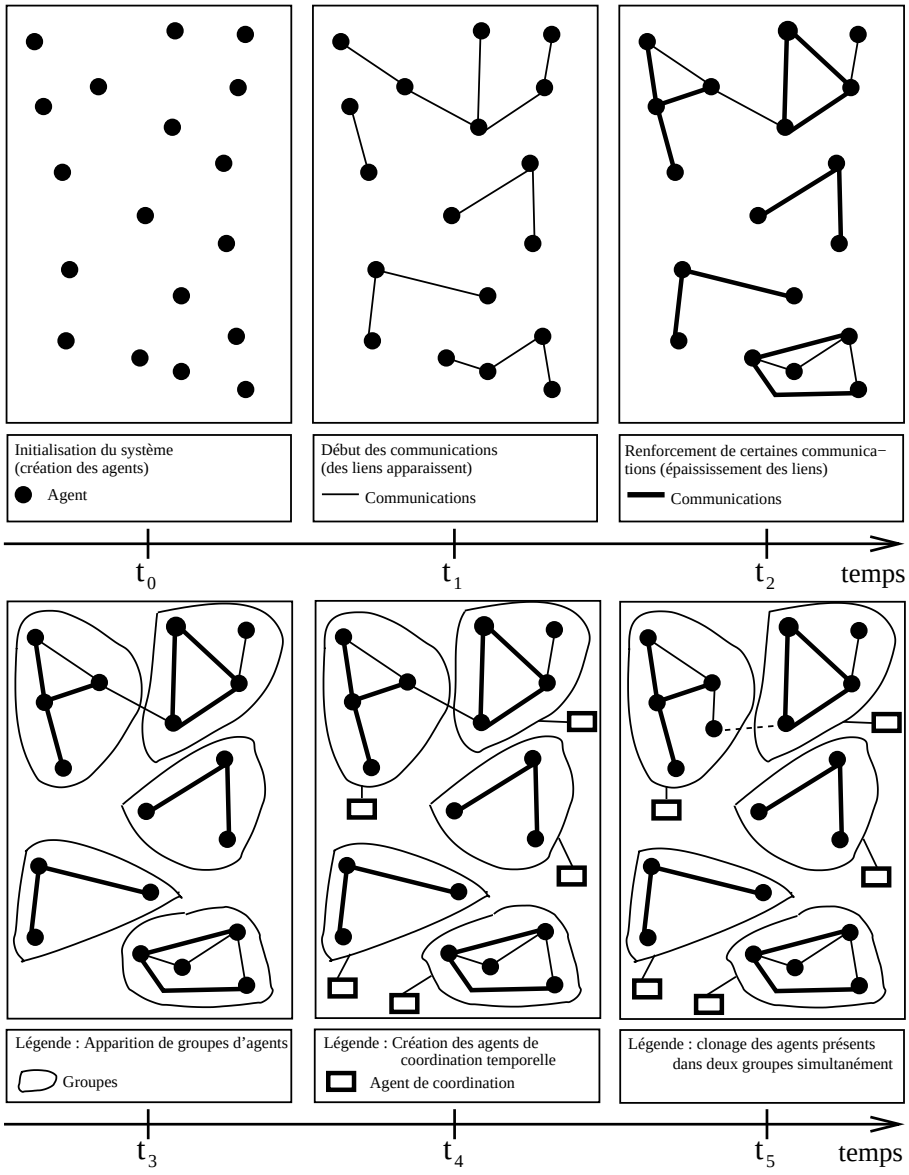

Figure 1. *Naissance des agents de coordination temporelle et réification des groupes à fortes communications*

Lorsqu'un agent travaille, il est amené, pour résoudre un problème, à faire appel à d'autres agents. Lorsque l'on considère des agents *anytime*, un agent qui sollicite un autre agent peut obtenir en réponse un résultat dont la qualité n'est pas optimale. On va alors utiliser la propriété de composition des algorithmes *anytime* [GRA 95], qui consiste à construire des résultats qui sont indirectement fonction des éléments fournis en entrée.

Lors du processus de résolution d'un problème, on voit intervenir des phénomènes tels que celui décrit précédemment et d'autres liés au fonctionnement des agents de coordination temporelle. Les agents de coordination temporelle seront rattachés à de petits groupes d'agents très fortement liés par leurs communications et qui sont donc soumis très fortement au phénomène de composition d'algorithmes *anytime*.

La formation des groupes d'agents aboutit à la naissance des agents de coordination temporelle. Elle va s'effectuer durant la phase d'apprentissage du système mais on peut imaginer que des changements interviendront au-delà de la phase d'apprentissage. La naissance des agents de coordination temporelle est schématisée dans la figure 1.

3.2. *Réification des groupes d'agents*

La réification des groupes d'agents telle qu'elle est présentée dans la figure 1 est fondée sur l'hypothèse suivante : un groupe d'agents qui travaille autour d'une même granularité de connaissance possède des relations fortes qui se traduisent par des communications interagents.

À l'initialisation du système, des agents sont créés et commencent à travailler pour atteindre leur objectif. Au cours de leur activité, les agents sont sollicités ou sollicitent d'autres agents au moyen d'échanges de messages (communications). Dans le modèle ANYMAS, nous mesurons ces communications (taux de communications en fonction du temps écoulé) et lorsque celles-ci deviennent nombreuses (épaississement des traits sur la figure 1) entre deux agents ou tout un groupe d'agents, on va décider de réifier un nouveau groupe et lui adjoindre un agent de coordination temporelle (cf. section 3.6). Un agent peut rejoindre ou quitter un groupe à tout moment de sa vie. Pour connaître le moment où il est nécessaire de réifier un groupe d'agents, on utilise une fonction de seuil paramétrée par le concepteur de l'application au sein de chaque agent. Ce paramètre peut être le même pour l'ensemble ou une partie des agents du système.

Lorsqu'un agent se retrouve dans deux ou plusieurs groupes, il peut décider de se cloner et ainsi se spécialiser par rapport à son rôle dans chaque groupe. Dans la version actuelle, pour réifier un agent de coordination temporelle, le système utilise comme critère de réification uniquement le nombre de communications. Il serait intéressant dans une future version de prendre en compte d'autres critères comme l'importance des communications et leurs poids éventuels.

3.3. *Les agents anytime*

Les agents *anytime* permettent d'obtenir des résultats partiels au niveau agent. Dans un agent *anytime*, nous appelons célérité d'un agent sa capacité et sa rapidité à construire un résultat. Cette notion de célérité permet aussi de mesurer les performances d'un agent [DUR 99].

Dans le modèle ANYMAS, un ATN (Augmented Transition Network) est utilisé pour stabiliser les agents dans différents états. La notion d'ATN est héritée de l'architecture d'agents utilisée dans la plate-forme DIMA [GUE 97] mais aussi dans d'autres architectures d'agent.

Comme nous l'avons déjà évoqué, nous considérons deux modes d'exécution : le mode d'apprentissage et le mode de fonctionnement normal. Durant le mode d'apprentissage, la vitesse de l'agent est constamment recalculée afin d'obtenir une valeur moyenne de la vitesse d'exécution. En mode réel d'exécution, cette valeur est utilisée afin d'estimer le temps nécessaire pour exécuter la prochaine tâche. Ces différentes valeurs moyennes mesurées lors du passage d'un état à l'autre de l'ATN sont alors discrétisées en une unité de temps que l'on appelle ΔT, à la fin du mode d'apprentissage.

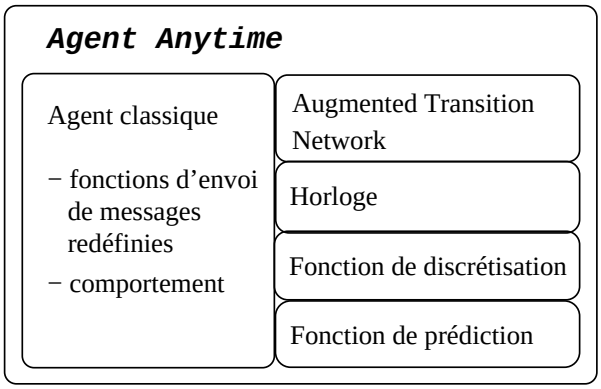

Figure 2. *Agent anytime*

Pour le fonctionnement d'un agent *anytime* (voir figure 2), les composants suivants sont utilisés :

– un ATN, qui permet de contrôler le comportement d'un agent durant sa phase active,

– une horloge temps réel, qui mesure le temps écoulé entre chaque état franchi de l'ATN,

– un module de réification des agents de coordination temporelle, dont nous parlerons plus en détails dans la section 3.6,

– une fonction de discrétisation du temps écoulé entre deux états de l'ATN,

– une fonction de prédiction du temps nécessaire au franchissement d'un état dans l'ATN à partir de l'état courant. Elle est appelée « fonction de prédiction temporelle ».

Nous rajoutons à cet ensemble de composants, un composant utilisé pour la constitution d'un réseau d'accointances dans chaque agent *anytime*.

3.3.1. *L'ATN*

L'utilisation d'un ATN permet de modéliser le comportement de l'agent *anytime* et de guider son comportement temporel. Chaque franchissement d'état dans l'ATN correspond à l'obtention d'un résultat dont la qualité est meilleure que celle obtenue à l'état précédent.

Cet ATN va nous permettre de stocker des informations sur le comportement temporel de l'agent *anytime* telles que :

– les temps minimal, moyen et maximal d'exécution d'une tâche située entre deux états de l'ATN,

– le nombre de communications échangées,

– le nombre d'exécutions effectuées entre deux états de l'ATN.

Toutes ces informations sont nécessaires au bon fonctionnement de la fonction de discrétisation et de la fonction de prédiction temporelle. L'ATN est intégré dans l'agent et sert alors de base pour la modélisation du comportement de l'agent. Ce comportement est construit suivant les techniques *anytime*. Il est en effet nécessaire de mettre des points de contrôle au sein de l'algorithme qui serviront d'états de transition pour l'ATN. Ces points de contrôle permettent de fournir des résultats de qualité dégradée si le temps requis pour obtenir le résultat complet est insuffisant. De nombreux travaux utilisent des ATN pour modéliser le comportement des agents. Cependant, ce type d'ATN n'est pas très approprié à notre modèle qui doit être couplé avec des techniques *anytime*, pour y intégrer l'aspect temps réel. Les deux composants sont intrinsèquement liés pour la conception de l'agent *anytime*. De plus, contrairement à la plupart des travaux dont le comportement des agents est modélisé par un ATN, les états représentent dans notre cas la possibilité pour l'agent de fournir un résultat de qualité dégradée (une solution intermédiaire).

En outre, on peut voir l'utilisation d'un ATN comme un guidage temporel de l'agent, c'est-à-dire qu'il permet de prendre une décision selon qu'on aura ou pas le temps d'effectuer la prochaine action. Dans le cadre d'un ATN « complexe » tel que celui présenté dans la figure 7, par opposition à l'ATN linéaire de la figure 3, certains franchissements d'états vont se résumer à prendre un chemin ou un autre en fonction du temps nécessaire pour atteindre le prochain état et de la qualité du résultat obtenu. Dans un ATN linéaire, l'agent aura deux possibilités en fonction du temps qui lui est alloué : continuer ou suspendre son activité.

3.3.2. *Algorithmes anytime vs design-to-time*

Dans le modèle ANYMAS, un ATN est initialisé pour modéliser le comportement d'un agent selon un algorithme *anytime* lors de sa conception. Lorsque l'ATN est non linéaire comme dans l'exemple de la figure 7, il se produit alors des choix dans les chemins d'exécution menant à une solution. On pourrait alors considérer que l'algorithme est de type *design-to-time* [GAR 93], c'est-à-dire un algorithme dans lequel on utilise un ensemble de méthodes pour lesquelles on connaît la durée d'exécution et la qualité du résultat produit. Lors de l'exécution de l'algorithme, des choix sont effectués parmi cet ensemble de méthodes en fonction du temps disponible afin d'obtenir la meilleure solution possible.

Bien que notre solution soit très proche de cette définition, nous considérons que la réévaluation dynamique des temps d'exécution lors du fonctionnement de nos algorithmes n'est pas tout à fait conforme à la définition d'un algorithme *design-to-time* qui se fonde sur des temps discrets d'exécution connus au préalable. On pourrait considérer que nous nous fondons sur des algorithmes *anytime* qui, dans certains cas, possèdent un comportement d'algorithmes *design-to-time*.

3.3.3. *Le réseau d'accointances*

Dans tout système multiagent, les agents possèdent des capacités de communication qui peuvent prendre différentes formes. Il peut s'agir de communications indirectes par émission de signaux telles qu'on peut les trouver dans le projet Manta d'Alexis Drogoul [DRO 93] ou de communications directes comme c'est beaucoup plus souvent le cas [FER 98].

Les agents avec lesquels un agent est amené à communiquer forment son réseau d'accointances, c'est-à-dire son réseau de « connaissances ». Le réseau d'accointances n'est pas obligatoirement réifié dans les agents. Nous avons notamment repris le cas du fonctionnement du modèle Aalaadin [FER 98] et plus techniquement de la plateforme MadKit qui implémente ce modèle [GUT 97].

En nous fondant sur le modèle Aalaadin, nous avons réifié ce réseau d'accointances afin de réduire au strict nécessaire le nombre d'agents avec lesquels un agent peut communiquer. De plus, la réification de ce réseau d'accointances va nous permettre de stocker des informations sur le nombre de communications effectuées entre les agents. Ces informations seront essentielles pour le module de réification des agents de coordination temporelle.

3.4. *La fonction de discrétisation*

L'objectif de cette fonction est de trouver une unité de temps qui puisse servir à mesurer le temps écoulé pour franchir deux états de l'ATN et qui soit plus grande que l'unité de temps de base (qui est en général la milliseconde). Cette unité de temps est

ensuite utilisée dans la fonction de prédiction temporelle afin d'effectuer des estimations sur les temps nécessaires au franchissement des états.

La discrétisation va se faire à partir des informations temporelles stockées sur les transitions de l'ATN. Pour expliquer le fonctionnement du système, nous allons nous fonder sur un ATN simplifié (linéaire). Cette discrétisation va se faire de la façon suivante.

Étant donné un ATN avec n états (par exemple l'ATN schématisé dans la figure 3, où n est égal à 5), $d_{i,i+1}$ ($1 \leq$ i <n-1) est la transition entre deux états i et i+1. Soit une variable ΔT représentant une sous-division du temps et $n_{i,i+1}$ le nombre de ΔT sous-divisions entre deux états i and i+1.

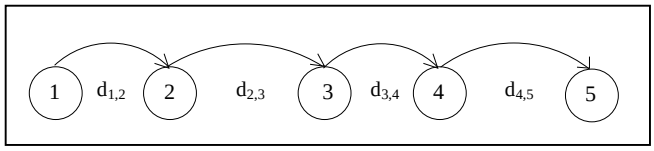

Figure 3. *Exemple d'ATN linéaire*

Alors, on obtient l'équation suivante :

$$d_{i,i+1} = \Delta T * n_{i,i+1}, \text{ où i=1,2,3,...} \Rightarrow \Delta T = \frac{d_{i,i+1}}{n_{i,i+1}}$$

Dans l'exemple de la figure 3, nous obtenons :

$$\Delta T = \frac{d_{1,2}}{n_{1,2}} = \frac{d_{2,3}}{n_{2,3}} = \frac{d_{3,4}}{n_{3,4}} = \frac{d_{4,5}}{n_{4,5}}$$

Le nombre de sous-divisions pour le premier intervalle est initialement fixé. Puis un algorithme (voir figure 5) nous permet de déterminer (1) la valeur de la variable ΔT (unité de temps sur l'ATN) et (2) le nombre de divisions entre deux états de l'ATN. La variable ε permet d'allouer la précision du calcul effectué pour déterminer la valeur de ΔT.

Après application de l'algorithme, nous obtenons l'ATN représenté sur la figure 4. L'algorithme présenté ici (figure 5) a été implémenté en langage JAVA et une simulation de cet algorithme est présentée afin d'illustrer son fonctionnement (figure 6). Dans la première et la deuxième colonne sont représentés les états successifs de l'ATN. La troisième colonne représente le temps moyen d'exécution pour passer d'un état au suivant. La quatrième colonne représente les résultats obtenus après l'exécution de l'algorithme. Dans notre exemple, pour une valeur ε = 5, la valeur calculée de ΔT est 4.

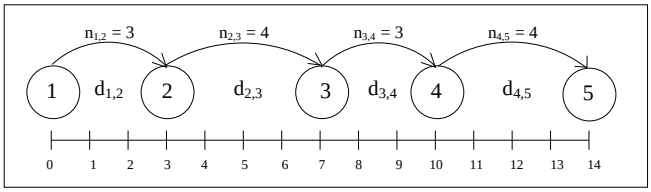

Figure 4. *Exemple de discrétisation du temps sur un ATN linéaire*

En entrée : $d_{i,i+1}$ le temps écoulé entre deux états de l'ATN
N le nombre d'états dans l'ATN
En sortie : ΔT la nouvelle unité de temps calculée
$n_{i,i+1}$ le nombre de ΔT entre les états i et i+1
DÉBUT
$i \leftarrow 1$
$n_{i,i+1} \leftarrow 1$
ΔT = arrondi ($d_{i,i+1}$ / $n_{i,i+1}$)
TANT QUE ((| $d_{i,i+1}$ - ΔT * $n_{i,i+1}$ |) > ε)
ET ((i+1) $\neq$ Dernier Etat))
i $\leftarrow$ 1
$n_{1,2} \leftarrow n_{1,2}$ + 1
ΔT = arrondi ($d_{i,i+1}$ / $n_{i,i+1}$)
TANT QUE (|$d_{i,i+1}$ - ΔT * $n_{i,i+1}$|) < ε
ET (i+1) $\neq$ Dernier Etat))
i $\leftarrow$ i + 1
$d_{i,i+1} = d_{i,i+1} + d_{i-1,i}$ - ΔT * $n_{i-1,i}$
$n_{i,i+1}$ = arrondi ($d_{i,i+1}$ / ΔT)
FIN TANT QUE
FIN TANT QUE
FIN

Figure 5. *Algorithme de discrétisation du temps sur un ATN linéaire*

En résumé, fondé sur une estimation moyenne des temps d'exécution, le but de cet algorithme est de permettre à l'agent de disposer des outils nécessaires à l'estimation de sa célérité et de déterminer ainsi s'il disposera de suffisamment de temps pour exécuter sa tâche avant l'échéance qui lui est fixée. Ceci permet de mettre en œuvre l'aspect prédictif du modèle ANYMAS.

Nous allons maintenant étendre l'algorithme précédent de façon à pouvoir l'appliquer à des ATN plus complexes (voir figure 7). En effet, les ATN que l'on est amené à utiliser sont rarement linéaires.

Etat i	Etat i+1	Temps en ms	Temps en nombre de ΔT
1	2	10556 ms	2640 ΔT
2	3	20173 ms	5043 ΔT
3	4	5443 ms	1361 ΔT
4	5	34256 ms	8564 ΔT
5	6	29654 ms	7413 ΔT
6	7	32432 ms	8108 ΔT
7	8	11284 ms	2821 ΔT
8	9	27236 ms	6809 ΔT
Valeur de ε : 5			
Valeur calculée de ΔT : 4			

Figure 6. *Simulation de l'algorithme*

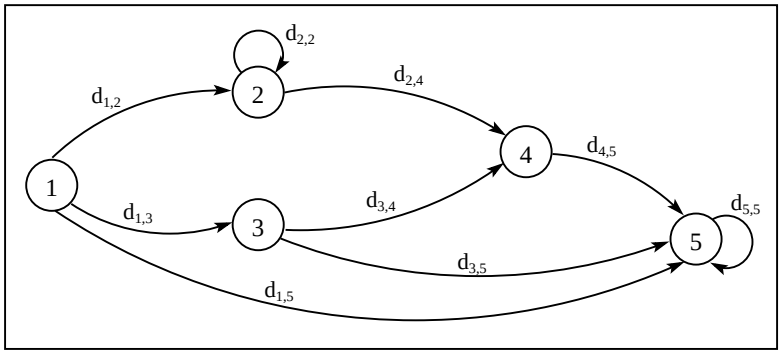

Figure 7. *Un ATN complexe avec des temps d'exécution*

Il s'agit de discrétiser le temps sur un ATN assimilable à un graphe. Nous allons utiliser le même principe :

1) commencer par initialiser le premier nombre d'unités de temps pour la première transition à *un* ;

2) calculer la première valeur de temps discret (ΔT) ;

3) continuer selon le même principe que pour l'algorithme précédent, mais cette fois le processus est récursif.

On obtient alors l'algorithme présenté dans la figure 8.

```
En entrée : d_{i,j} le temps écoulé entre les états de l'ATN
            ε l'erreur tolérée
En sortie : ΔT la nouvelle unité de temps calculée
            n_{i,j} le nombre de ΔT entre les états i et j

DÉBUT
   FirstState ← Etat initial de l'ATN
   CurrentState ← Premier successeur de 'FirstState'
   n_{FirstState,CurrentState} ← 1
   ΔT ← d_{FirstState,CurrentState} / n_{FirstState,CurrentState}
   TANT QUE CalculerAPartir (FisrtState, 0) est FAUX FAIRE
               n_{FirstState,CurrentState} ← n_{FirstState,CurrentState} + 1
               ΔT ← d_{FirstState,CurrentState} / n_{FirstState,CurrentState}
   FIN TANT QUE
FIN

FONCTION CalculerAPartir (state, error) :
Résultat : une valeur booléenne

DÉBUT
   POUR TOUT succ successeur de state FAIRE
         trans ← la transition entre state et succ
         SI trans n'a pas été visité ALORS
            Mettre trans à visiter
            time ← d_{state,succ} + error
            n ← arrondi (time / ΔT)
            error ← time - ΔT * n
            SI error > ε RETOURNER FAUX FINSI
            SI CalculerAPartir (succ, error) est FAUX
               RETOURNER FAUX
            FINSI
         FIN SI
   FIN POUR
   RETOURNER VRAI
FIN
```

Figure 8. *Algorithme de discrétisation du temps sur un ATN complexe*

3.5. *La fonction de prédiction temporelle*

Dans un système multiagent *anytime*, l'objectif est de fournir des résultats dont la qualité varie avec le temps alloué. Pour mettre en œuvre ce principe, nous avons choisi d'influencer le fonctionnement des groupes d'agents en en privilégiant certains au détriment d'autres. Ceci est partiellement fait grâce aux agents de coordination temporelle que nous allons présenter dans la section 3.6. Des critères objectifs sont

nécessaires pour pouvoir influencer le fonctionnement des groupes d'agents. Parmi ces critères, nous prenons en compte la rapidité des agents à fournir un résultat de qualité.

Pour cela, il est nécessaire de connaître avec quelle rapidité un agent va fournir un résultat. Par conséquent, il faut pouvoir estimer les temps d'exécution nécessaires pour fournir un résultat. Pour répondre à ce besoin, nous introduisons au sein de l'agent une fonction de prédiction temporelle. Cette fonction permet de « prédire » le temps d'exécution nécessaire au franchissement du prochain état de l'ATN à partir de l'état courant. Cette estimation se fait par rapport au temps d'exécution nécessaire au franchissement des états précédents et par rapport au temps d'exécution discrétisé qui est habituellement nécessaire pour franchir l'état à atteindre.

En effet, l'ATN étant dans un état x, on souhaite connaître le temps d'exécution nécessaire pour aller jusqu'à l'état y. L'algorithme de la figure 8 décrit la manière de calculer le temps nécessaire pour que l'ATN passe de l'état courant x à l'état y. Les valeurs des paramètres dont dispose l'algorithme sont la durée moyenne qu'a nécessité par le passé ce franchissement et la durée courante d'exécution.

3.6. *Les agents de coordination temporelle*

Les agents de coordination temporelle sont la seconde composante du modèle ANYMAS.

La réification d'agents de coordination temporelle s'effectue au moyen d'un module inséré dans l'agent *anytime*. Ce module agit comme une fonction de seuil. Il déclenche la création d'agents de coordination temporelle lorsque le nombre de communications échangées devient trop important et risque de surcharger le système. Ce seuil permet de contrôler le nombre de communications échangées : lorsque le seuil (en nombre de messages échangés) est atteint par agent, les agents concernés par les communications doivent se synchroniser afin de lancer la réification d'un seul agent de coordination temporelle. Un groupe d'agents fortement liés par leurs communications se crée alors autour de ce nouvel agent. D'autres agents pourront adhérer à ce groupe s'ils échangent un nombre important (seuil décrit précédemment) de messages avec les agents de ce groupe. Pour un agent donné, lorsque sa fonction de seuil lui indique que ses communications ont dépassé ce seuil, il a deux possibilités : (1) adhérer à un groupe existant si dans son réseau d'accointances (les agents avec qui il communique) un ou plusieurs agents se sont déjà regroupés autour d'un agent de coordination temporelle, (2) ou réifier un nouvel agent de coordination temporelle et fonder ainsi un nouveau groupe.

L'agent de coordination temporelle est un agent léger [WOO 94] qui scrute en permanence l'état d'un petit groupe d'agents et contrôle leurs communications. Ce contrôle s'effectue en demandant aux agents des informations sur leur célérité et le nombre de communications échangées. Cet agent doit tenir compte de l'échéance globale du système et doit coopérer pour cela avec d'autres agents de coordination tem-

porelle afin de négocier l'importance relative de son groupe. Ceci se fait au moyen de priorités fixées sur les tâches effectuées par les agents. La réduction de l'activité de certains groupes permet d'augmenter les performances d'autres groupes afin qu'ils convergent plus rapidement vers une solution. De cette façon, une solution partielle et progressive peut être construite.

Les agents de coordination temporelle tels qu'ils sont décrits ici constituent un cadre conceptuel pour la conception d'applications selon le modèle ANYMAS. Chaque application conçue se verra ainsi dotée d'agents de ce type suivant un schéma et un fonctionnement propres au domaine d'application considéré ainsi qu'aux granularités de connaissances qui pourront être extraites. Il sera par exemple à la charge du concepteur de fixer les seuils permettant de déclencher la réification de groupes d'agents ou leur destruction.

4. Une application de gestion de marché

Dans la gestion de marchandises en ville, il s'agit de passer des commandes à des fournisseurs qui doivent ensuite livrer les marchandises à leur client. Il s'agit d'un problème très difficile à résoudre. Dû au fonctionnement en flux tendu (un minimum de stocks) du système, les transactions doivent être effectuées dans des délais « raisonnables ». De plus, les clients peuvent avoir recours à différents fournisseurs pour un même produit et doivent par conséquent choisir celui qui pratique le meilleur rapport entre le prix et les délais de livraison.

La première phase de notre étude concerne la commande des produits et le respect des délais de livraison. La seconde phase, qui concerne la gestion du transport de ces marchandises en ville, n'est pas étudiée dans cet article.

L'objectif est de concevoir un système d'aide à la décision permettant à des clients d'obtenir des informations sur les produits, leur prix et les délais de livraison puis de passer des commandes en se basant sur les informations ainsi obtenues. Il est alors nécessaire d'avoir recours à un ensemble de systèmes d'information distants contenant les données disponibles sur les fournisseurs, les produits, les clients, etc. L'approche SMA est choisie pour concevoir ce système d'aide à la décision.

Pour la réalisation du système, nous nous sommes appuyés sur la plate-forme de développement de SMA, MadKit[2] et l'ORB ORBacus[3] que nous avons dotés de caractéristiques temps réel et plus particulièrement *anytime*. De plus, l'utilisation de CORBA au travers de l'ORB ORBacus nous a permis de doter la plate-forme MadKit d'un mécanisme de communication entre agents physiquement distribués fondé sur une distribution statique des agents [DUV 01]. Les systèmes d'information sont implantés sous forme de bases de données Oracle. Il s'agit de modéliser et de concevoir un système informatique qui permette à des utilisateurs d'interroger ces systèmes

2. http ://www.madkit.org/
3. http :// www.orbacus.com/ob/

d'information (SI) afin de prendre les meilleures décisions possibles concernant la gestion des marchandises en fonction des résultats des requêtes. Pour expliquer notre démarche, nous procédons en deux temps :

1) Nous présentons le système de gestion de marché (section 4.1).

2) Puis, nous modélisons le système suivant le modèle ANYMAS (section 4.2).

4.1. *Présentation générale*

Ce système d'aide à la décision repose en grande partie sur des systèmes d'information distants (voir section 4.1.1) où il doit effectuer des interrogations. Le système est multi-utilisateur et permet l'accès aux systèmes d'information distants. Il devra donc être « distribuable » sur plusieurs sites (voir section 4.1.3). Dans la suite de cette section, nous présentons un prototype que nous avons développé pour valider nos travaux et qui consiste en une application de gestion de marché en milieu urbain (voir section 4.1.4). Nous allons maintenant décrire en détails les différents aspects de cette application.

4.1.1. *Conception du système d'information*

Il s'agit de modéliser et de réaliser un système d'information (SI) contenant l'ensemble des informations manipulées dans la gestion de marchandises. La réalisation sera effectuée au moyen du SGBDR Oracle. Les informations contenues dans les SI sont relatives aux clients, aux fournisseurs, aux produits et aux commandes passées (délais de livraison, respect de ces délais par les fournisseurs, prix pratiqués, cours du marché, etc.)

Certaines informations disponibles sont de caractère privé, c'est-à-dire qu'elles sont disponibles uniquement en local pour un utilisateur (il s'agit par exemple de la liste des commandes passées par un client), et d'autres sont de caractère public (un fournisseur met à disposition les produits qu'il est susceptible de fournir accompagnés de tous les renseignements nécessaires).

4.1.2. *Interrogation des systèmes d'information par des agents*

Les fonctionnalités principales considérées dans le système concernent l'interrogation des systèmes d'information afin de répondre aux requêtes des utilisateurs. Les réponses ainsi obtenues leur permettent de prendre la meilleure décision possible concernant des choix à effectuer (par exemple passer une commande à un fournisseur suivant différents critères : prix pratiqués, délais, respect des délais, etc.). On obtient le schéma d'interaction représenté dans la figure 9.

Une même question permet à un utilisateur d'obtenir une réponse plus ou moins précise suivant le temps dont il dispose pour prendre une décision. Pour mettre en œuvre ce principe, nous nous appuyons sur le modèle ANYMAS.

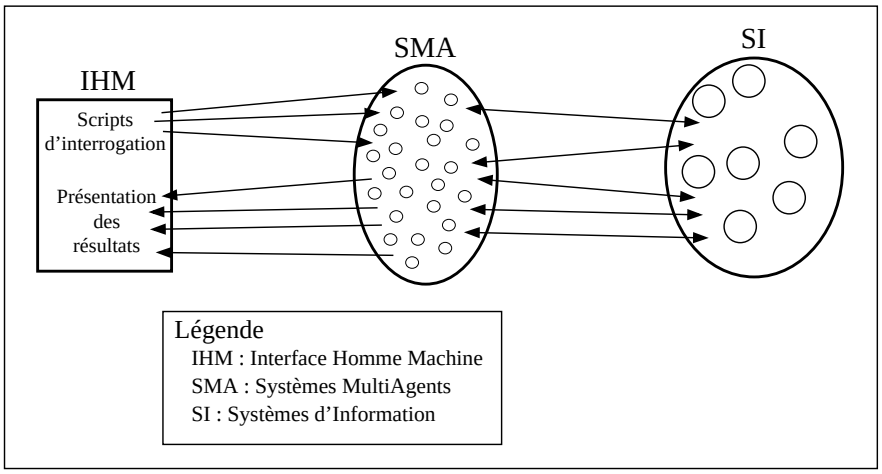

Figure 9. *Schéma d'interaction de l'application entre l'utilisateur et le système*

4.1.3. *Conception d'un système distribué*

Le principe de conception du système de gestion de marchandises est le suivant. Des clients souhaitent passer des commandes auprès d'un ensemble de fournisseurs. L'ensemble des acteurs (clients et fournisseurs) sont physiquement distribués. Pour mettre en œuvre cet aspect distribué, nous nous sommes appuyés sur le modèle DISMAS [DUV 01].

4.1.4. *Réalisation d'un prototype*

Ce système (voir figure 10) est implémenté en langage Java. Pour la conception des systèmes d'information (SI), nous avons utilisé les services offerts par le SGBD Oracle. Les SI sont matérialisés donc sous forme de bases de données relationnelles Oracle. Pour la partie multiagent, nous avons utilisé la plate-forme de développement MadKit, que nous avons étendue avec des techniques *anytime* (ANYMAS) et par des composants de gestion de l'aspect distribué (DISMAS [DUV 01]).

4.2. *Modélisation de l'application*

Dans cette section, nous allons présenter les différents éléments qui composent le modèle de notre système de gestion de marché : les acteurs du système, les objets manipulés et les requêtes qui pourraient lui être soumises.

4.2.1. *Les acteurs du système de gestion de marché*

Dans l'application, il existe deux types d'acteurs : les clients et les fournisseurs. Nous allons donner une modélisation de ces acteurs.

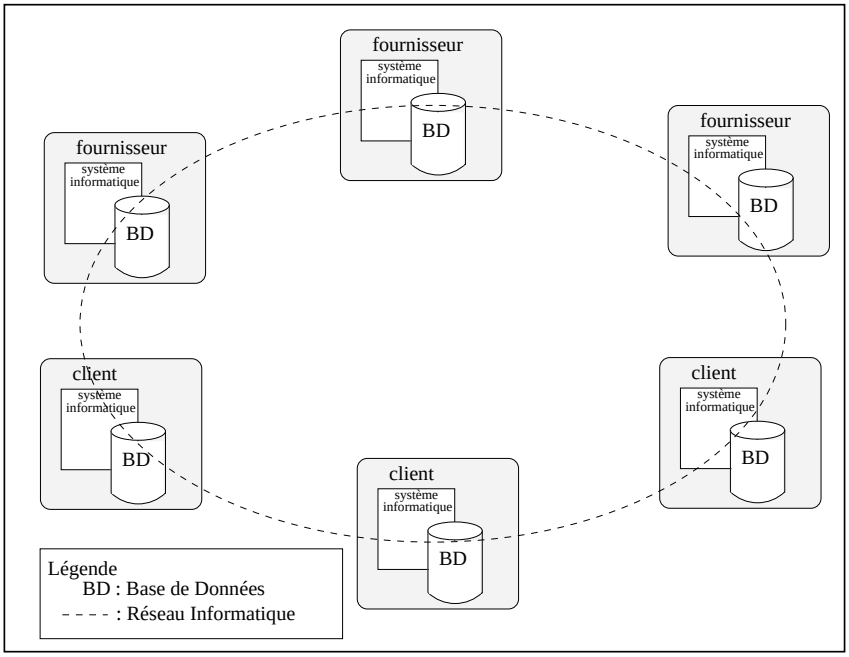

Figure 10. *Application de gestion de marchandises en milieu urbain*

4.2.1.1. Les fournisseurs

Les fournisseurs ont pour objectif de vendre des produits aux clients. Cela suppose qu'ils possèdent des stocks de marchandises à leur disposition. Ces stocks sont modélisés par des systèmes d'information. Un fournisseur va avoir pour principal objectif de gérer les commandes effectuées par ses clients et de répondre aux questions qui lui sont posées.

Nous allons introduire des agents afin de modéliser les fournisseurs selon le paradigme SMA. Le premier de ces agents, appelé *agent de stock*, sera chargé de gérer les stocks du fournisseur. Il procédera à un renouvellement des stocks lorsque ceux-ci descendront au-dessous d'un seuil critique. Cet agent sera complètement autonome et passera son temps à scruter l'état des stocks, c'est-à-dire qu'il examinera le système d'information, produit par produit, et lorsque cela sera nécessaire, il réalimentera le stock. Un fonctionnement en flux tendu rend nécessaire parfois de modifier le seuil au-dessous duquel on doit repasser une commande en tenant compte des délais de livraison. L'agent de stock a donc la capacité d'ajuster ce seuil en fonction de la fréquence des commandes effectuées par les clients.

Un *agent de commande* est chargé de traiter les commandes effectuées par les clients. Il reçoit les commandes sous forme de messages envoyés par des agents situés au niveau des clients. Pour chaque commande traitée, il renvoie un message vers le

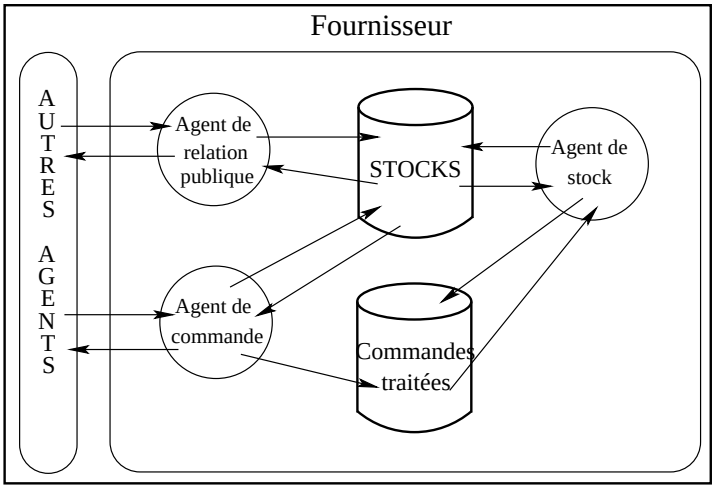

Figure 11. *Modélisation du fournisseur*

fournisseur qui lui fournit le résultat du traitement de la commande (satisfaite, non satisfaite, délais de livraison, etc.).

Un *agent de relation publique* est en charge de répondre aux questions éventuelles des clients. La forme de ces questions est déterminée dans la section 4.2.1.6. Cet agent doit également être en mesure de communiquer avec les agents situés au niveau des clients.

Pour modéliser les agents constituant le fournisseur (voir figure 11), nous avons utilisé le modèle DISMAS [DUV 01].

4.2.1.2. Les clients

Les clients ont deux objectifs :

– récupérer suffisamment d'informations pour choisir un ou plusieurs fournisseurs ;

– passer des commandes aux fournisseurs sélectionnés.

La première partie constitue la partie aide à la décision de notre système de gestion de marché. Elle se traduit par la possibilité d'effectuer des interrogations sur les systèmes d'information mis à leur disposition. Ces interrogations seront décrites dans la section 4.2.1.6. Nous avons modélisé ces interrogations par des agents de requête, selon un comportement *anytime* et avec la possibilité d'évoluer en environnement distribué. Chaque interrogation est effectuée au travers d'une interface conviviale. Elle est contrôlée par un agent d'interface qui la redirige vers les agents de requête concernés.

La seconde partie est gérée par un *agent de commande* qui, à partir des éléments fournis dans l'interface de l'application, envoie un message à son homologue concerné

du côté fournisseur. Cette commande est ensuite enregistrée dans le système d'information du client avec les informations qui lui sont associées (nom du fournisseur, délai de livraison, etc.).

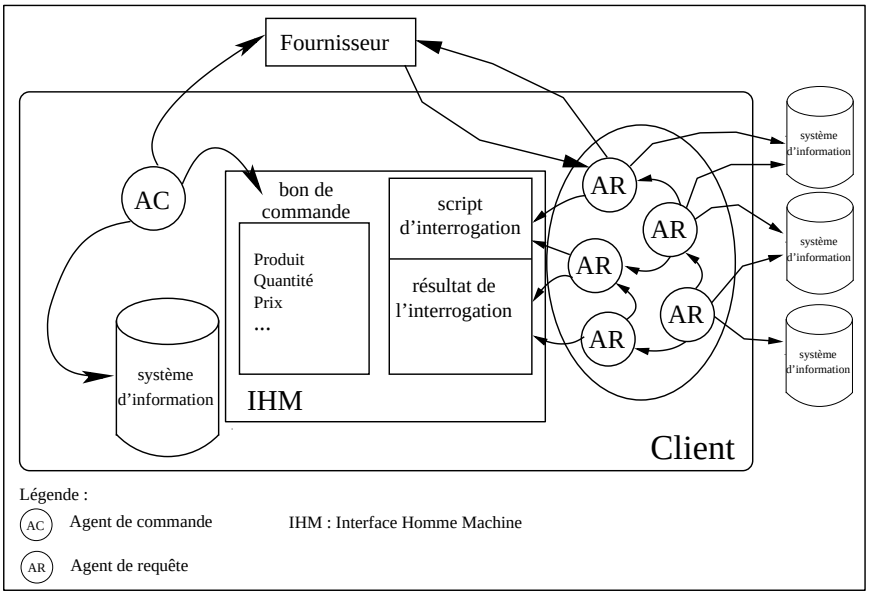

Figure 12. *Modélisation du client*

La modélisation du client (voir figure 12) repose d'une part sur l'utilisation d'une Interface Homme-Machine (IHM) qui permet à l'utilisateur d'interagir avec le système informatique, et d'autre part sur l'utilisation d'un SMA qui permet l'interaction avec les autres composantes du système (fournisseurs, systèmes d'information, etc.).

4.2.1.3. Les objets de l'application

L'application de gestion de marché manipule différents objets qui vont déterminer les objets manipulés par les agents et les tables présentes dans les différents systèmes d'information. Parmi les objets manipulés, il y a les produits qui forment le système d'information *stock* disponible auprès d'un fournisseur. On trouve également l'objet *commande* qui correspond aux objets manipulés par les clients et les fournisseurs. Nous allons présenter ces objets dans la section suivante.

4.2.1.4. Les produits

Les produits constituent les éléments de base du système de gestion de marché. Un produit possède des caractéristiques comme le nom, le prix et la quantité.

Ces données servent à alimenter les bases de données *stock* présentes au niveau de chaque fournisseur. Dans chaque commande effectuée, sont précisés le type de produit, la quantité commandée et le prix unitaire.

4.2.1.5. Les commandes

Les commandes sont des éléments échangés entre clients et fournisseurs. Ces échanges sont représentés au moyen d'*agents de commandes* qui traitent ces commandes en échangeant des messages. Le traitement d'une commande par un agent situé du côté fournisseur consiste à vérifier la disponibilité du produit en stock, puis à effectuer la commande, c'est-à-dire mettre à jour la base de données *Stock* et la base de données *Commandes*. Mettre à jour la base de données *Stock* consiste à décrémenter les quantités des produits concernés. Mettre à jour la base de données *Commandes* consiste à insérer les différents éléments de la commande ainsi que la date. Ces éléments peuvent ainsi être utilisés par les *agents de stock* pour gérer les stocks des fournisseurs.

4.2.1.6. Les interrogations

Il s'agit de décrire les interrogations effectuées essentiellement par les clients. Nous avons considéré deux types d'interrogations :

– les interrogations directes : ce sont les interrogations qui consistent à poser des questions directement à un fournisseur. Nous en donnons quelques-unes :

- le prix pratiqué pour un produit X par le fournisseur F ;
- la quantité minimale de produit X vendue ;
- etc.

– les interrogations générales : il s'agit de questions qui nécessitent une analyse plus fine et le recours à des sous-interrogations vers plusieurs systèmes d'information (par exemple : quel est le cours du marché pour le produit X). Ces interrogations constituent une part importante du système d'aide à la décision car elles permettent d'obtenir des informations liées aux prises de décision. Nous allons donner ici quelques exemples de ces interrogations :

- le cours moyen du produit X sur les six derniers mois ;
- les fournisseurs susceptibles de fournir le produit X et le délai moyen ;
- la quantité d'un produit X que le fournisseur F peut fournir ;
- etc.

4.3. *Un exemple d'interrogation*

Dans cette section, nous allons présenter un exemple d'interrogation tel qu'il est implémenté dans l'application que nous avons développée. Dans un premier temps, un schéma d'interrogation se présente sous la forme d'une interface (cf. figure 13),

permettant à un utilisateur de rentrer ses données (ici, demander quel est le cours du marché d'un produit sur une période donnée).

Figure 13. *Un exemple d'interrogation*

Les informations permettant de répondre à cette question sont stockées dans une base de données et il s'agit donc d'effectuer de simples requêtes pour y accéder et les présenter à l'utilisateur. L'agent chargé de cette interrogation est modélisé suivant un ATN (cf. figure 14). Cet agent va, en fonction du temps dont il dispose, fournir une réponse plus ou moins précise (prendre en compte plus ou moins de données) et plus ou moins élaborée (donner un résultat brut ou un résultat sous forme d'une courbe graphique, par exemple).

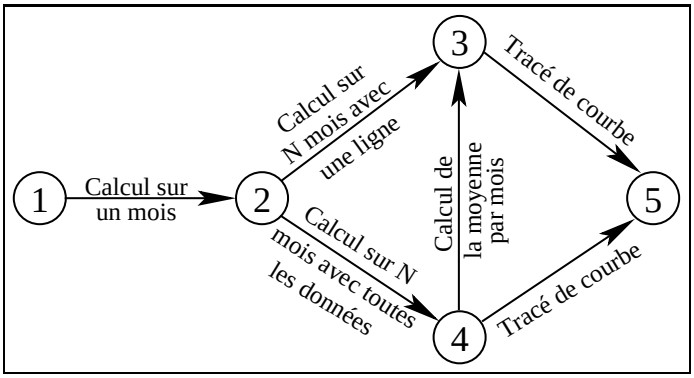

Figure 14. *ATN correspondant au traitement de l'interrogation de la figure 13*

Dans la figure 15 est présenté un résultat complet (cumul des résultats intermédiaires) concernant un calcul de moyenne. L'algorithme a permis de fournir un résultat approximatif très tôt. Ce résultat a ensuite été enrichi au fur et à mesure du déroulement de l'algorithme (donc du temps disponible).

L'exemple présenté donne un aperçu du fonctionnement de notre prototype mais n'utilise pas toutes les capacités du modèle ANYMAS.

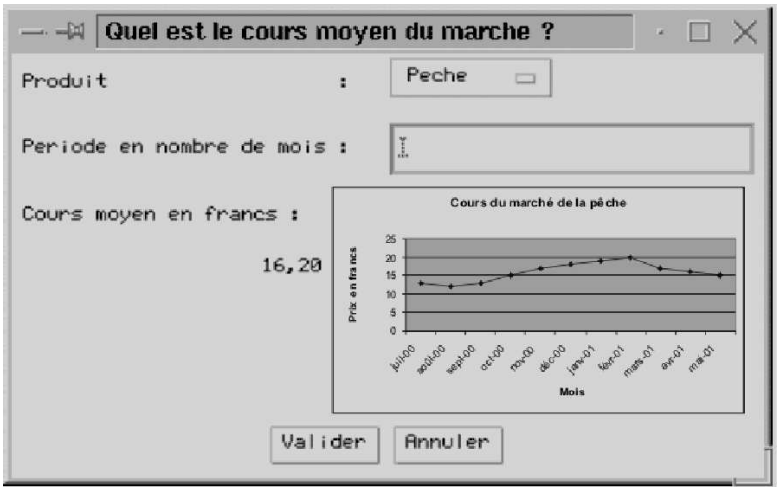

Figure 15. *Réponse graphique à une interrogation (cf. figure 13)*

4.4. *Bilan de l'application*

Nous avons présenté, dans cette section, une application de gestion de marchés. La description des principaux composants de l'application nous a permis de dégager une modélisation suivant le paradigme multiagent. Cette application permet de valider nos travaux concernant les systèmes multiagents *anytime*. Néanmoins, le prototype réalisé nécessite des améliorations. La version implémentée montre seulement la faisabilité du système.

5. Conclusion et perpectives

Dans cet article, nous avons présenté notre contribution dans le domaine des systèmes multiagents temps réel. Cette contribution est concrétisée par la conception du modèle ANYMAS, qui est un modèle de système multiagent où est pris en compte l'aspect temps réel, fondé ici sur une approche *anytime*. Le modèle ANYMAS permet de construire des applications temps réel complexes pouvant bénéficier des avantages de l'approche multiagent tout en respectant les échéances des transactions. Pour mettre en œuvre le modèle ANYMAS, nous avons choisi une application dans le domaine de l'aide à la décision : la gestion de marché électronique. Cette application est pour l'instant restreinte à des interrogations simples dans le domaine des produits alimentaires. Nous espérons pouvoir prochainement l'étendre à un environnement plus large et plus complexe. La réalisation de cette application n'a pas permis d'exploiter pleinement le modèle ANYMAS. Elle devrait pouvoir être améliorée pour servir de base à des applications permettant des interrogations plus complexes.

Une autre perspective de notre travail est l'application du modèle ANYMAS dans le cadre d'applications temps réel critiques, telles que la gestion d'un système de contrôle des sirènes d'alerte pour la prévention et la détection des risques technologiques sur un site industriel d'une ville, par exemple. Ce travail consiste à doter les instances de décision de moyens d'alerte leur permettant de détecter des situations potentiellement anormales. Il s'agit de donner aux décideurs un état courant de la situation de sites industriels à hauts risques [BOU 00]. Dans ce type d'applications, les décideurs ont la charge de l'instant de prise de décision et peuvent par conséquent attendre ou pas, d'obtenir des informations supplémentaires pour avoir une meilleure analyse de l'état d'une situation.

Parmi les possibilités d'extensions de notre modèle, nous envisageons de le rendre plus générique, notamment au niveau de la réification des agents de coordination temporelle qui est, à l'heure actuelle, en grande partie laissée à la charge des concepteurs d'applications. L'agent *anytime* fixe un cadre pour la conception sous forme d'une extension de la plate-forme MadKit. Pour cette réification, nous pourrons prendre en compte d'autres aspects que les communications, tels que l'activité ou le poids des agents. Un cadre applicatif plus large nous permettrait sans doute d'affiner le modèle ANYMAS. Un autre exemple d'extension de ce modèle pourrait être la prise en compte de la mobilité au niveau des agents *anytime* et par conséquent des agents de coordination temporelle. Lorsqu'un agent de coordination temporelle est créé, il est rattaché à un groupe d'agents *anytime*. Si la majorité des agents de ce groupe migrent sur une autre machine, il serait naturel que cet agent y migre également afin de réduire le coût des communications. D'autres extensions peuvent être envisagées comme l'implantation de mécanismes de tolérance aux fautes dans un contexte distribué, en utilisant la réplication d'agents.

6. Bibliographie

[ABB 88] ABBOTT R., GARCIA-MOLINA H., « Scheduling Real-Time Transactions : A Performance Evaluation », *International Journal of Distributed and Parallel Databases*, vol. 1, n° 2, 1988.

[BOU 00] BOUKACHOUR H., DUVALLET C., CARDON A., « Multiagent System to Prevent Technological Risks », *Proceedings of ACIDCA'2000*, March 2000.

[BRI 01] BRIOT J.-P., DEMAZEAU Y., « *Introduction aux systèmes multi-agents* », p. 17-25, Collection IC2, Hermes Science Publications, 2001.

[CAR 99] CARDON A., « La modélisation des systèmes dynamiques : caractères d'une modélisation non locale », *Revue d'Intelligence Artifi cielle*, vol. 13, nº 2, 1999, p. 169-200, Hermès.

[DEA 88] DEAN T., BODDY M., « Solving time-dependant planning problems », *Proceedings of the 7^{th} National Conference of Artifi cial Intelligence*, Minneapolis, Minnesota, 1988, p. 419-454.

[DRO 93] DROGOUL A., « De la simulation multi-agents à la résolution collective de problèmes : une étude de l'émergence de structure d'organisation dans les systèmes multi-agents », PhD thesis, Université de Paris VI, 1993.

[DUR 99] DURAND S., « Représentation des points de vues multiples dans une situation d'urgence : une modélisation par organisations d'agents », PhD thesis, Université du Havre, 1999.

[DUV 99] DUVALLET C., MAMMERI Z., SADEG B., « Les SGBD temps réel », *Technique et science informatiques*, vol. 18, n° 5, 1999, p. 479-516.

[DUV 01] DUVALLET C., « Des systèmes d'aide à la décision temps réel et distribués : modélisation par agents », PhD thesis, Université du Havre, 2001.

[ENG 88] ENGELMORE R., MORGAN T., *Blackboard Systems*, Adison-Wesley Publishing Compagny, 1988.

[FER 95] FERBER J., *Les systèmes multi-agents*, InterEditions, Paris, 1995.

[FER 98] FERBER J., GUTKNECHT O., « A Meta-Model for the Analysis and Design of Organizations in Multi-Agent Systems », *Proceedings of International Conference on Multi Agent Systems*, Cité des Sciences - La Villette, Paris, France, July 1998.

[GAR 93] GARVEY A., LESSER V., « Design-to-time Real-Time Scheduling », *IEEE Transactions on Systems, Man and Cybernetics, Special Issue on Planning, Scheduling and Control*, vol. 23, n° 6, 1993, p. 1491-1505.

[GRA 95] GRASS J., ZILBERSTEIN S., « Programming with Anytime Algorithms », *Proceedings of the 14^{th} Internationnal Joint Conference on Artifi cial Intelligence*, Montreal, 1995, p. 22-27.

[GUE 97] GUESSOUM Z., BRIOT J.-P., « From Concurrent Objects to Autonomous Agents », *8^{th} European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, Ronneby, 1997.

[GUT 97] GUTKNECHT O., FERBER J., « MadKit : Organizing heterogeneity with groups in a platform for multiple multi-agent systems », rapport, 1997, LIRMM, UMR 9928, Université de Montpellier II.

[HOR 87] HORVITZ E., « Reasonning about beliefs and actions under computational resource constraints », *Proceedings of the Workshop on Uncertainty in Artifi cial Intelligence*, Seattle, Washington, 1987.

[HOR 02] HORLING B., LESSER V., VINCENT R., WAGNER T., « The Soft Real-Time Agent Control Architecture », *Proceedings of the AAAI/KDD/UAI-2002 Joint Workshop on Real-Time Decision Support and Diagnosis Systems*, July 2002.

[MAC 98] MACKINNON L. M., MARWICK D. H., WILLIAMS M. H., « A Model for Query Decomposition and Answer Construction in Heterogeneous Distributed Database Systems », *Journal of Intelligent Information System*, vol. 11, 1998, p. 69-87, Kluwer Academic Publishers.

[MOU 93] MOUADDIB A., « Contribution au raissonnement progressif temps réel dans un univers multi-agents », PhD thesis, Université de Nancy I, Octobre 1993.

[MOU 99] MOUADDIB A., « Anytime Coordination for Progressive Planning Agents », *Proceedings of the 16^{th} National Conference on Artifi cial Intelligence and 11^{th} Conference on on Innovative Applications of Artifi cial Intelligence*, Orlando, Florida, 1999, AAAI Press/The MIT Press, p. 564-569.

[OCC 94] OCCELLO M., DEMAZEAU Y., « Building Real-Time Agent using Parallel Blackboards and its use for Mobile Robotics », *IEEE International Conference on Systems, Man and Cybernetics*, San Antonio, 1994.

[RAM 93] RAMAMRITHAM K., « Real-Time Databases », *Proceedings of the* 14^{th} *International VLDB Conference*, 1993.

[RUS 91] RUSSELL S., ZILBERSTEIN S., « Composing Real-Time Systems », *Proceedings of the* 12^{th} *Int. Joint Conf. on Artifi cial Intelligence*, Sydney, Australia, 1991, p. 212-217.

[SAL 97] SALVANT T., BRUNESSAUX S., « Multi-Agent Based Time-Critical Decision Support for C31 Systems », *Practical Applications of Intelligent Agents and Multi-Agents, PAAM*, London, 1997.

[SAL 98] SALVANT T., « CAAM, Un Modèle d'Agent Anytime Coopératif pour l'Aide à la Décision en Temps Critique », PhD thesis, ENST Paris, 1998.

[STA 88] STANKOVIC J., RAMAMRITHAM K., *Hard real-time systems : a tutorial*, IEEE Computer Society Press, 1988.

[WAG 00] WAGNER T., LESSER V., « Design-to-Criteria Scheduling : Real-Time Agent Control », *Proceedings of AAAI 2000 Spring Symposium on Real-Time Autonomous Systems*, Stanford, CA, March 2000, p. 89-96.

[WIL 92] WILLIAMS M., HU J., « Communication between Heterogeneous Medical Databases », *Journal of Information Science and Technology*, vol. 2, 1992, p. 12-30.

[WOO 94] WOOLDRIDGE M., JENNINGS N., « Agent Theories, Architectures and Language : A Survey », WOOLDRIDGE M., JENNINGS N., Eds., *Intelligent Agents, ECAI 1994*, vol. LNAI 890, Springer Verlag, 1994, p. 1-32.

[WOO 95] WOOLDRIDGE M., JENNINGS N. R., « Intelligent Agents : Theory and Practice », *The Knowledge Engineering Review*, vol. 10, n° 2, 1995, p. 115-152.

[ZIL 94] ZILBERSTEIN S., « On the Utility of Planning », *SIGART Bulletin, Special Issue on Evaluating Plans, Planners, and Planning Systems*, vol. 6, n° 1, 1994, p. 42-47.

[ZIL 96] ZILBERSTEIN S., « Resource-Bounded Sensing and Planning in Autonomous Systems », *Autonomous Robots*, vol. 3, 1996, p. 31-48.

Article reçu le 6 janvier 2003
Version révisée le 6 novembre 2003
Rédacteur responsable : Zahia Guessoum

Claude Duvallet *est maître de conférence à l'université du Havre. Il a effectué ses travaux de thèse sur les systèmes multiagents temps réel. Il travaille actuellement au sein de l'équipe SGBD temps réel du Laboratoire d'Informatique du Havre (LIH) sur la problématique de la gestion de la qualité de service dans les SGBD temps réel et dans les systèmes multimédias distribués.*

Bruno Sadeg *est maître de conférence à l'université du Havre. Il dirige l'équipe SGBD temps réel du laboratoire LIH. Ses centres d'intérêt portent principalement sur la gestion des transactions temps réel : contrôle de concurrence, validation, modèles étendus, qualité de service et qualité de données. Les techniques Anytime sont l'une des pistes sur laquelle travaille l'équipe pour satisfaire l'objectif de certaines applications temps réel : celui de fournir des résultats exploitables (même altérés) avant une échéance donnée.*