Des systèmes d'aide à la décision temps réel et distribués : modélisation par agents

Claude Duvallet

Université du Havre

Des systèmes d'aide à la décision temps réel et distribués : modélisation par agents

THÈSE

présentée et soutenue publiquement le 5 octobre 2001

pour l'obtention du

Doctorat de l'université du Havre

(spécialité informatique)

par

Claude Duvallet

Composition du jury

Président du jury :	Gérard Duchamp, Professeur (LIFAR, Rouen)
Rapporteurs :	Alexis Drogoul, HDR (LIP6, Université de Paris VI) Zoubir Mammeri, Professeur (IRIT, UPS Toulouse)
Examinateurs :	Abdel-Illah Mouaddib, Professeur (GREYC, Caen) Bruno Sadeg, co-directeur, MCF (LIH, Le Havre) Yann Pollet, Ingénieur, Docteur (Matra MSI, Val de Reuil)
Directeur de thèse :	Alain Cardon, Professeur (LIH, Le Havre)

Laboratoire d'Informatique du Havre

Remerciements

Je tiens tout d'abord à remercier le Professeur Alain Cardon, mon directeur de thèse, qui fut d'abord pour moi l'enseignant qui m'a donné le goût de la recherche. Il m'a permis de découvrir le domaine de l'Intelligence Articielle, plus particulièrement celui des systèmes multi-agents. Aujourd'hui, j'achève des travaux de thèse, ce qui n'aurait pas été possible sans lui.

Mes remerciements vont aussi à Bruno Sadeg qui m'a co-encadré durant toute cette thèse. Mon travail, notamment la rédaction d'articles de recherche et de ce document n'aurait pas été le même sans lui. Grâce à lui et aux nombreuses relectures de ce document, mon travail de rédaction a pu être amélioré.

Je tiens à remercier Messieurs Mammeri et Drogoul pour avoir accepté de rapporter sur ce travail. Leur avis et leurs précieux conseils m'ont permis d'améliorer la rédation de ce document. Le professeur Zoubir Mammeri m'a apporté toute la rigueur nécessaire à la présentation de ces travaux de thèse. Je tiens aussi à le remercier pour son soutien au début de mes travaux de thèse. Alexis Drogoul restera pour moi un exemple à suivre concernant les travaux effectués dans le domaine des systèmes multi-agents. La première thèse que j'ai lue dans le domaine était la sienne. J'ai eu la chance d'avoir comme rapporteurs deux personnes que j'admire beaucoup pour leur rigueur scientifique. Ils m'ont donc fait grand honneur en rapportant sur ma thèse.

Je tiens aussi à remercier les Professeurs Gérard Duchamps et Abdel-Illah Mouaddib, ainsi que Monsieur Yann Pollet d'avoir accepté de me faire l'honneur de participer à mon jury de thèse.

Je tiens à remercier tous les collègues du laboratoire d'informatique, ainsi que les enseignants-chercheurs que j'ai cotoyés au cours de cette thèse. Ils m'ont permis de tenir le coup lorsque j'ai eu parfois envie d'arrêter ma thèse mais aussi qui ont fait parti de ma vie durant toute cette période. Je ne peux pas citer l'ensemble des personnes qui ont fait le quotidien de ma thèse.

Des remerciements particuliers pour Cyrille Bertelle et Véronique Jay qui m'ont permis d'améliorer la qualité de mon exposé en participant de façon active aux pré-soutenances que j'ai effectuées. Véronique fait aussi partie des personnes ayant relue ce document et corriger les fautes qui s'y trouvaient.

Parmi ces collègues qui sont parfois devenus des amis, je citerai Frédéric Guinand qui m'a convaicu de ne pas arrêter au bout de ma première année. Il y aussi Aziz qui est pour moi un véritable ami avec qui j'ai eu de très intéressantes discussions, Alexandre qui m'a permis de me détendre certains soirs et de profiter de ces moments d'insouciance autour d'un verre. Il y a tous ces collègues avec qui j'ai travaillé à la fac ou à l'IUT : Michel, Patrick, Christophe, Frédéric, Véronique, Damien, Cyrille, Laurent, Bruno, Gaële. Il y a aussi les autres membres du personnel : Yasmina, notre charmante secrétaire du département informatique de l'IUT ; tous les ingénieurs systèmes, Daniel, Catherine, Olivier, Sandrine, Eric, Philippe, etc.

Mes derniers remerciements vont aux autres doctorants qui ont déjà ou pas encore soutenu : Stéphane, Franck, Mina, Jean-Philippe, Mustapha, Pierrick, Olivier, Djemila, Stéphane. Ils ont aussi contribué chacun à leur façon à faire que ce travail soit possible. Je mets une mention particulière pour Mina qui m'a supporté durant toute cette période.

Je dédie cette thèse
à ma famille qui m'a soutenu tout au long de mes études, et à mes amis.

Table des matières

Table des figures

Introduction générale

1 Contexte

Les systèmes d'aide à la décision (SAD) [Lévine and Pomerol, 1989] reposent de plus en plus souvent sur des systèmes informatiques qui sont chargés de fournir au décideur le maximum d'éléments dans les meilleurs délais et qui l'aideront dans sa prise de décision. Dans la majorité des cas, il s'agira pour l'application informatique, d'explorer des systèmes d'information (SI) et d'en extraire les éléments les plus pertinents. La complexité des systèmes informatiques devant être mis en place pour explorer, analyser et présenter les informations à l'utilisateur a amené beaucoup de chercheurs à utiliser une modélisation reposant sur le paradigme multi-agents [Ferber, 1995] [Wooldridge and Jennings, 1994]. Cette approche permet en effet de tenir compte de la complexité de ces systèmes en extrayant les entités d'actions qui doivent être présentes dans la modélisation du système.

Comme nous l'exprimions dans le paragraphe précédent, les SAD sont soumis à des contraintes de temps. Concevoir des SAD revient donc à concevoir des systèmes temps réel [Stankovic and Ramamritham, 1988] capables de fournir des résultats dans un délai contrôlé. Or, l'utilisation du paradigme multi-agents implique la nécessité d'y intégrer des mécanismes qui tiennent compte du contrôle de la dimension temporelle. Cela rend nécessaire la conception de systèmes multi-agents (SMA) dotés de caractéristiques temps réel.

Obtenir un résultat avant une échéance fixée à l'avance reste très difficile malgré les progrès enregistrés dans le domaine des outils informatiques (performances du matériel) ansi que des logiciels temps réel. Cependant, dans le cadre des systèmes d'aide à la décision, un résultat incomplet ou partiel obtenu dans les temps est souvent préférable à un résultat précis et complet obtenu hors délais, et donc inexploitable. L'approche *anytime* apparaît comme étant très prometteuse pour la prise en compte de telles situations [Zilberstein and Russell, 1996] [Zilberstein and Russell, 1993]. C'est la raison pour laquelle, nous avons pensé qu'il serait pertinent d'employer ces techniques (anytime) pour la conception de notre modèle de SMA temps réel utilisé pour l'aide à la décision. Nous présenterons dans ce document les raisons de ce choix.

L'objet de nos travaux est donc la prise en compte du temps réel dans un système

basé sur le paradigme multi-agent pour la conception de systèmes d'aide à la décision distribués. Nous examinerons dans la section suivante la problèmatique de ce travail.

Nous avons illustré nos travaux sur un problème de prise de décision dans une application de gestion de marchés pour aider les utilisateurs à effectuer des transactions commerciales dans des conditions les meilleures possibles. Pour l'interrogation de systèmes d'information par le SAD, il est nécessaire d'avoir recours à des agents ayant des caractéristiques temps réel. Il faut en effet pouvoir disposer de l'information qui permet de prendre une décision, dans un temps contraint. Les systèmes d'information utilisés sont structurés de façon à permettre une exploration à différents niveaux de profondeur allant du plus général au plus précis.

Dans ce mémoire, nous présentons la contribution que nous avons apportée à l'étude des systèmes multi-agents. Elle consiste essentiellement en la conception d'un modèle de SMA temps réel basé sur l'utilisation de techniques anytime : ANYMAS (abbréviation de ANYtime Multi-Agent System) et en la réalisation d'un modèle de SMA physiquement distribué : DISMAS (abbréviation de DIStributed Multi-Agent System).

Pour l'implémentation de notre modèle de SMA temps réel ainsi que du modèle de SMA distribué, nous nous sommes basés sur une plate-forme multi-agent existante : MadKit, qui sera sera présentée en annexe. Elle a été enrichie pour permettre la mise en œuvre de techniques anytime au sein des agents et permettre de plus l'utilisation en mode distribué de systèmes d'agents.

2 Problèmatique

Nous nous plaçons dans le domaine des systèmes d'aide à la décision (SAD) et plus particulièrement de ceux dont la conception repose sur un modèle multi-agents. Ces systèmes sont conçus de façon à mettre à la disposition des décideurs un outil informatique leur permettant de prendre des décisions par rapport à une situation donnée avec un maximum de clairvoyance. Cela consiste à avoir une vue aussi précise que possible de la situation et se traduit par l'obtention d'un maximum d'informations pertinentes. Comme certaines décisions doivent être prises dans l'urgence (avant des échéances), il est nécessaire que l'information soit mise à disposition des décideurs le plus tôt possible. Il s'agit ici de l'aspect temps réel des systèmes d'aide à la décision. L'information disponible est présente dans des systèmes d'informations (SI) qui se complètent les uns les autres mais dont l'information peut être redondante, voire contradictoire. Il est donc nécessaire lors de l'interrogation d'effectuer une fusion des informations afin d'en retenir l'essentiel sous sa forme la plus pertinente possible ; nous avons donc eu recours au paradigme multi-agent.

Un système d'aide à la décision doit également permettre à des utilisateurs multiples d'extraire des informations réparties dans des systèmes d'informations distants et distri-

bués. On doit alors tenir compte de la distribution du système.

Pour résumer, les systèmes d'aide à la décision que nous ciblons doivent reposer sur des systèmes multi-agents temps réel et distribués pour permettre l'interrogation de systèmes d'information physiquement distribués. Nous allons extraire les points essentiels de cette problèmatique :

- la conception de SMA temps réel (paragraphe 2.1),
- la conception de SMA physiquement distribués (paragraphe 2.2),
- l'interrogation de SI par des SMA temps réel et distribués (paragraphe 2.3).

2.1 Les SMA temps réel pour les SAD

Nous nous intéressons à une classe d'applications particulière : les applications reposant sur des systèmes d'aide à la décision (SAD) qui exploitent des systèmes d'information distincts. Dans cette classe d'applications, on va chercher à extraire toute l'information pertinente par rapport à une situation donnée dans des délais contraints. En effet, certaines situations requièrent des prises de décision dans l'urgence ou du moins avant des échéances temporelles fixées.

Cependant, il arrive souvent que toute l'information disponible à propos d'une situation ne puisse pas être extraite avant l'échéance impartie. Nous avons donc eu recours aux techniques anytime, qui semblent une solution prometteuse. En effet, ces techniques permettent d'extraire un maximum d'informations exploitables progressivement. Plus le système dispose de temps, mieux seront exploitables les informations extraites. Dans le meilleur cas (quand le système dispose de tout le temps nécessaire), les résultats obtenus seront complets et précis.

Dans notre cadre de travail, des SMA vont devoir acquérir un comportement anytime. L'acquisition de ce comportement anytime va devoir se faire à deux niveaux :

- le niveau local : il s'agit du niveau agent. Nous allons donc doter les agents d'un comportement anytime.
- le niveau global : il s'agit du niveau SMA. Nous allons donc devoir influencer le comportement du SMA de façon à ce qu'il devienne anytime.

Pour répondre à ces besoins, nous avons proposé un modèle de SMA anytime, appelé modèle ANYMAS (ANYtime MultiAgent System). Nous le présentons dans le chapitre 6. Il s'agit du premier apport scientifique de nos travaux qui consiste donc à doter de propriétés temps réel les systèmes multi-agents au moyen des techniques anytime. Pour cela, nous étendons les travaux réalisés jusqu'à maintenant dans le domaine des systèmes multi-agents en leur apportant une dimension supplémentaire. D'une manière réciproque, nous développons une nouvelle façon de concevoir des systèmes temps réel en utilisant les caractéristiques d'adaptabilité des systèmes multi-agents.

2.2 Distribution de SMA dans les SAD

Les systèmes d'aide à la décision sont très souvent des systèmes d'information distribués multi-utilisateurs. Il est donc nécessaire qu'ils reposent sur des systèmes informatiques distribués : des utilisateurs répartis sur plusieurs sites coopèrent au travers de l'outil informatique. C'est pourquoi dans le cadre de nos travaux, nous nous sommes intéressés aux systèmes multi-agents physiquement distribués.

Dans les systèmes multi-agents, il s'agit d'interconnecter des systèmes multi-agents situés sur des machines distinctes pouvant être géographiquement distribuées. Cela signifie donc que les agents présents dans chaque système doivent être en mesure d'échanger des informations ou des services. Nous allons présenter plus en détails la problématique de la réalisation de SMA distribués ainsi qu'un modèle répondant à cette problématique dans le chapitre 7. Le modèle de distribution ainsi conçu n'est pas doté de caractéristiques temps réel et devra donc être combiné avec le modèle ANYMAS pour la conception de systèmes multi-agents temps réel et distribués. Nous avons en effet tenu à séparer les problèmes de distribution physique et de comportement temps réel des systèmes multi-agents.

2.3 Interrogation de SI par des SMA temps réel et distribués

Dans le cadre de nos travaux, le composant essentiel des SAD est constitué par des SI physiquement répartis car ils appartiennent le plus souvent à des entités différentes. L'interrogation de ces systèmes permet de fournir au décideur les éléments nécessaires pour sa prise de décision. La qualité et la pertinence des informations extraites est primordiale car elle influencera la décision.

Lorsqu'une demande d'information est effectuée par un utilisateur, il s'agit pour le système d'aller extraire l'information la plus pertinente possible dans les délais impartis et la présenter à l'utilisateur (décideur). Le système informatique chargé d'effectuer l'interrogation doit choisir l'information à extraire selon les critères de temps et de pertinence. Plus le système disposera de temps pour effectuer son extraction, plus il pourra se permettre d'augmenter la qualité du résultat et la précision des informations extraites.

Pour mettre en œuvre une interrogation anytime, il est donc nécessaire de réaliser un schéma d'interrogation qui réponde à ces caractérisques et qui par conséquent interviendra dans l'agentification du système. De plus, il est nécessaire de prévoir une structuration des systèmes d'information afin qu'elle puisse répondre à ces critères (information plus précise qu'une autre mais équivalente, information complémentaire, connexe, etc.). Cette partie reposera sur les deux modèles évoqués précédemment et proposera une méthode de conception pour les systèmes ayant recours à des interrogations temps réel (anytime) et distribuées.

L'ensemble de nos travaux ayant pour objectif la conception de SAD avec des carac-

téristiques anytime et distribuées, nous présentons une application de système de gestion de marché permettant de mettre en application ces travaux. Dans la suite de cette introduction générale, nous décrivons l'organisation de ce mémoire.

3 Organisation du mémoire

Dans ce document, nous étudions les systèmes multi-agents temps réel et plus particulièrement l'approche anytime utilisée pour implémenter les aspects temps réel dans les SMA et leur application dans le cadre des systèmes d'aide à la décision. Nous commençons par présenter dans le chapitre 1, les travaux relatifs aux systèmes d'aide à la décision.

Le chapitre 2 est consacré à l'etude des systèmes temps réel et des algorithmes anytime dans un cadre plus général que celui des SAD.

Dans le chapitre 3, nous présentons les systèmes multi-agents, en particulier les modèles sur lesquels nous fondons nos travaux. Ce chapitre présente les notions qui seront utilisées tout au long du document.

Pour nos travaux sur les systèmes d'aide à la décision, nous avions besoin d'étudier les systèmes multi-agents distribués et les agents mobiles. Nous donnerons donc un aperçu des travaux effectués dans ce domaine et en quoi nous pouvons retenir certains aspects pour nos travaux. Nous verrons pourquoi nous nous limiterons aux systèmes multi-agents sans prendre en compte la mobilité des agents.

Dans le chapitre 4, nous présentons les travaux relatifs aux systèmes multi-agents distribués et aux agents mobiles. Nous y présentons les travaux de l'OMG [OMG, 1997] concernant l'intéropérabilité des plate-formes agents.

Le chapitre 5 sera consacré à l'étude des travaux relatifs aux systèmes multi-agents temps réel et aux domaines connexes. Ce chapitre a pour objectif de souligner l'originalité de nos travaux par rapport aux travaux existants.

Le chapitre 6 présente le modèle ANYMAS : un modèle de conception pour la réalisation de systèmes multi-agents temps réel basé sur l'utilisation de techniques anytime. Ce chapitre constitue une part de la contribution de nos travaux présentés dans cette thèse.

Le chapitre 7 concerne la réalisation de DISMAS : un modèle de système multi-agent distribué. Il s'agit là du second volet de nos travaux de recherche.

Le chapitre 8 concerne l'exploration de systèmes d'information dans le cadre d'un système d'aide à la décision par des SMA anytime et distribués. Il s'agit de l'exploitation de nos travaux décrits dans les deux chapitres précédents dans la conception d'un modèle d'exploration des systèmes d'information.

Le chapitre 9 est consacré à la présentation d'une application de gestion de marché en milieu urbain et sa modélisation. L'objectif principal de cette application est d'illustrer nos travaux de recherche sur les SMA anytime et distribués. Il s'agit d'une application

entrant dans le cadre des systèmes d'aide à la décision multi-participants.

Le chapitre 10 présente la réalisation informatique de cette application en insistant sur les choix de conception que nous avons effectués. Cette présentation replacera les choix techniques qui ont été effectués dans le contexte de la modélisation présentée dans le chapitre précédent.

Dans le dernier chapitre de cette thèse, nous présentons le bilan de nos travaux et donnerons quelques pistes sur l'exploitation qui pourrait en être faite ainsi que des pistes sur les travaux de recherche qui permettraient de compléter ou d'approfondir ceux présentés dans ce document.

Chapitre 1

Les systèmes d'aide à la décision

Résumé

La recherche sur les systèmes d'aide à la décision (SAD) existe depuis environ trente ans. Elle est donc antérieur à l'existence des systèmes multi-agents. Les travaux sur les SAD sont nombreux et couvrent un grand champ d'applications. En effet, de nombreuses applications font intervenir à différents niveaux de complexité des SAD dans les entreprises et le monde industriel. Ces SAD s'appuient sur des travaux qui font intervenir plusieurs domaines de recherche informatique tels que les systèmes de gestion de bases de données (SGBD), la recherche opérationnelle, l'intelligence artificielle, etc. Dans ce chapitre, nous allons tenter de cerner le domaine des SAD en présentant leurs caractéristiques, les outils qu'ils utilisent, et les domaines d'application. Nous replacerons nos travaux par rapport à cet existant.

1.1 Introduction

Depuis environ trente ans, des outils sont apparus dans les entreprises pour aider au traitement de l'information et à la prise de décision [Benchimol et al., 1990] : ils sont connus en France sous le nom de systèmes d'aide à la décision (SAD) ou systèmes interactifs d'aide à la décision (SIAD) [Lévine and Pomerol, 1989]. Dans le monde anglo-saxon, ils sont appelés « decision support system » (DSS).

On pouvait distinguer, à l'origine, deux types de SAD. Ceux qui, incorporant des statistiques et de la recherche opérationnelle [Cohen, 1995], font une large place aux algorithmes d'optimisation et aux calculs numériques, et ceux qui sont tournés exclusivement vers la gestion de l'information (bases de données, gestion de fichiers et des flux d'informations dans l'entreprise).

Une caractéristique commune à tous ces systèmes est l'interactivité. Lors de la conception de ces systèmes, il s'agit en effet de prévoir une interaction facilitée au maximum

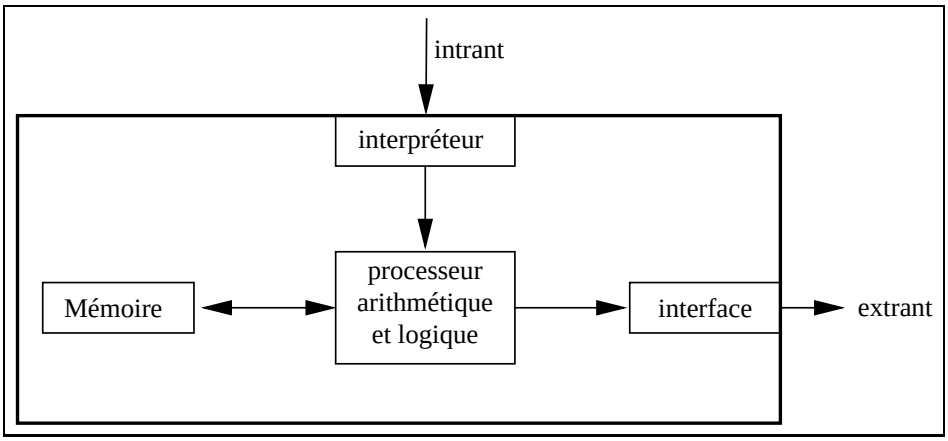


FIG. 1.1 – *STI à modèle fixé*

entre le système et l'utilisateur. L'utilisateur doit pouvoir disposer de l'information qui lui est nécessaire sous sa forme la plus adéquate pour prendre des décisions. L'utilisation d'outils en provenance de l'intelligence artificielle [Lévine and Pomerol, 1989] a permis de renforcer le traitement de l'information en éliminant les informations non pertinentes et en présentant des faits déduits à partir des données présentes dans les systèmes d'information. Le lien entre l'intelligence artificielle et les SAD n'a fait que se renforcer au cours des années [Pomerol, 1996] [Courbon et al., 1994].

Dans ce chapitre, nous allons définir les systèmes d'aide à la décision et présenter leurs caractéristiques essentielles (voir section 1.2). Nous présenterons ensuite les systèmes experts et l'intelligence artificielle (voir section 1.3) qui sont indiscutablement liés aux SAD. Nous donnerons un aperçu des différents domaines d'utilisation des SAD dans la section 1.4. Nous conclurons ce chapitre en soulignant l'apport de nos travaux pour l'extension possible du domaine d'utilisation des SAD.

1.2 Qu'est-ce qu'un SAD?

Un SAD est un outil permettant d'assister le décideur dans sa prise de décision. En effet, un processus de décision ne peut pas être entièrement automatisable [Pomerol, 1992]. Un SAD est un Système de Traitement de l'Information (STI, voir figures 1.1 et 1.2) [Lévine and Pomerol, 1989] qui permet d'extraire et de donner au décideur l'information nécessaire au processus de prise de décision. Cependant, cela n'est pas suffisant pour caractériser un SAD et le différencier des systèmes d'information classiques. Il est beaucoup plus significatif d'ajouter qu'un SAD est un résolveur de problèmes (*problem solver*, voir figures 1.2 et 1.3).

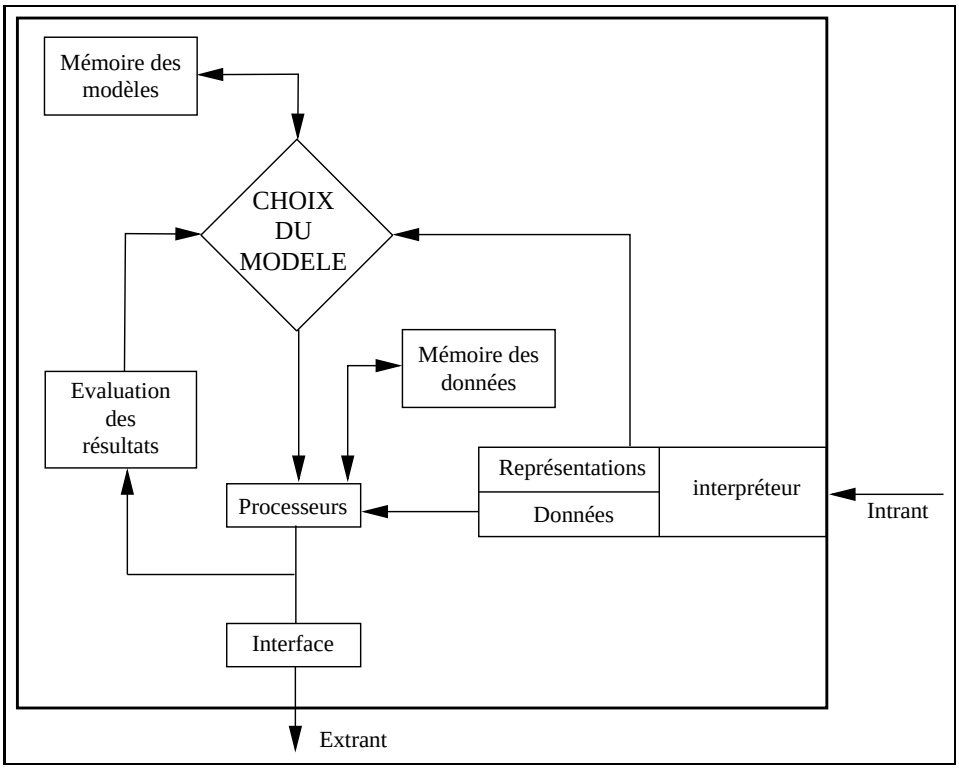


FIG. 1.2 – *STI à modèles multiples*

1.2.1 Rôle d'un SAD

Un SAD est un système informatique dont le rôle est d'assister le décideur tout au long du processus de décision. Un processus de décision, dans le cadre défini par la gestion des organisations, se compose de quatre phases [Lévine and Pomerol, 1989] : une phase d'information, une phase de conception, une phase de choix, et une phase d'évaluation du choix. La procédure décrite par cette succession de phases n'est pas purement séquentielle. Des retours en arrière peuvent se produire, notamment lors de la phase de conception ; l'élaboration d'un scénario peut nécessiter l'acquisition d'informations supplémentaires.

Si la phase de choix relève du seul décideur, un système d'information a sa place dans les deux phases de préparation à cette prise de décision, et dans la phase d'évaluation du choix. En effet, la capacité de traitement des informations des ordinateurs permet au décideur, pendant la phase d'information, d'accéder rapidement à des informations brutes ou traitées concernant la situation courante et le champ des manœuvres autorisées par exemple. Cette capacité de traitement de l'information peut être aussi utilisée lors de l'évaluation des scénarios décrivant les différentes options envisagées par le décideur lors de la phase de conception. Dans cette phase, le système d'information peut fournir des éléments d'évaluation des scénarios à l'aide d'indicateurs calculés à partir de modèles ou de procédures de calcul adaptés. La phase d'évaluation du choix correspond à une évaluation

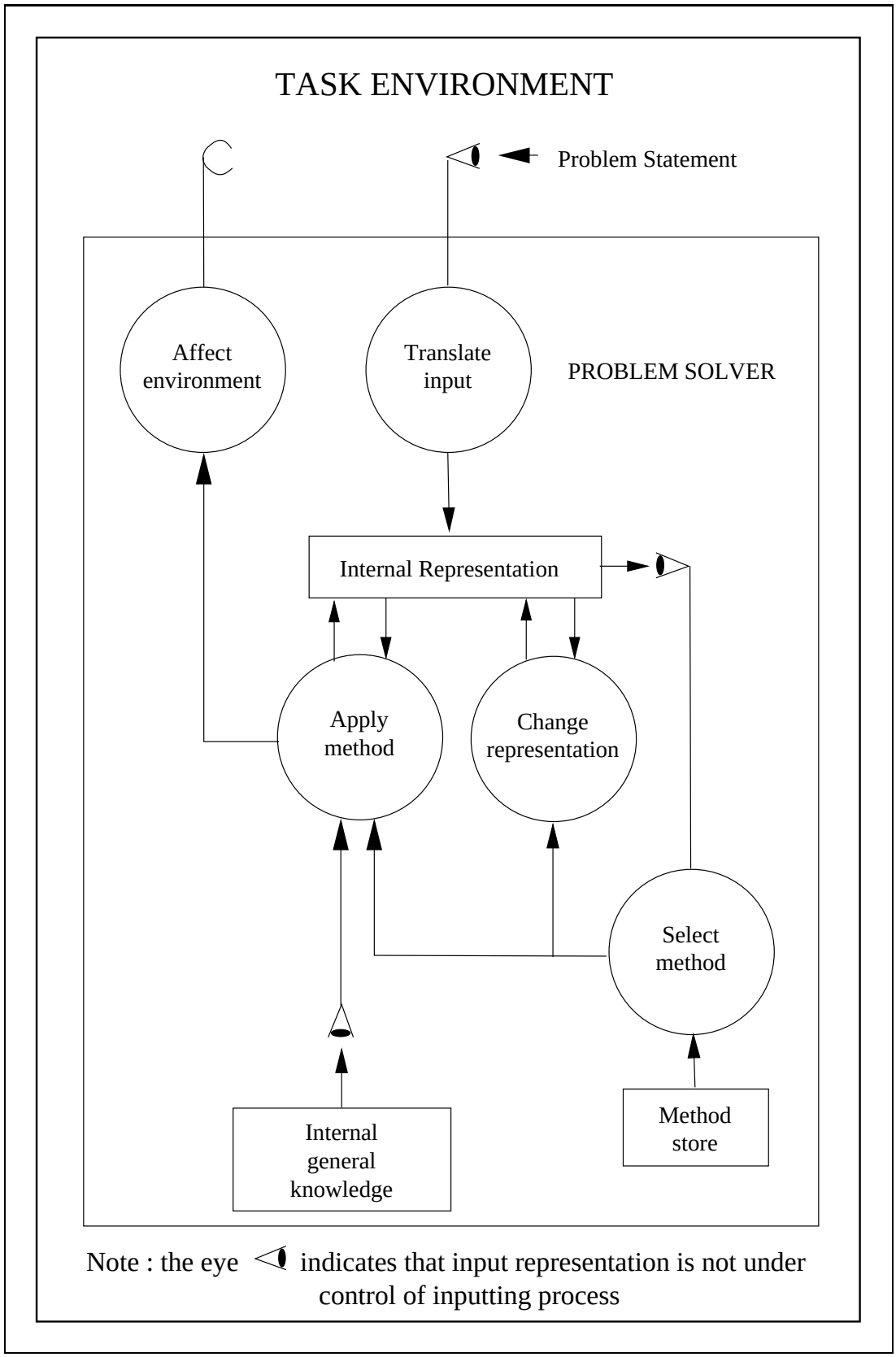


FIG. 1.3 – *Organisation d'un « problem solver » [Newell and Simon, 1972]*

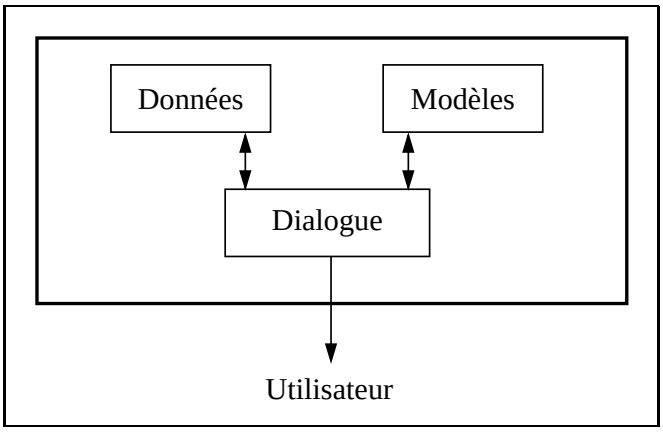


FIG. 1.4 – *Principe des SAD [Sprague, 1987]*

a posteriori du choix du décideur ; cette évaluation permet de corriger les petites erreurs. La détection des erreurs et des aspects à améliorer peut être facilitée par l'apport d'informations et d'indices calculés par le système d'information [Kersten and Mallory, 1990] [Lévine and Pomerol, 1989].

Un SAD, en favorisant l'association d'un système informatique et d'un décideur, permet la formation d'un système de décision complet. Le décideur, de par sa connaissance pratique, possède un méta-modèle du processus de décision, et le SAD, par sa capacité de traitement de l'information, l'aide à structurer le modèle [Lévine and Pomerol, 1989]. En d'autres termes, le décideur contrôle le processus de décision et le SAD l'assiste en effectuant les calculs standards et répétitifs sur les données [Kersten and Mallory, 1990]. Dans une telle perspective, le processus de décision s'identifie à une recherche heuristique menée par le décideur ; le système jalonne le processus de recherche à l'aide d'indicateurs et d'informations. Le décideur, en fonction de ces informations produites à l'issue du traitement des données, continue l'exploration heuristique des actions possibles, ou arrête si tout lui indique que la solution construite rencontre ses buts de façon satisfaisante [Pomerol, 1997] [Lévine and Pomerol, 1989]. De ce fait, la coopération entre le décideur et le système informatique ne peut être fructueuse que dans le cadre d'un système interactif [Pomerol, 1997] [Pomerol, 1992] [Lévine and Pomerol, 1989].

1.2.2 Principe d'un SAD

Un SAD se compose de trois modules : un module de dialogue, un module contenant les données et un module contenant les procédures de calcul ou modèles. Comme l'indique la figure 1.4, le module de dialogue est interconnecté avec les deux autres modules. Il constitue l'interface entre l'utilisateur et le reste du système. Dans les paragraphes qui suivent, nous présentons le rôle et la composition de chacun des modules.

Le module de dialogue est un composant essentiel du SAD. Par l'intermédiaire des interfaces gérées par ce module, le décideur accède aux données et aux fonctions de calcul, et le système utilise le même vecteur pour lui communiquer le résultat des manipulations effectuées par le décideur. Les échanges sont d'autant plus favorisés que les représentations des résultats (tout comme le mode de questionnement du système) correspondent aux représentations mentales du décideur. Ainsi, le décideur peut exercer son contrôle et effectuer sa recherche heuristique dans de bonnes conditions.

Le module base de données assure la fonction de mémoire ; il stocke non seulement les données, de façon permanente ou temporaire, mais il gère également l'enregistrement de données volatiles ainsi que l'effacement de ces données selon le souhait de l'utilisateur. Les données volatiles correspondent aux résultats obtenus au cours de traitements des données. Les données permanentes sont les statistiques ou autres données qui décrivent la situation courante et passée. Parmi ces données, il peut aussi y avoir des estimations concernant l'évolution de certains paramètres environnementaux.

Le module « modèle » contient l'ensemble des procédures de calcul utilisées dans les différents traitements standards des données mis à disposition de l'utilisateur. Il peut s'agir de calculs standards (d'indices et d'indicateurs par exemple) et de procédures de représentation des données. Si les procédures de représentation intègrent une dimension spatiale (cartes géographiques, plan d'usine par exemple) alors on est face à un « Système Spatial Interactif d'Aide à la Décision » (Spatial Decision Support System).

1.3 Les systèmes experts et l'intelligence artificielle

Il est maintenant classique de chercher parmi les outils de l'intelligence artificielle (IA) des idées pour améliorer les performances des SAD [Lévine and Pomerol, 1989]. Beaucoup de systèmes experts furent explicitement placés par leurs concepteurs sous le signe de l'aide à la décision. En effet, MYCIN fut dès l'origine présenté comme un SAD alors que PROSPECTOR est conçu comme le prolongement des techniques de la décision bayésienne.

1.3.1 Des raisonnements aux systèmes experts

Un des buts fondamentaux que se sont assignés les chercheurs en intelligence artificielle est la reproduction par les machines des raisonnements humains [Benchimol et al., 1990]. La première tâche consiste donc à savoir comment l'homme raisonne. Selon les tâches qu'il doit accomplir, le raisonnement d'un homme diffère. Le raisonnement d'un joueur d'échec n'est pas le même que celui d'un chef d'entreprise.

En fait, il existe de nombreuses façons de raisonner et d'envisager des situations complexes ; un bref aperçu des principaux modes de raisonnement peut le prouver. Il existe,

en effet, deux modes de base :

- *la déduction* qui permet de tirer des conclusions à partir de règles dont on vérifie les prémisses ;
- *l'induction* qui conduit aux règles à partir d'observations partielles conduites suivant des schémas prédéterminés.

Ces deux raisonnements peuvent être utilisés de façon *analytique*, (on découpe le raisonnement en sous-modules plus faciles à appréhender), ou *synthétiques* (on rassemble au contraire des éléments épars). Nous nous arrêterons ici pour la description des différents modes de raisonnements mis en évidence pour la conception de systèmes intelligents : des détails peuvent être trouvés dans [Haton et al., 1991].

La fonction dévolue aux systèmes experts est le raisonnement, ce qui est — comme nous venons de le voir — déjà assez vaste. Nous pouvons donc donner la définition suivante d'un système expert : c'est un programme informatique capable de reproduire des raisonnements humains.

1.3.2 Principe de base des systèmes experts

Un système expert est un système de production capable de reproduire les raisonnements d'un expert dans un domaine bien défini. Si plusieurs domaines d'expertise interviennent, on parle alors de systèmes multi-experts [Benchimol et al., 1990].

Que fait un expert ? Il part d'un certain nombre de faits ou de données, les analyse et rend un verdict. Les experts élaborent des enchaînements de raisonnements. Dans un système expert, les raisonnements sont mis sous forme de règles de production du type :

Si la *précondition est satisfaite* alors *exécuter une action.*

Parmi les règles qui entrent dans la catégorie précédente, nous ferons une place à part aux règles déductives (ou *inférences*) qui sont du type :

Si *prémisse est vraie* alors *conclusion est vraie.*

Les règles de production de ce dernier type sont à la base de la plupart des systèmes experts. Elle entrent dans le type général précédemment défini, en ce sens que la précondition est satisfaite si la prémisse se trouve dans la base des faits avérés ; l'exécution de la règle conduit alors à ajouter la conclusion à cette même base des faits vrais.

1.4 Exemples de domaines d'application des SAD

Si l'on souhaitait résumer l'ensemble des domaines d'application des SAD, on dirait que des SAD peuvent être mis en place au niveau de chaque activité humaine nécessitant un processus de décision élaboré. Ce résumé donne un aperçu de l'ampleur du domaine

d'application des SAD. Il existe déjà des SAD fonctionnant au quotidien dans les entreprises. Nous allons présenter quelques applications de ce type. D'autres SAD restent du domaine de la recherche et nécessitent encore des études approfondies pour pouvoir être opérationnels.

1.4.1 Un système d'allocation de wagon

A la SNCF, la répartition des wagons de marchandises est effectuée, sur le plan national, par un service appelé « central répartition » [Lévine and Pomerol, 1989]. Au sein du « central répartition », les experts appelés « répartiteurs », reçoivent par téléphone les demandes de wagons vides en provenance des zones, centralisent les informations sur les wagons vides disponibles et assurent la répartition entre les zones déficitaires et les zones excédentaires. Autrement dit, ils prennent des wagons dans les zones excédentaires pour les envoyer vers les zones utilisatrices.

Outre les demandes ou les offres téléphonées, les répartiteurs disposent des données de GCTM (Gestion Centralisée du Transport Marchandise) qui fournit un état de la situation à trois heures du matin, état qui sort sur imprimante vers huit heures du matin à l'arrivée des répartiteurs. Cette situation donne un état des besoins et des wagons présents en zone, ainsi que la moyenne des besoins et des ressources des jours précédents.

Le SAD, mis en place dans ce cadre, a pour objectif d'aider le répartiteur dans sa tâche. Il est basé sur l'utilisation d'un système à base de connaissances. Les critères principaux qui régissent le travail des répartiteurs et qui doivent se retrouver dans le SAD sont les suivants :

- minimisation des coûts de déplacement à vide ;
- respect des délais ;
- satisfaction des clients.

Suite à l'observation du travail des répartiteurs, d'autres heuristiques ont pu être dégagées pour la mise en œuvre du SAD.

Il s'agit là d'un premier exemple de SAD qui peut être généralisé à l'ensemble des problèmes d'allocation de ressources en extrayant les caractériques propres à chaque problème.

1.4.2 La gestion de production

Un autre type de SAD pouvant être utile dans l'industrie concerne les systèmes de gestion de production.

Un projet de ce type a été mené par le Carnegie-Mellon Robotics Institute sous la direction de M.S. Fox [Fox, 1981]. Dans ce type d'application, il s'agit d'optimiser la production en influant sur certains critères : changer les techniques de conception, changer

le matériel, anticiper les pannes en proposant des solutions de remplacement, synchroniser au mieu les différentes étapes de production. Un SAD va aussi permettre de répondre à certaines questions que l'on peut se poser :

- Quel est le bon niveau de mesure à différentes étapes de la production ?
- Que se passe-t-il si une machine tombe en panne ?
- Quand dois-je changer un équipement ?
- Une pièce est en avance, une autre est en retard :
  - De combien la date de livraison va-t-elle être modifiée ?
  - Quels sont les changements de personnels nécessaires pour tenir les dates ?
  - Comment le retard va-t-il se répercuter de proche en proche ?
- Comment choisr la pièce à mettre en main et la machine à utiliser ? Que faire en cas de modification du plan standard ?
- Comment faire circuler l'information technique ?
- Comment faire de la maintenance préventive ?
- Comment détecter les baisses de rendement des machines ?
- etc.

À partir de ces questions, on peut rédiger un « cahier des charges » correspondant au SAD à mettre en place. Ce SAD doit permettre de faire du suivi de production. Il permet également d'anticiper les problèmes pouvant survenir et proposer des solutions de rechange ou tout au moins effectuer des simulations (répondre aux questions précédentes) par rapport à certaines modifications de paramètres.

Ce type de SAD pour la gestion de production est applicable à de nombreuses usines de production.

1.4.3 La gestion de marché et le commerce électronique

Avec l'explosion de l'utilisation d'Internet, on a vu s'accroître le nombre de systèmes de gestion de marchés permettant au travers de sites WEB d'effectuer des transactions commerciales. L'utilisateur qui a recours à ce genre de transactions n'est pas un expert, d'où la réticence du plus grand nombre à y avoir recours. Prenons par exemple le cas des systèmes permettant d'effectuer des transactions boursières. Dans ce domaine, seuls les agents de change possèdent toute l'expertise nécessaire pour effectuer les meilleurs choix concernant l'achat/vente d'actions, d'obligations, etc.

Lorsqu'un utilisateur va vouloir investir en bourse, il a besoin d'avoir les meilleurs conseils possibles en fonction de ses envies et de ses choix potentiels. Un agent de change, pour le conseiller, lui poserait un certain nombre de questions :

- Voulez-vous un investissemnt à long terme, à court terme ?

- Voulez-vous un rapport sûr et stable?
- Voulez-vous maximiser votre gain?
- Êtes-vous prêt à prendre des risques et dans quelles proportions?
- Voulez-vous étudier tel investissement plus particulièrement? Cours de l'action sur les six derniers mois? Rapports d'activités? etc.

Un SAD devra être en mesure de répondre à toutes ces questions qui correspondront à autant de critères de choix concernant un investissement. Il aura pour but de conseiller l'investisseur potentiel tout comme le ferait un agent de change.

Dans les systèmes de gestion de marché et de commerce électronique, il s'agit pour le SAD de fournir de la valeur ajoutée en extrayant des informations qui habituellement ne sont disponibles qu'au prix d'un lourd effort (comparaison de prix entre plusieurs magasins, lecture de magazines spécialisés, études des descriptifs techniques, etc.)

1.4.4 Un système de surveillance pour les sites insdustriels à haut risque

Un dernier domaine que nous allons présenter est celui de la gestion de risques industriels. Il fait actuellement l'objet d'une étude dans le cadre de recherches universitaires.

La présence de sites industriels à très hauts risques technologiques en milieu urbain représente, comme leur nom l'indique, un risque non négligeable pour la population [Boukachour et al., 2000]. Une solution serait l'installation d'un réseau de sirènes qui doit permettre le confinement des populations. Il s'agit pour un opérateur préposé à la détection des risques majeurs de déclencher ces sirènes au moment opportun. Pour prendre sa décision, il doit être en mesure d'avoir une vision globale de la situation et des événements survenus ainsi que de leurs conséquences éventuelles (réaction en chaîne). La connaissance concernant l'évolution d'une situation est dispersée auprès de différents experts. On prend ici comme exemple le risque de formation d'un nuage toxique et sa propagation dans l'atmosphère. Pour avoir une évaluation de la situation, on va devoir interroger des experts en chimie, en météorologie, en traitement des situations d'urgence (gestion des populations, traitement des blessés par les services concernés, traitement des accidents routiers par les pompiers, etc.).

L'ensemble des informations à traiter et à synthétiser par l'opérateur, qui prendra au final la décision de déclencher les sirènes de confinement, est vaste. Le SAD, qui sera mis en place pour aider l'opérateur dans sa tâche, devra avoir recours à cette multi-expertise et mettre en évidence l'information pertinente. La connaissance requise dans ce cadre est floue et difficile à traiter contrairement aux exemples de SAD précédemment cités.

1.5 Conclusion

Dans ce chapitre, nous avons présenté le domaine des systèmes d'aide à la décision en donnant leurs principales caractéristiques. En résumé, on peut dire qu'un SAD est un système permettant à un décideur une prise de décision en toute connaissance de cause. Il assiste le décideur sans jamais se subtituer à lui. Nous avons eu également un aperçu de l'étendue du domaine d'application au travers de quelques exemples.

Nous proposons dans le cadre de nos travaux une approche nouvelle pour la conception et le traitement des SAD qui utilise des systèmes multi-agents temps réel et distribués. En effet, jusqu'à maintenant on avait tendance à traiter la résolution des problèmes liés aux SAD par le biais de systèmes experts dont les bases de connaissances étaient figées lorsque l'expertise menée sur le terrain était terminée. Ces systèmes par la taille des bases de connaissances à prendre en compte étaient souvant très lourds et très complexes à mettre en place. Leur fonctionnement reposant sur l'utilisation de moteurs d'inférence exploitant des bases de connaissances très lourdes, il était quasiment impossible de doter de capacités temps réel les SAD. Notre approche permet de traiter des systèmes complexes en simplifiant la modélisation et de disposer de capacités à s'adapter accrues. De plus, dans cette approche nous dotons les SAD de capacités temps réel et distribuées. Un autre aspect de cette approche est qu'elle permet d'accroître, comme nous le verrons, le domaine d'utilisation des SAD. Notre méthode permet également de reprendre certaines applications liées aux SAD et de proposer un nouveau traitement afin d'optimiser leurs performances.

Chapitre 2

Systèmes temps réel et algorithmes anytime

Résumé

Les applications temps réel classiques sont présentes dans de nombreux domaines, notamment dans celui des industries. Une application temps réel est une application où l'exécution des actions est soumise à des contraintes temporelles (souvent sous forme d'échéances). On a l'habitude de classer les actions ou tâches dans les systèmes temps réel selon les conséquences provoquées par le non-respect de leurs échéances. La première catégorie concerne les tâches temps réel à échéances strictes critiques ; le manquement de l'échéance par une tâche peut avoir des conséquences catastrophiques en terme de vies humaines, de matériel, d'environnement, etc. La seconde catégorie concerne les tâches temps réel à échéances strictes non critiques ; le non-respect de l'échéance par une tâche entraîne l'inutilité et donc l'abandon de la tâche correspondante. La troisième et dernière catégorie concerne les tâches temps réel à échéances non strictes ; le dépassement de l'échéance est alors sans conséquence et la tâche peut continuer de s'exécuter ; seule la qualité de service sera diminuée. Dans le domaine de l'intelligence artificielle temps réel, une autre voie que celle utilisée dans les systèmes temps réel classiques a été suivie : il s'agit des algorithmes anytime. Le terme anglais « anytime » ne se traduit pas dans ce contexte et reste donc utilisé sous sa forme anglo-saxone par la communauté de recherche francophone. L'important dans les algorithmes anytime est le respect des contraintes temporelles. Pour cela, on admet que des applications puissent utiliser des résultats dont la qualité n'est pas optimale. Le principe de base des techniques anytime consiste à fournir des résultats de meilleure qualité au fur et à mesure que l'on accorde davantage de temps au système pour s'exécuter. Nous abordons dans ce chapitre les travaux effectués dans les domaines des systèmes temps réel classiques, puis, celui des algorithmes anytime. Ce chapitre est essentiel pour la compréhension des choix que nous avons effectués lors de la conception

de notre modèle de systèmes multi-agents temps réel.

2.1 Les applications temps réel et leur spécification

Dans cette section, nous allons présenter les applications temps réel et leurs caractérisques essentielles. Puis, nous aborderons un point essentiel de ces applications : la spécification des contraintes temporelles [Duvallet et al., 1999] [Mammeri, 1998] [Ramamritham, 1996]. C'est en effet cette spécification qui permet de décrire une application comme étant temps réel.

2.1.1 Caractéristiques des applications temps réel

Une application temps réel est généralement gérée par un système contrôleur (le système informatique) qui agit sur un système contrôlé (l'environnement physique de l'application) à l'aide d'actionneurs. Le fonctionnement général des applications temps réel peut se résumer ainsi :

- acquisition de données depuis l'environnement à l'aide de capteurs,
- traitement des données et élaboration des résultats au bout d'un délai limité,
- envoi d'ordres de commande à l'environnement à l'aide d'actionneurs.

A ces fonctions de base, et selon le domaine d'application considéré, viennent s'ajouter d'autres fonctions telles que la maintenance de l'installation industrielle et des équipements informatiques, la gestion du personnel, le service après-vente, ...

Un exemple d'application temps réel est celui de la commande d'une chaîne d'assemblage composée de convoyeurs, de stations d'assemblage et de robots. Dans cette application, des robots munis de caméras servent à acquérir les caractéristiques des objets (transportés par les convoyeurs) qui défilent devant leur « champ de vision ». Cette acquisition est contrainte par l'intervalle de temps durant lequel l'objet est présent devant la caméra du robot. Cela revient à dire que l'action d'acquisition des caractéristiques doit se terminer avant la disparition de l'objet : on dit que l'action possède une échéance.

Réservés il y a quelques années aux installations industrielles (telles que les laminoirs, les raffineries et les usines de fabrication de véhicules), les systèmes temps réel font leur apparition dans beaucoup d'autres secteurs tels que le transport, le multimédia, les consoles de jeux, le suivi de malades, etc. En termes de complexité, les systèmes temps réel couvrent un large spectre allant du simple microcontrôleur (pour le contrôle du système de freinage d'une voiture, par exemple), jusqu'aux systèmes répartis (pour le contrôle du trafic aérien, par exemple). Les enjeux économiques et les intérêts scientifiques liés aux systèmes temps réel sont multiples. C'est la raison pour laquelle on assiste, depuis

les années soixante-dix, à une profusion de langages, de méthodes, d'algorithmes et de protocoles de communication pour le temps réel.

De nombreuses définitions ont été proposées pour clarifier la notion de système (et application) temps réel. Cependant, comme les caractéristiques des systèmes temps réel sont très variées, aucune des définitions proposées n'est vraiment complète pour tenir compte de tous les domaines d'applications. En effet, selon l'aspect abordé dans les systèmes temps réel, on choisit une définition qui s'approche le plus de la problématique traitée. Ainsi, on assimile, selon le cas, un système temps réel à un système rapide, à un système en interaction directe avec un procédé physique, à un système réactif, à un système qui ne fournit pas de réponse en différé, à un système avec un comportement prédictible, à un système qui travaille sur des données « fraîches », à un système robuste, etc.

Cependant, toutes les applications temps réel ont en commun la prépondérance du facteur temps. En effet, les applications temps réel doivent réagir en tenant compte de l'écoulement du temps. Cette caractéristique fondamentale qui distingue globalement les applications temps réel des autres types d'applications informatiques (de gestion ou autres) est exprimée par la définition suivante qui est la plus couramment citée dans la littérature sur le temps réel : "A real-time system is defined as a system whose correctness of the system depends not only on the logical results of computations, but also on the time at which the results are produced" [Stankovic and Ramamritham, 1988].

En plus de l'existence de contraintes de temps, et selon les domaines d'applications, les systèmes temps réel doivent satisfaire d'autres contraintes primordiales, notamment la prédictibilité des comportements et la tolérance aux fautes. En effet, dans les applications temps réel dites temps critique (telles que la commande de procédés industriels ou d'engins militaires), le respect des contraintes de temps est une nécessité en toutes circonstances.

Toute application temps réel est en interaction (forte ou faible selon les cas) avec son environnement. Lequel environnement peut être un procédé industriel, un moteur d'avion, un malade, un groupe de participants à une téléconférence, etc. La nature de l'environnement a une incidence directe sur la "criticité" des actions entreprises dans une application temps réel. La notion de "criticité" (ou "criticality" en anglais) est utilisée comme critère pour pouvoir classer les applications temps réel selon la sévérité, en termes de coût engendré par le non respect des contraintes temporelles. Ainsi, on classe les applications temps réel en deux types : les applications à contraintes temporelles strictes (« hard real-time applications »), où le non-respect des contraintes de temps peut conduire à des défaillances avec des conséquences pouvant être graves, et les applications à contraintes temporelles relatives (« soft real-time applications »), où le dépassement des échéances est considéré comme une faute bénigne.

2.1.2 Spécification des contraintes temporelles

Les contraintes temporelles peuvent prendre des formes diverses (périodes, échéances absolues ou relatives, etc.) et s'appliquer à divers composants d'une application, c'est-à-dire à des actions (par exemple, une transaction doit se terminer avant une échéance fixée), à des données (par exemple, la validité temporelle des données est limitée) ou à des événements (par exemple, tel événement doit apparaître x unités de temps après tel autre événement). Pour plus de détails sur l'expression des contraintes temporelles, le lecteur pourra se référer à [Mammeri, 1998] [Ramamritham, 1996].

Une fois les contraintes de temps élaborées, ces contraintes doivent être clairement spécifiées avant d'entamer la phase de conception d'applications. La qualité de service de l'application finale repose en grande partie sur la rigueur de la spécification. En effet, la phase de spécification est importante pour toute application informatique et elle est plus cruciale pour les applications temps réel étant données les conséquences que peut avoir une spécification incorrecte ou incomplète. Les applications temps réel sont des plus difficiles à spécifier, en particulier lorsqu'elles sont réparties et qu'elles exigent un haut degré de sûreté de fonctionnement. C'est à partir des années quatre-vingts que la communauté de recherche en informatique a pleinement pris conscience de l'importance de l'élaboration de méthodes et langages (en particulier, ceux fondés sur un formalisme rigoureux) pour la spécification des contraintes temporelles.

Dans les techniques de spécification proposées, on distingue les techniques opérationnelles et les techniques descriptives. Les techniques opérationnelles sont celles définies en termes d'états et de transitions ; elles sont proches de l'exécution. Les techniques descriptives sont souvent fondées sur un formalisme mathématique et produisent des spécifications précises, rigoureuses et donnent une vue abstraite du système. Le système est décrit par des propriétés, forçant le spécifieur à exprimer ce que le système doit faire plutôt que de dire comment le système va le faire. Les spécifications formelles peuvent être traitées automatiquement pour en vérifier la complétude et la cohérence. Il faut noter que ces techniques sont peu utilisées dans le domaine du temps réel, comparées aux techniques opérationnelles, à cause, à la fois, du manque d'outils d'utilisation et des difficultés de spécifier formellement des systèmes complexes tels que les systèmes temps réel. Plusieurs classifications des techniques descriptives ont été proposées, dont en voici une :

- Techniques descriptives fondées sur les méthodes algébriques : elles utilisent les types abstraits. En particulier, des extensions de LOTOS, comme RT-LOTOS (Real-Time LOTOS) [Courtiat et al., 1993] et ET-LOTOS [Emerson and Halpern, 1986], commencent à émerger pour la spécification de systèmes temps réel.
- Techniques descriptives fondées sur les logiques mathématiques : elles permettent de décrire le comportement d'un système à l'aide de règles qui spécifient comment le

système peut évoluer. Pour prendre en compte les aspects temporels, des extensions des logiques temporelles classiques sont utilisées : CTL (Computational Tree Logic) [Emerson and Halpern, 1986], RTL (Real-Time Logic) [Johanian and Mok, 1986], RTIL (Real-Time Interval Logic) [Razouk and Gorlick, 1989], MTL (Metric temporal logic) [Koymans, 1989], RTTL (Real Time Temporal Logic) [Ostroff, 1989], TCTL (Timed CTL) [Alur and Henziger, 1990], TPTL (Timed Propositional Temporal Logic) [Alur and Henziger, 1990], TRIO [Ghezzi et al., 1990] et TRIO+ qui est présenté dans [Morzenti and Pietro, 1994].

Les techniques opérationnelles se divisent en deux catégories : les techniques fondées sur les modèles à transitions (machines d'états et réseaux de Pétri) et les techniques fondées sur des notations abstraites. Les machines d'états finis (MEF) permettent de vérifier des propriétés telles que l'atteignabilité des états. Pour être adaptées aux systèmes temps réel, les MEF doivent fournir des possibilités d'expression des contraintes de temps (échéances, périodes, ...). Beaucoup d'extensions de MEF ont ainsi été proposées. Des langages fondés sur les MEF, comme PAISLey (Process-oriented Applicative and Interpretable Specification Language), SDL (Specification and Description Language) [UIT, 1996], ESTEREL [Berry and Cosserat, 1984] et Statecharts [Harel et al., 1987], sont utilisés pour la spécification de systèmes temps réel. En ce qui concerne les techniques opérationnelles fondées sur les réseaux de Pétri (RdP), plusieurs extensions des RdP de base ont été proposées pour prendre en compte des contraintes temporelles. Il s'agit notamment des RdP temporisés [Berthomieu and Diaz, 1991] et des RdP stochastiques [Molloy, 1985].

Les techniques basées sur des notations abstraites sont adaptées à l'analyse et à la conception de systèmes, par décomposition d'un système en sous-systèmes. Elles sont souvent destinées à fournir une représentation visuelle du système pour réduire en quelque sorte l'effort du spécifieur. En général, elles ne modélisent pas l'aspect comportemental des systèmes et par conséquent elles ne sont pas directement utilisables pour une simulation ou pour une exécution du système. Par ailleurs, elles sont souvent informelles ou semi-formelles. Ces techniques incluent des extensions des méthodes SADT (telles que DARTS [Gomaa, 1986] - Design Approach for Real-Time Systems - et SDRTS [Ward, 1986] - Structured Design for Real-time Systems) ou HOOD (telles que HRT-HOOD [Burns and Wellings, 1994] - Hard Real-Time HOOD), largement utilisées dans la spécification d'applications non temps réel.

Il existe également des techniques mixtes telles que ESM/RTTL qui est une approche intégrant les machines d'états (Extended State Machines) et la logique temporelle RTTL [Ostroff and Wohnam, 1987].

Enfin, il faut souligner que dans la pratique, une technique de spécification n'est utile pour un spécifieur que lorsqu'elle est accompagnée d'outils permettant de spécifier, vérifier et simuler des comportements, de générer des implantations, etc. Actuellement, les outils

qui existent sont surtout sous forme de prototypes de recherche. Au niveau commercial, peu de produits existent pour appréhender les contraintes temporelles. Le lecteur pourra se référer à [Bucci et al., 1995] pour une présentation de différents outils et techniques de spécification d'applications temps réel. Une fois spécifiées, les contraintes temporelles d'une application doivent être prises en compte par des mécanismes adéquats.

Le point faible des systèmes temps réel classiques est qu'ils ne gèrent pas de manière efficace les applications qui travaillent sur de grandes quantités de données, notamment la cohérence et le contrôle de concurrence d'accès aux données. Les SGBD temps réel répondent de manière partielle à ces problèmes [Duvallet et al., 1999]. Mais les algorithmes anytime couplés avec les SMA semblement être la solution la plus prometteuse. Dans le paragraphe suivant, nous présenterons les algorithmes anytime.

2.2 Algorithmes anytime

Cette section est consacrée à l'etude d'une branche particulière des applications et système temps réel : les algorithmes anytime. Nous présentons ici ce qui constitue l'un des fondements de nos travaux concernant les SMA temps réel.

2.2.1 Présentation

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 notamment des systèmes permettant de contrôler des robots mobiles qui 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 dans certaines entreprises). Il peut s'agir par exemple d'un objet se présentant devant le robot et pouvant éventuellement constituer un obstacle. Il faut que le robot puisse évaluer à temps la nature de l'objet afin d'engager une procédure d'évitement si nécessaire. Connaître la nature exacte de l'objet peut amener le robot à engager la procédure d'évitement trop tard alors qu'une vision même floue de l'objet 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 donnerait son résultat quelque soit l'instant où l'échéance se produit est plus que souhaitable. Une nouvelle classe d'algorithme a permit de donner un début de réponse à ce type de problèmes : les algorithmes anytime.

L'apparition de ces algorithmes date de la fin des années 80 [Dean and Boddy, 1988] [Horvitz, 1987]. La définition de base d'un algorithme anytime est la suivante : un algorithme anytime est un algorithme qui échange du temps d'exécution contre une qualité

de résultat supérieure. En effet, dans un algorithme anytime plus le temps d'exécution laissé pour effectuer une tâche sera grand, plus la qualité du résultat fourni en sortie sera bonne.

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 dans les temps est souvent préférable à un résultat parfait mais qui arrive trop tard. Un de ces domaines où les algorithmes anytime trouvent particulièrement leur place, est le domaine de l'intelligence artificielle. Dans ce domaine, la construction d'un raisonnement qui se ferait de façon progressive au cours du temps permettrait de répondre à certains besoins concernant les échéances à respecter.

L'objectif de cette section est de présenter les algorithmes anytime, leurs principales caractéristiques, leur contexte d'utilisation. Dans la suite, nous commençons par présenter les deux types d'algorithmes anytime que nous pouvons rencontrer. Nous présentons les caractéristiques des algorithmes anytime avant de donner un exemple d'application de ces algorithmes : le problème du voyageur de commerce. Un des problèmes qui se pose souvent dans ce domaine est celui de la composabilité des algorithmes anytime que nous abordons finalement avant de conclure.

2.2.2 Algorithmes anytime par contrat versus algorithmes interruptibles

Habituellement, on extrait de la classe des algorithmes anytime, deux types d'algorithmes (voir figure 2.1) :

- algorithmes anytime par contrat,
- algorithmes anytime interruptibles.

Les algorithmes anytime par contrat nécessitent d'avoir connaissance, avant le début de l'exécution, du temps alloué pour pouvoir construire un résultat cohérent. Les algorithmes anytime interruptibles constituent l'idéal des algorithmes anytime : il s'agit d'algorithmes capables d'être interrompus à n'importe quel moment de leur exécution et de rendre un résultat cohérent puis de continuer à s'exécuter. Dans la suite de ce chapitre, nous nous consacrerons uniquement aux algorithmes anytime interruptibles que nous désignerons simplement par le terme algorithmes anytime.

2.2.3 Caractéristiques des algorithmes anytime

Les algorithmes anytime possèdent un certain nombre de propriétés qui permettent de les caractériser. Ce sont les suivantes :

- la qualité du résultat est fonction du temps alloué à l'exécution mais aussi de la qualité des ressources dont l'algorithme dispose en entrée ;

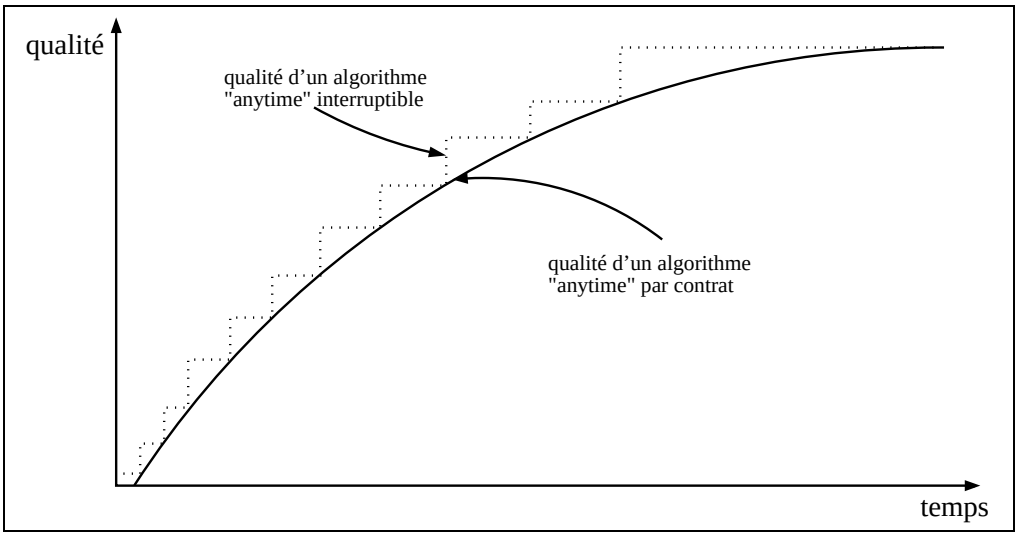


FIG. 2.1 – *Profils de performance selon le type d'algorithme anytime*

- des mécanismes doivent permettre la mesure de la qualité du résultat ;
- la prévisibilité : un algorithme anytime contient des informations statistiques sur la qualité du résultat en fonction du temps alloué, utilisables pour le méta-raisonnement sur les ressources en temps de calcul à donner à un algorithme anytime ;
- l'interruptabilité et la continuité : un algorithme anytime doit être capable de fournir un résultat quelque soit l'instant où l'on interrompt son exécution mais aussi de continuer à s'exécuter au delà du temps alloué initialement ; ceci permet de changer les temps de calcul alloués à un algorithme anytime en cours d'exécution ;
- la monotonicité : la qualité du résultat augmente toujours ou reste stable en fonction de la croissance du temps de calcul alloué mais ne diminue jamais.

Les algorithmes anytime mettent en corrélation la qualité d'un résultat avec le temps alloué au moyen de graphes appelés « Profil de Performances » (cf. figure 2.2). Ces profils de performances font partie des trois principaux éléments d'un algorithme anytime et qui sont :

1. une fonction itérative d'amélioration de la qualité : cette fonction augmente la qualité du résultat à chaque pas de l'exécution.
2. un évaluateur de résultat : cette fonction évalue le résultat et la fonction itérative d'amélioration pour retourner des informations sur la qualité du résultat.
3. un profil de performance (PP) : cette fonction retourne une estimation de la qualité étant donnée la quantité de temps alloué pour l'exécution de l'algorithme anytime. Si le PP utilise la qualité des données en entrée, il sera appelé profil de performance conditionnel (cf. figure 2.3). Sur cette figure, nous avons cherché à représenter différents profils de performance selon la qualité des données (q_1, q_2, q_3, q_4) prises en

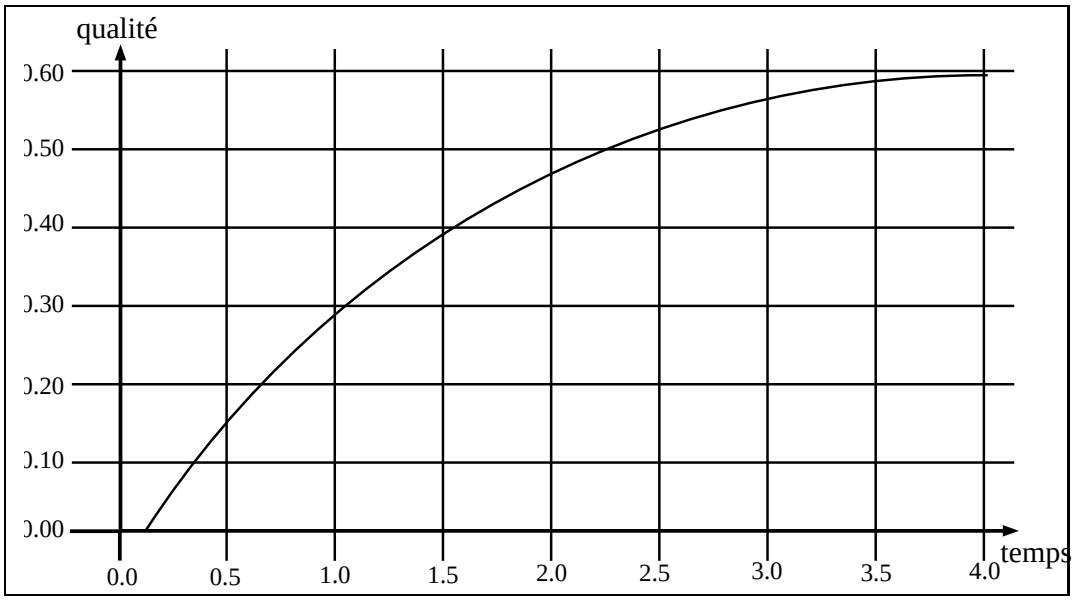


FIG. 2.2 – *Profil de performance standard*

entrée de l'algorithme anytime.

2.2.4 Exemple d'application des algorithmes anytime : Le problème du voyageur de commerce

Le problème du voyageur de commerce met en scène un voyageur devant visiter N villes différentes en ne passant qu'une seule et unique fois par ville avec de surcroît la contrainte de commencer et terminer par la même ville. Une fonction de coût permet de mesurer le coût pour aller directement d'une ville i vers une ville j. Il s'agit de trouver un chemin optimal permettant de visiter l'ensemble des villes avec un coût minimal. Le problème du voyageur de commerce est connu comme étant NP-complet, car il devient impossible de trouver un chemin optimal en temps raisonnable lorsque le nombre de villes est grand.

L'application du principe des algorithmes anytime à ce problème permet d'obtenir une solution dont la qualité sera fonction du temps alloué pour sa résolution. Dans [Zilberstein and Russell, 1996], l'algorithme anytime pour la résolution du problème du voyageur de commerce est traité de la façon suivante : dans un premier temps, le trajet du voyageur de commerce est initialisé aléatoirement. Ensuite, lors de chaque itération, l'algorithme cherche à améliorer partiellement son graphe en recherchant aléatoirement des arcs qui permettraient de réduire le coût du trajet. Lors de chaque itération, une fonction de coût permet de vérifier si la modification proposée permet l'amélioration de la qualité du résultat. Si la nouvelle solution ne permet pas d'améliorer la précédente alors la solution est abandonnée. L'algorithme détaillé est présenté dans la figure 2.4.

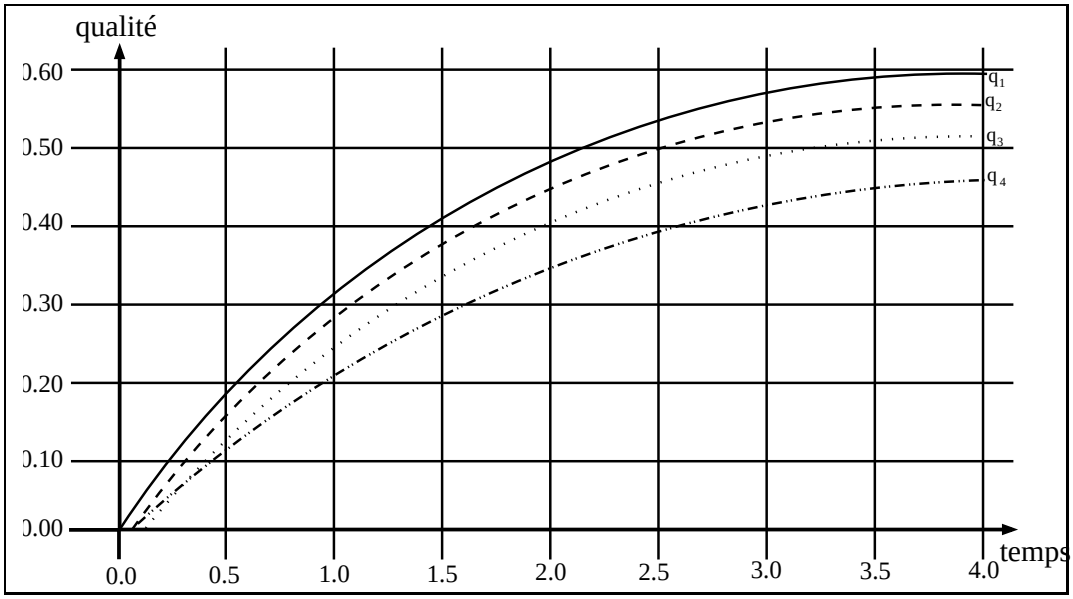


FIG. 2.3 – *Profil de performance conditionel*

```
ANYTIME-TSP (V, iter)
1    Tour <– INITIALISE-TOUR (V)
2    cout <– COUT (Tour)
3    ENREGISTRER-RESULTAT (Tour)
4    pour i <– 1 to iter
5        e1 <– RANDOM-EDGE (Tour)
6        e2 <– RANDOM-EDGE (Tour)
7        ε <– COUT (Tour)- COUT (SWITCH (Tour, e1, e2))
8        si ε >0 alors
9            Tour<– SWITCH (Tour, e1, e2)
10           cout<–cout-ε
11           ENREGISTRER-RESULTAT (Tour)
12       finsi
13       SIGNAL (TERMINAISON)
14       ARRÊTER
```

FIG. 2.4 – *Un algorithme anytime pour le problème du voyageur de commerce*

2.2.5 Composabilité des algorithmes anytime

Dans certaines applications temps réel complexes, on a recours à la décomposition du système en modules élémentaires qui vont se communiquer leurs résultats depuis le module d'entrée jusqu'au module de sortie. Dans le cadre d'applications ayant des propriétés anytime, chacun des modules doit être élaboré lui-même au moyen d'algorithmes anytime. Chaque module est indispensable dans l'élaboration du résultat global de l'application. Ce résultat doit être fourni selon des caractéristiques anytime. Lorsqu'un module communique son résultat à un autre module la qualité du résultat est variable.

On a vu précédemment que la qualité du résultat d'un algorithme anytime dépend à la fois du temps alloué pour l'exécution de cet algorithme mais aussi de la qualité des éléments fournis en entrée. Dans le cadre d'un module anytime servant à la conception d'un système complexe, la qualité en entrée de ce module correspond très souvent à la qualité du résultat fourni en sortie par un autre module. Pour permettre à un système composé de plusieurs modules de fournir un résultat anytime, il faut donc tenir compte de ces différentes interactions entre les modules. La solution consiste à utiliser les profils de performances pour concevoir des applications anytime qui peuvent alors, après résolution d'un ensemble d'équation, résoudre un problème dans un temps donné. Les premiers travaux effectués dans ce sens par [Zilberstein, 1996] se basent sur une résolution de ces équations avant le début de l'exécution du système et la compilation des équations. Cette méthode restreint donc le champ d'application des algorithmes anytime.

2.2.6 Conclusion

Dans ce chapitre, nous avons commencé par présenter les système temps réel en général, et montré qu'ils ne répondent pas aux besoins des applications de prise de décision. Nous avons donc présenté une autre approche, que nous avons utilisée dans la suite de nos travaux : les algorithmes anytime, pour tenir compte de l'aspect temps réel. L'aspect gestion de données est lui assuré par les SGBD traditionnels. Les caractéristiques des algorithmes anytime ont été présentés ainsi qu'un exemple illustrant leur application : le problème du voyageur de commerce. Le problème de la composabilité des algorithmes et les pistes explorées pour résoudre des systèmes complexes nous permettent d'entrevoir la possibilité d'appliquer les algorithmes anytime au niveau des systèmes multi-agents. En effet, dans ce type de système, un agent qui serait caractérisé par des propriétés anytime intervient au sein d'un système vaste composé d'agents qui peuvent être à leur tour anytime. Ces agents s'échangent généralement leur résultat dont la qualité peut varier et entraîner des dégradations rapides dans un SMA composé d'agents anytime. En effet, les interactions étant diverses et nombreuses, il est très difficile de prévoir de façon simple les performances d'un système multi-agent composé d'agents anytime et d'orienter la qualité

du résultat. Dans nos travaux, nous avons tenté d'atténuer ce problème en effectuant des regroupements d'agents diminuant ainsi le nombre de communications.

Chapitre 3

Systèmes Multi-Agents

Résumé

Les agents autonomes et les systèmes multi-agents (SMA) représentent une nouvelle façon d'analyser, de concevoir et d'implanter des systèmes informatiques complexes. Les agents sont utilisés dans une variété d'applications sans cesse croissante. Lorsque l'on aborde ce domaine, il est sans doute nécessaire de commencer par définir les termes « agents », « systèmes à base d'agents » et « systèmes multi-agents ». Or, lorsque l'on recherche la définition de ces termes dans la littérature, on s'aperçoit que les définitions sont sensiblement différentes selon les auteurs. Nous utiliserons celle qui est le plus communément admise par la communauté des chercheurs français. Les travaux sur les systèmes multi-agents ont vu naître un grand nombre de sous-domaines de recherche qui sont parfois en contradiction les uns avec les autres ou se complètent pour former une théorie plus vaste. La classe des applications concernées par les SMA s'élargit de plus en plus. Pour certains chercheurs, il s'agit d'une nouvelle étape dans la modélisation des applications, qui suit celle du paradigme objet. Nous n'entrerons pas dans ce débat mais nous nous contenterons de présenter le domaine des SMA dans ce chapitre.

3.1 Introduction

Les systèmes multi-agents (SMA) sont des systèmes informatiques distribués. Comme la plupart des systèmes distribués, ils sont composés d'entités informatiques qui interagissent entre elles. A la différence des systèmes distribués classiques, les entités qui les constituent sont « intelligentes ». En effet le domaine des systèmes multi-agents est issu du domaine de l'Intelligence Artificielle Distribuée (IAD). Il s'agit ici d'un des axes principaux du domaine de recherche décrit dans ce document. Nous allons dans ce chapitre décrire les concepts qui découlent du domaine de recherche que constituent les systèmes multi-agents. Nous donnerons un état d'avancement des travaux sur ce domaine avant de

conclure sur les possibilités et les voies de recherche qu'offrent ce domaine. Bien sûr, nous aborderons le domaine des systèmes multi-agents temps réel qui concerne directement nos travaux et qui fera l'objet du chapitre 5.

3.2 Concepts et définitions

Cette première section va nous permettre de présenter l'ensemble des concepts nécessaires à l'étude du domaine des systèmes multi-agents, qui sont issus du domaine de l'Intelligence Artificielle Distribuée. Ce domaine permet de modéliser un système informatique sous forme d'entités autonomes qui vont coopérer pour résoudre un problème complexe difficile à traiter de façon centralisée. Les agents autonomes et les systèmes multi-agents ont donné lieu à une nouvelle façon de voir, d'analyser, de modéliser et de concevoir les systèmes informatiques complexes. Les systèmes multi-agents reposent sur les concepts d'agents, d'organisation d'agents, d'interaction entre agents et sur la communication entre eux. Nous allons présenter dans cette section l'ensemble de ces concepts tels qu'ils ont été définis dans la littérature.

3.2.1 La notion d'agent

L'agent représente le composant de base des systèmes multi-agents mais a donné lieu à des utilisations très diverses. La définition la plus communément admise en France est celle donnée par Jacques Ferber [Ferber, 1995] et reprise par Alain Cardon [Cardon, 2000] : « On appelle agent une entité réelle ou abstraite qui est capable d'agir sur elle-même et sur son environnment, qui dispose d'une représentation partielle de cet environnement, qui, dans un univers multi-agent, peut communiquer avec d'autres agents et dont le comportement est la conséquence de ses observations, de sa connaissance et des interactions avec les autres agents ». Cette définition très large pose la notion d'agent comme une entité d'action définie à la construction d'un système et qui opère dans le cadre d'un problème distribué bien posé.

Bien que la définition donnée par J. Ferber soit reprise par A. Cardon, nous pouvons voir qu'elle aboutit à des principes de modélisation et de conception de systèmes multi-agents complètement différents. Dans le premier cas, nous pouvons citer les travaux de A. Drogoul [Drogoul, 1993] qui sont basés sur le principe d'une distribution naturelle du problème posé : un agent correspond à une entité du monde réel. Dans le second cas, on citera les travaux d'A. Cardon [Cardon, 1998a] [Cardon, 1997] [Cardon and Durand, 1997] [Cardon and Lesage, 1997] et de Franck Lesage [Cardon and Lesage, 1998] [Lesage, 2000]. Dans ces derniers travaux, on va chercher à identitifier des entités qui ne font pas partie du monde réel mais qui vont permettre une agentification du problème. Cette approche

peut permettre d'envisager l'agentification de tout problème complexe par identification de structures de traitement plus simples.

Récemment, une autre définition a été proposée par Jennings, Wooldridge et Sycara dans [Jennings et al., 1998] : « Un agent est un système informatique, situé dans son environnement, et qui agit de façon autonome et flexible pour atteindre les objectifs pour lesquels il a été conçu ». Par situé, ils entendent que l'agent reçoit des informations de son environnement par des capteurs et qu'il peut agir sur ce dernier afin de le modifier. L'autonomie d'un agent signifie qu'il peut agir sans avoir besoin de l'intervention d'un humain ou d'un autre agent et qu'il a le contrôle de ses actions et de ses états internes. Enfin, un agent est flexible signifie [Wooldridge and Jennings, 1995] que :

- il est capable de percevoir des changements dans son environnement et d'y répondre en temps limité (réactif) ;
- ses actions ne se limitent pas aux réponses à des modifications d'environnement, mais qu'il doit faire preuve d'opportunisme, de comportement dirigé par des buts et qu'il doit pouvoir prendre des initiatives au moment opportun (pro-activité) ;
- il doit interagir avec des humains ou d'autres agents artificiels si nécessaire afin de résoudre ses propres problèmes et qu'il doit être capable d'aider les autres dans leurs activités (social).

À partir de ces définitions et pour résumer, on peut extraire plusieurs caractérisques ou propriétés d'un agent. Un agent est :

- *situé* : il évolue dans un environnement ;
- *autonome* : il a la capacité d'agir sans aucune intervention extérieure ;
- *actif* : Il est :
  - *réactif* : il réagit aux changements de son environnement dans un temps limité,
  - *proactif* : il est capable de prendre des initiatives sans qu'on le sollicite,
  - *social* : il communique avec le monde qui l'entoure (des êtres humains ou d'autres agents.)

3.2.2 Une organisation d'agents

Un agent existe rarement seul : il fait partie d'une organisation dans laquelle il interagit avec d'autres agents. Nous allons d'abord donner une définition d'une organisation au sens général avant de préciser ce que l'on entend par une organisation d'agents.

3.2.2.1 Une organisation : définition générale

Le terme d'organisation est employé dans des contextes très divers. Il peut s'agir par exemple de l'organisation d'une conférence scientifique ou encore on peut l'employer en

parlant d'une organisation politique ou de tout autre type d'organisation. Attachons-nous au second emploi d'organisation qui correspond le plus souvent à un regroupement de personnes. Ce qui caractérise alors les organisations de ce type, c'est le regroupement de personnes ayant un objectif commun.

3.2.2.2 Une organisation d'agents : définition

En général, les systèmes multi-agents sont constitués d'un grand nombre d'agents. Ces agents forment des organisations. Une organisation d'agents est donc un ensemble d'agents qui sont agrégés autour d'un même niveau de granularité de connaissances. Cet ensemble peut évoluer au cours du temps tant en quantité d'agents qu'en qualité (c'est-à-dire que la nature des agents peut changer, ils acquièrent de nouvelles connaissances, de nouvelles capacités). Une organisation d'agents peut donc évoluer au cours du temps : s'agrandir, s'adapter aux situations, etc.

3.2.3 Interactions entre agents

La notion d'interaction est au centre de la problèmatique des SMA [Ferber, 1995]. Une interaction est une mise en relation dynamique de deux ou plusieurs agents par le biais d'un ensemble d'actions réciproques. Les interactions s'expriment ainsi à partir d'une série d'actions dont les conséquences exercent en retour une influence sur le comportement des futurs agents. Les situations sont diverses et variées : l'aide d'un robot par un autre, l'échange de données entre serveurs informatiques, l'utilisation d'une imprimante par deux programmes simultanément, la répartition de charge sur plusieurs processeurs, etc. Les interactions sont non seulement la conséquence d'actions effectuées par plusieurs agents en même temps, mais aussi l'élément nécessaire à la constitution d'organisations sociales.

3.2.4 La communication entre agents

La communication dans les systèmes multi-agents est à la base des interactions et de la création des organisations dont nous avons parlé précédemment. Sans la communication, un agent n'est qu'un individu isolé, sourd et muet qui ne fait qu'agir sur lui-même. La communication permet aux agents d'échanger des informations, des demandes de service, etc. Elle permet aussi à un agent d'agir sur un autre en lui fournissant des informations qui auront pour conséquence la remise en question de son comportement ou encore en lui demandant de modifier son comportement.

La communication entre agents peut revêtir diverses formes. Des langages tels que KQML [1], ACL [2], etc. ont été élaborés afin de normaliser la communication entre agents.

1. Knowledge Query and Manipulation Language
2. Agent Communication Language

Lorsque deux agents vont communiquer, leur objectif est d'échanger des informations mais avant tout de se comprendre. Donc, quelque soit le langage ou la forme de communication, l'importance réside dans la possibilité pour un agent de pouvoir comprendre les autres agents. Dans la majorité des cas, la communication se fait par envoi de messages et parfois par envoi de signaux ou stimuli dans l'environnement [Drogoul, 1993]. Un exemple de ce type de langage (KQML) est donné dans la suite.

Le langage KQML

KQML (Knowledge Query and Manipulation Language) est un langage issu des travaux de la Knowledge Sharing Effort (KSE) soutenue par l'ARPA (Advance Research Projects Agency) [ARPA, 1993]. Il s'agit d'un langage qui vise à définir un ensemble d'actes de langages qui soient standards et utiles. Ces actes de langages, appelés aussi performatifs, sont utilisés par les agents pour échanger des informations.

Le langage KQML [Finin et al., 1994] a été proposé pour supporter la communication inter-agents. Ce langage définit un ensemble de types de messages (appelés abusivement « performatifs ») et des règles définissent les comportements suggérés pour les agents qui reçoivent ces messages. Les types de messages KQML sont de natures diverses : simples requêtes et assertions (exemples : « ask », « tell ») ; instructions de routage de l'information (« forward » et « broadcast ») ; commandes persistantes (« subscribe », « monitor ») ; commandes qui permettent aux agents consommateurs de demander à des agents intermédiaires de trouver les agents fournisseurs pertinents (« advertise », « recommend », « recruit » and « broker »).

Comme le font remarquer Cohen et Levesque [Cohen and Levesque, 1995], ce langage a été développé de façon ad-hoc pour les besoins des développeurs d'agents logiciels : le terme "performatif" a été utilisé pour nommer diverses commandes qui ont une certaine ressemblance avec des verbes utilisés de façon performative dans le langage naturel ; l'interprétation sémantique actuelle de ces commandes n'est pas satisfaisante. Il serait approprié de réviser le langage KQML en utilisant un cadre théorique d'interprétation des actes de discours tel que proposé par exemple dans les travaux de Vongkasem et Chaib-draa [Vongkasem and Chaib-draa, 1999] [Chaib-draa, 1999].

3.3 Principe de programmation

Dans la première partie de ce chapitre, nous avons présenté les différentes notions concernant les agents et les systèmes multi-agents. Ces notions sont celles sur lesquelles nous fondons nos travaux. Elles peuvent apparaître assez abstraites et doivent donc être ramenées dans un cadre beaucoup plus concret qu'est celui de la conception d'applications informatiques. Dans cette section, nous décrivons ce que peut être la programmation de

systèmes multi-agents au travers de la notion de programmation orientée agent et du concept de plate-forme de développement de SMA.

3.3.1 Programmation orientée agent

Les applications reposant sur le paradigme multi-agent doivent permettre d'exprimer des caractères radicaux, tels que la réification de la notion d'émergence, la propriété de reproduction auto-contrôlée de groupes d'agents et le comportement non linéaire [Cardon, 1998b]. La conception d'applications informatiques relève aujourd'hui d'une démarche basée sur les principes du Génie Logiciel dans son approche orientée objets. Il existe en effet de nombreux langages à objets et des méthodes telles que UML [Lai, 1997] [Muller, 1997] utilisent le concept objet comme entité de base de la modélisation.

La démarche de modélisation par objets se fonde sur un a priori qui restreint le domaine des systèmes que l'on y modélise. En effet, le système est, dans l'approche objet, considéré comme *fonctionnellement bien décomposable* [Cardon, 1996]. Or, dans les systèmes basés sur le paradigme agent, on aborde la conception de systèmes complexes qui ne sont pas *fonctionnellement bien décomposables*. Le problème qui se pose alors est celui de l'*agentification*. Ce terme désigne le passage de la modélisation d'un système reposant sur le paradigme agent vers la réalisation de ce système. Quelques méthodes tentent d'apporter des réponses à la modélisation d'un système basé sur le paradigme agent [Collinot et al., 1996] [Gutknecht and Ferber, 1997].

Pour en terminer avec la programmation orientée agent, nous citerons les travaux de Yoav Shoham [Shoham, 1993] qui propose un langage de programmation orienté agent de la même manière qu'il existe des langages de programmation orientés objets. Il s'agit de considérer que l'on programme des agents à l'aide d'éléments de haut niveau tels que des buts, des choix, des compétences, des croyances, etc. et que les types de messages que les agents s'échangent se réfèrent à des mécanismes de communication de haut niveau eux aussi, en définissant des messages d'informations, de requête, d'offre, de promesse, de refus, d'acceptation, etc. Malgré tous ces travaux, il n'existe pas encore de langage de programmation orienté agent qui serait l'équivalent de langages tels que C++ ou Java dans le monde des langages orientés objets.

3.3.2 Plates-formes de développement de systèmes multi-agents

De la même façon que pour la conception de systèmes utilisant le paradigme objet, on a vu appaître des outils de développement rapides pour la conception de systèmes reposant sur l'utilisation du paradigme agent. Des plate-formes génériques sont elles-aussi apparues. Après avoir examiné plusieurs de ces plaformes, très peu apparaissent comme pouvant servir réellement à la conception de tout type d'applications. En effet, la plupart

s'adapte souvent à la conception d'un type d'applications ou d'une classe d'applications mais rarement plus. Il apparaît, de plus, que la définition d'un agent est beaucoup trop vague pour qu'un modèle générique et universel puisse exister.

Parmi les plate-formes les plus connues on peut citer SWARM [Burkhart, 1994] [Daniels, 1999] [Daniels, 2000], Voyager (`http ://www.objectspace.com/products/voyager/`), DIMA [Guessoum, 1996], MadKit [MADKIT, 2001], Zeus [Azvine et al., 2000], etc. SWARM est une plate-forme de développement qui possède une forte orientation vers le développement d'applications de simulation à événements discrets de systèmes complexes. Elle a été développée en Objective C et un portage vers Java est en cours. De nombreuses applications de simulations reposent sur l'utilisation de la plate-forme SWARM, en voici quelques exemples :

- Heatbugs : un modèle temporel d'agents locaux interagissant au travers d'un environnement pour créer un comportement global complexe.
- Mousetrap fission : une simulation à événements discrets de fission nucléaire.
- Market : un petit modèle de marché pour une matière première.

De nombreux autres exemples d'applications basées sur SWARM [Booth, 1997] [Kohler and Carr, 1996] [Luna and Stefannson, 2000] [Downing and Zvirinsky, 1999] peuvent être trouvés à l'adresse `http ://www.swarm.org/`.

Voyager est une plate-forme de développement commerciale. Elle fonctionne en environnement distribué et repose sur le standard Corba. Son domaine d'application est essentiellement orienté vers le commerce électronique et les serveurs d'applications WEB.

DIMA (Développement et Implémentation de systèmes Multi-Agents) a été développée par Zahia Guessoum dans le cadre de ses travaux de thèse [Guessoum, 1996]. L'architecture des agents qui a été choisie pour le développement de cette plate-forme est une architecture modulaire (module de surpervision, module de communication, module de perception, module de raisonnement).

Zeus est une plate-forme de développement de SMA générique, personnalisable et pourrait être augmentée par l'adjonction de nouveaux composants. Elle est présente sous forme d'un ensemble de classes implémentées dans le langage Java. En cela, elle est complètement portable et peut être utilisée sur tout système disposant d'une machine virtuelle Java. Un agent est composé de plusieurs objets et *threads*. Ces classes sont catégorisées dans trois groupes fonctionnels : une librairie de composants, un outil de développement et un outil de visualisation. De par sa généricité, elle peut être utilisée dans la conception d'un grand nombre d'applications mais en retour nécessitera des augmentations assez nombreuses et spécifiques à chaque type d'application.

On peut trouver une liste recensant un grand nombre de plate-formes de développement à l'adresse URL `http ://www.agentbuilder.com/AgentTools/index.html`. Pour nos travaux, nous avons néanmoins décidé d'utiliser une plate-forme générique car celle-ci

répond particulièrement bien à nos besoins. Cette plate-forme, nommée MadKit [Gutknecht and Ferber, 1997] [MADKIT, 2001], repose sur la notion d'agents légers (voir paragraphe 3.2.1 et [Wooldridge and Jennings, 1995]). Elle est présentée en annexe A.

3.4 Domaines d'application des systèmes multi-agents

Les systèmes multi-agents étant issus du domaine de l'intelligence artificielle distribuée, ils permettent de modéliser des applications où une modélisation classique est inadéquate, et où le système est souvent naturellement distribué. Si un problème n'est pas naturellement distribué, on peut choisir de le distribuer pour des raisons de simplification. C'est alors à la charge du concepteur de trouver la meilleure distribution possible afin qu'elle corresponde au problème qu'il souhaite résoudre. Les applications qui sont naturellement distribuées sont, par exemple, celles concernant la simulation d'écosystèmes où l'on associe à chaque individu un agent [Drogoul, 1993] [Farrell and Arthur, 1998] [Prietula et al., 1998] [Axelrod, 1997] [Deangelis and Gross, 1992].

3.4.1 Écosystèmes et modèles individus centrés

Les modèles individus centrés, c'est-à-dire orientés individus sont des simulations basées sur les conséquences globales d'interactions locales entre membres d'une population. Ces individus peuvent représenter des plantes et des animaux dans un éco-système, des véhicules dans la circulation, des personnes dans une foule, ou des personnages autonomes dans une animation ou un jeu. Ces modèles consistent typiquement en un environnement ou un cadre global dans lequel les interactions se passent et un certain nombre d'individus definis par leur comportement (régles comportementales) et des paramètres caractéristiques. Dans un modèle orienté-individu, les caractéristiques de chaque individu peuvent être suivies de façon continue. Ceci contraste avec les techniques de modélisation où le modèle tente de simuler des changements dans les caractéristiques moyennées d'une population globale. Les modèles orientés individus sont aussi appelés "orientés entités" ou "orientés-agents", et on parle de simulation multi-agent.

Dans ces systèmes, un agent (individu) correspond toujours à une entité du monde réel (un animal, une personne, un objet, une bactérie, etc.). De nombreuses applications de simulation utilisent des SMA basés sur un modèle individu centré [Booth, 1997] [Luna and Stefannson, 2000] [Downing and Zvirinsky, 1999] [Kohler and Carr, 1996]. Des plate-formes telles que SWARM leur sont parfois complètement dédiées.

3.4.2 Systèmes complexes

Les systèmes complexes sont des systèmes où les techniques de modélisation classique sont difficilement utilisables. En effet, dans ces systèmes, les paramètres sont beaucoup trop nombreux ou contradictoires pour pouvoir être pris en compte. Parfois même, il n'est pas possible de connaître l'ensemble des paramètres qui interviennent pour la modélisation. L'approche multi-agent permet alors d'avoir recours à une modélisation locale. Cette modélisation permet grâce aux principes d'émergence présents dans les SMA d'obtenir un système ayant les propriétés attendues. Il s'agit notamment de l'ensemble des systèmes permettant la simulation d'écosystèmes. Un bon exemple concerne la simulation d'une fourmilière [Drogoul, 1993]. En effet, lors de ses travaux de thèses, Alexis Drogoul a modélisé une foumilière en utilisant des agents pour modéliser les fourmis. Il a alors réussi à démontrer que l'on pouvait obtenir un but global qui était la survie de la fourmilière sans jamais avoir programmé cet élément dans le système mais uniquement à partir de l'interaction des agents par émergence organisationnelle.

3.4.3 Systèmes d'aide à la décision et SMA

Les systèmes d'aide à la décision (SAD) sont présents dans de nombreux domaines et ont pour objectif d'aider le décideur dans sa tâche en lui fournissant tous les éléments pertinents pour la prise de décision. Cela consiste très souvent à extraire de l'information depuis de multiples sources et à la traiter. Des traitements de l'information pour une prise de décision multi-critères sont alors nécessaires. Les systèmes multi-agents apparaissent comme étant bien adaptés pour traiter de l'information qui peut revêtir diverses formes et provenir de diverses sources. En effet, il faut pouvoir faire la corrélation entre l'ensemble des éléments obtenus pour les présenter à l'utilisateur. L'approche à base d'agents permet d'effectuer cette corrélation en utilisant la négociation et la coopération entre agents. Il s'agit donc d'un domaine d'application pour les SMA. Nous ne nous étendons pas plus sur les SAD puisqu'il ont fait l'objet d'un chapitre complet de ce mémoire.

3.4.4 Le domaine du commerce électronique et les agents du WEB

Le domaine du commerce électronique est un domaine en plein essor qui permet de favoriser les transactions commerciales [Collis and Lee, 1998]. Ce domaine utilise l'outil informatique et plus particulièrement les ressources mises à disposition par Internet pour rapprocher les acteurs commerciaux dans certains domaines. Ce domaine se rapproche de celui de l'aide à la décision et peut même être confondu avec celui-ci dans certaines circonstances car il est caractérisé par la mise en place de moyens permettant d'extraire de l'information sur les produits, les marchés, les acteurs du marché [Ghidini and Serafini, 1998]

[Eriksson et al., 1998].

L'utilisation de systèmes multi-agents est une bonne solution ici aussi car elle permet comme pour le domaine de l'aide à la décision de faire de la fusion d'informations. La dénomination « agents du WEB » désigne les agents qui circulent sur le réseau Internet pour extraire de l'information. Ces agents qui sont en général mobiles se déplacent au travers des sites WEB. Certains systèmes ont pour objectif de fournir au consommateur une valeur ajoutée en lui fournissant des informations supplémentaires afin qu'il fasse un choix approprié selon les critères qu'il s'est fixé [Guttman and Maes, 1998]. Il peut s'agir d'informations comparatives entre produits ou encore d'études qualitatives.

Prenons un exemple concret : vous souhaitez acheter une nouvelle voiture mais vous n'avez pas encore décidé du modèle exact. Jusqu'à maintenant, vous deviez visiter un certain nombre de concessionnaires, étudier des documentations techniques (parfois ardues), acheter des magazines spécialisés qui effectuent eux-mêmes des tests et des études comparatives. Tous ce travail est fastidieux et coûteux en temps. Le but des applications dans le domaine du commerce électronique va être d'extraire toute cette information qui est disponible à divers endroits du WEB et de n'en retenir que celle qui vous intéresse. Il va s'agir d'un travail d'extraction, de filtrage et d'analyse de l'information disponible. La multiplicité des sources d'informations disponibles sur le WEB et la complexité des traitements à effectuer rend particulièrement appropriée l'utilisation du paradigme agent pour le développement des applications de commerce électronique.

3.5 Conclusion

Dans ce chapitre, nous avons présenté les systèmes multi-agents et les concepts sur lesquels nous avons fondé nos travaux. De nombreux travaux ont été effectués concernant les architectures d'agents et de systèmes multi-agents. On parle désormais de conception orientée agent comme il avait été question en d'autres temps de conception orientée objet. Pour permettre la conception de systèmes multi-agents, des plate-formes de développement apparaissent un peu partout dans le monde. Certaines sont complétement génériques (MadKit, Zeus) et fournissent uniquement les mécanismes de base nécessaires au développement de SMA ; d'autres plate-formes sont plus spécialisées. Il s'agit par exemple de la plate-forme SWARM qui se spécialise dans les applications de simulation.

Après la domination de la technologie objet, la technologie agent semble prendre de plus en plus d'importance. En effet, elle permet de répondre aux besoins de nombreux domaines d'application jusque là difficilement abordables par des moyens tradionnels. Il s'agit notamment des systèmes complexes, des systèmes d'aide à la décision, du commerce électronique, etc.

Nous allons compléter cet état de l'art dans les deux chapitres suivants en présentant

les travaux qui nous intéressent plus particulièrement : les systèmes multi-agents distribués et les systèmes multi-agents temps réel.

Chapitre 4

Systèmes distribués et agents mobiles

Résumé

Nos travaux sur les systèmes multi-agents temps réel et plus particulièrement dans le cadre des systèmes d'aide à la décision nous ont amené à nous intéresser aux systèmes multi-agents physiquement distribués et à la notion d'agent mobile. En effet, les systèmes d'aide à décision font parfois intervenir des systèmes multi-participants. Lorsque de tels systèmes, distribués physiquement, reposent sur des architectures multi-agents, il arrive que deux systèmes multi-agents situés sur deux machines différentes doivent interagir. Dans ce cadre, il est alors nécessaire que deux agents sur des machines physiquement éloignées puissent communiquer au travers du réseau. La communication entre agents habituellement utilisée dans des environnements centralisés ne prévoit pas l'envoi de messages à un agent situé sur une machine différente. En effet, cela nécessite pour un agent donné d'être en mesure de le situer physiquement et de disposer de fonctions de communication étendues pour les agents distribués. La notion d'agents mobiles consiste à doter les agents de capacités de déplacement au travers de réseaux locaux ou globaux. Cette notion est utile dans les systèmes d'aide à la décision car elle permet de concevoir des agents qui ont la capacité de se déplacer au travers d'un réseau de systèmes d'information pour extraire de l'information utile pour la prise de décision.

4.1 La problématique de la distribution de systèmes multi-agents

Dans cette section, nous allons tenter de définir la problématique de la conception de systèmes multi-agents distribués. On peut distinguer deux domaines d'étude non disjoints :

- La distribution des systèmes multi-agents et leur fonctionnement en mode distribué.
- Les agents mobiles et la migration d'agents.

Le fonctionnement en mode distribué des systèmes multi-agents nécessite que les agents disposent de capacités particulières telles que des capacités de communications étendues. En effet, lorsqu'un agent envoie un message à un autre agent dans un même environnement cela ne pose pas de problèmes particuliers, alors que dans un environnement distribué, il faut connaître la localisation de l'agent destinataire (la machine sur laquelle il se trouve). Ce problème devient plus complexe si les agents ont la possibilité de migrer. Ce sont là les problèmes qui se posent pour la conception de SMA distribués et que nous allons examiner dans les paragraphes suivants.

4.1.1 La distribution d'agents et la communication entre agents distants

Comme nous l'avons déjà évoqué, les SMA distribués sont composés d'agents distribués qui doivent pouvoir communiquer ensemble. Il est donc question de distribution d'agents et de communication distante entre agents.

4.1.1.1 Distribution d'agents

La distribution d'agents consiste à placer des agents sur des machines différentes. Les raisons pour lesquelles des agents faisant partie d'un système plus vaste, doivent communiquer entre eux, sont diverses :

- ils font partie d'applications différentes devant communiquer entre elles. Il peut, par exemple, s'agir d'une application de gestion de commerce où chaque utilisateur se connecte individuellement et effectue ensuite des transactions avec d'autres utilisateurs distants. Le système de gestion de commerce électronique englobe alors l'ensemble des utilisateurs connectés via une interface identique mais vue par chaque utilisateur comme une application locale.
- il peut s'agir d'agents faisant partie d'une application qui a été artificiellement distribuée sur plusieurs sites afin de répartir la charge. On peut par exemple citer les applications ayant pour objectif de simuler des écosystèmes constitués de milliers, voire de millions d'individus.

4.1.1.2 Communication entre agents distribués

Nous avons déjà abordé le domaine de la communication de manière restrictive dans le chapitre concernant les systèmes multi-agents. Ici, nous allons étendre cette notion afin de considérer la distribution physique. Dans un SMA distribué, les agents qui constituent le système doivent pouvoir communiquer entre eux. Par exemple, un agent situé sur une machine A doit pouvoir envoyer un message à un agent situé sur une machine B. Pour mettre en œuvre cette fonctionnalité, il est nécessaire de disposer d'un certain nombre

d'outils permettant de situer un agent (sur quelle machine il se trouve), de transmettre des données au travers du réseau (les messages des agents), de pouvoir reconnaître l'origine physique d'un message, etc.

Un autre point essentiel déjà présent dans les systèmes non distribués concerne la compréhension entre les agents. En effet, lorsqu'un agent reçoit un message, il doit être en mesure de l'interpréter. Ceci reste vrai sur les systèmes distribués et soulève le problème de communication entre systèmes hétérogènes.

La communication entre agents distribués constitue donc l'un des problèmes essentiels de la distribution de SMA.

4.2 Les spécifications de l'OMG pour la conception de SMA distribués d'agents mobiles

L'OMG (Object Management Group) regroupe plus de 900 entreprises dans l'objectif de définir une norme pour la distribution d'objets. Cette norme est appelée CORBA (Common Object Request Broker Architecture) et facilite la communication et l'échange d'information entre objets distribués [Geib et al., 1997]. Dans cette section, nous nous intéressons plus particulièrement aux travaux sur la mobilité des agents qui ont été présentés dans [OMG, 1995].

4.2.1 Présentation

La normalisation des systèmes d'agents distribués fait partie du projet MASIF initié par l'OMG [OMG, 1997]. Les origines de ce projet sont liées à la constatation qu'il y avait trop de différences entre les systèmes d'agents mobiles existant. De plus, face à la prolifération de la technologie agent, il était absolument nécessaire de promouvoir l'interopérabilité.

L'interopérabilité des systèmes consiste à normaliser le transfert d'agents, de classes et la gestion d'agents. S'il existe une similarité entre la source et la destination, il est alors plus facile de mettre en place l'interopérabilité des systèmes d'agents. Par contre, si deux systèmes d'agents sont radicalement différents, il est beaucoup plus difficile de mettre en place cette interopérabilité. Néanmoins, selon les travaux de l'OMG, une interopérabilité minimale est possible.

Cette interopérabilité est spécifiée sous la désignation MAF (Mobile Agent Interoperability Facility) dans les travaux de l'OMG.

4.2.2 Interopérabilité entre systèmes d'agents

Pour qu'une intéropérabilité entre systèmes d'agents soit possible, il faut mettre en place des normes dans ces systèmes. Ce qui doit être standardisé est :

- la gestion normalisée des agents (« Agent Management ») ;
- le transfert des agents (« Agent Transfert ») ;
- les agents et leur système de nommage (« Agent and Agent System Names ») ;
- le typage des systèmes d'agents (Agent System Type) ;
- la syntaxe du repérage des agents (« Location Syntax »).

4.2.2.1 La gestion normalisée des agents

Cette gestion consiste à utiliser un système d'administration des agents. De plus, il doit être possible de créer un agent à partir du nom d'une classe. Cela signifie une simplification de l'instanciation des agents. Il faut aussi rendre possible la suspension, la terminaison ou le réveil d'un agent. Chaque agent doit donc disposer d'un ensemble minimal de fonctions communes pour la création, la suspension, la terminaison et le réveil.

4.2.2.2 Le transfert des agents

Il est ici question de mettre en œuvre le transfert d'agents d'une machine à une autre dans le but de réduire le coût des communications entre agents en privilégiant leur proximité. Une autre raison que l'on peut mettre en avant pour le transfert d'agents est liée à la proximité nécessaire des ressources utilisées.

4.2.2.3 Les agents et leur système de nommage

Il s'agit ici de la normalisation du nom des agents et du système de nommage des agents ainsi que du repérage de la position des agents. On doit en effet pouvoir dire quelle est l'origine d'un agent à partir de son nom. De plus, on doit s'assurer de l'unicité des noms d'agents bien que n'ayant pas connaissance de l'ensemble des agents qui sont instanciés au travers du réseau.

4.2.2.4 Le typage des systèmes d'agents et le repérage des agents

Il est nécessaire de rendre plus facilement identifiables les agents et leur rôle, et d'assurer l'unicité du nom des types de systèmes d'agents. Cette notion rejoint donc la précédente.

4.2.3 Les concepts de base

Pour pouvoir appréhender de façon correcte le domaine des agents mobiles, il est nécessaire de connaître le vocabulaire utilisé. Voici un extrait du vocabulaire défini dans le document de l'OMG :

- Agent, Stationary Agent (agent sédentaire), Mobile Agent (agent mobile),
- Agent State (état de l'agent), Agent Execution State (état d'exécution ou état courant de l'agent),
- Agent Authority (Autorité de l'agent),
- Agent Name (nom de l'agent),
- Agent Location (position de l'agent),
- Agent System (système d'agents), Agent System Type (type du système d'agents),
- Agent System to Agent System Interconnection (Interconnexion des systèmes d'agent),
- Place/Regions.

4.2.3.1 Agent

Un agent est un programme informatique autonome qui possède son propre "thread" d'exécution. Il exécute des tâches de sa propre initiative.

4.2.3.2 Agent sédentaire

Les agents sédentaires sont rattachés au système d'instanciation. Ils utilisent des mécanismes de communication distante pour obtenir de l'information.

4.2.3.3 Agent mobile

Un agent mobile est un agent qui n'est pas rattaché au système responsable de son instanciation contrairement aux agents sédentaires. Il possède la capacié de se déplacer d'un système vers un autre au travers d'un réseau afin de se rapprocher des ressources qu'il utilise.

4.2.3.4 Etat de l'agent

Il s'agit des moyens dont dispose l'agent pour suspendre et reprendre son exécution.

4.2.3.5 Etat courant d'exécution de l'agent

Il s'agit tout simplement de l'état d'exécution dans lequel se trouve l'agent au moment où on l'interroge.

4.2.3.6 Autorité de l'agent

Il s'agit de la personne ou de l'organisation pour laquelle l'agent travaille.

4.2.3.7 Nom de l'agent

Un nom d'agent est unique. Cette propriété est contrôlée par le système de nommage.

4.2.3.8 Position de l'agent

Il doit être possible à tout moment de retrouver un agent et de savoir le lieu où il se trouve (la machine sur laquelle il s'exécute) : c'est la position de l'agent.

4.2.3.9 Système d'agents

Il s'agit de la plate-forme d'exécution des agents (voir figure 4.1). Plusieurs systèmes d'agents peuvent coexister sur une même machine.

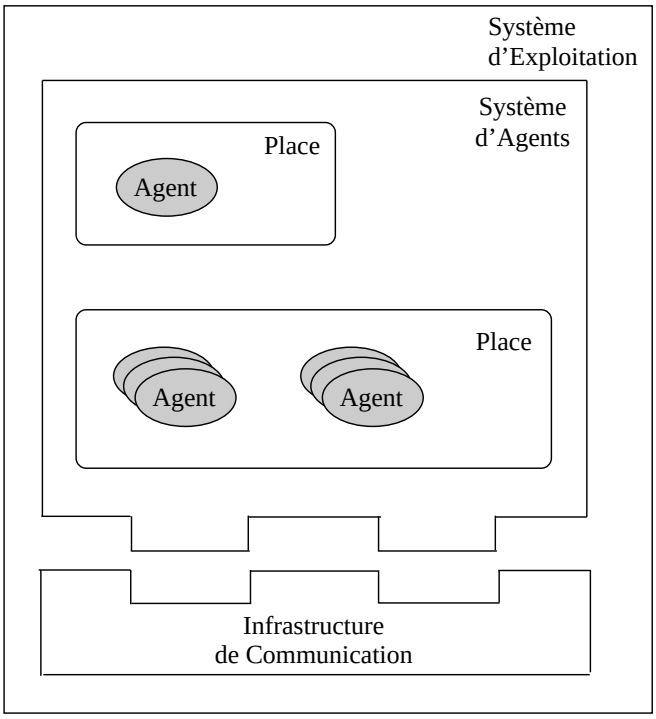


FIG. 4.1 – *Système d'agents*

4.2.3.10 Typage des systèmes d'agents

Il décrit le profil d'un agent, c'est-à-dire le langage utilisé, la méthode de sérialisation qu'il implémente, son type, etc.

4.2.3.11 Interconnexion des systèmes d'agents

Toutes les communications entre systèmes d'agents se font au travers de l'infrastructure de communication. L'administrateur de régions définit les services de communication intra-région et inter-régions (voir figure 4.2).

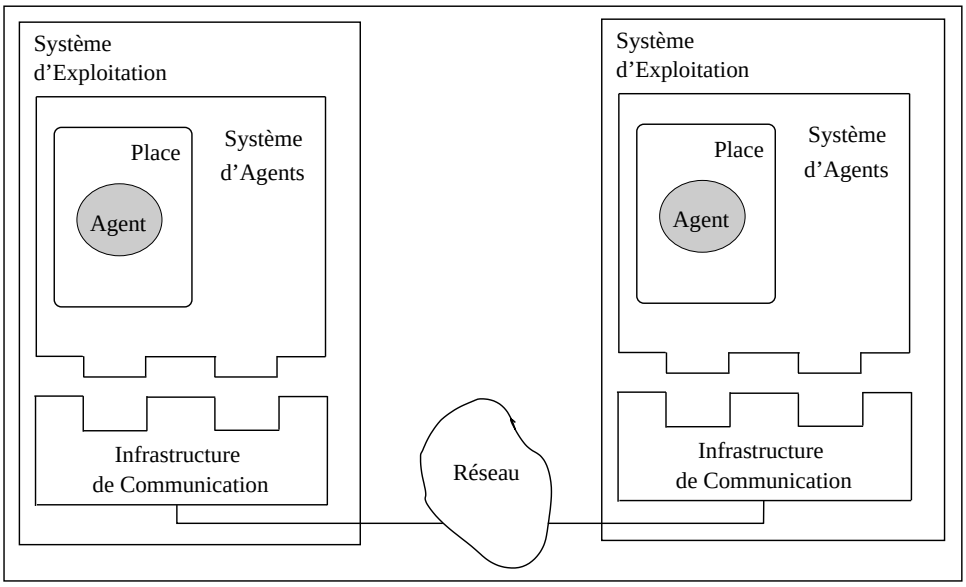


FIG. 4.2 – *Interconnexion de systèmes d'agents*

4.2.3.12 Place

Lorsqu'un agent est en cours de transfert, il voyage entre deux environnements d'exécution appelés Places. Une place est un contexte à l'intérieur d'un système d'agents dans lequel un agent peut s'exécuter.

4.2.3.13 Régions

Une région est un ensemble de systèmes d'agents qui ont la même autorité mais qui ne font pas nécessairement partie du même type de systèmes d'agents (voir figure 4.3).

4.3 Les plates-formes d'agents mobiles

4.3.1 Généralités

Il existe de nombreuses plates-formes de construction d'agents mobiles : *aglets* [Lange and Oshima, 1998], *concordia* [Castillo et al., 1998], *Gossip* (`http://www.tryllian.com/`), *Voyager* [Voyager, 1997]). Des informations sur ces plates-formes sont disponibles à l'adresse URL suivante : `http://www.agentbuilder.com/AgentTools/index.html`.

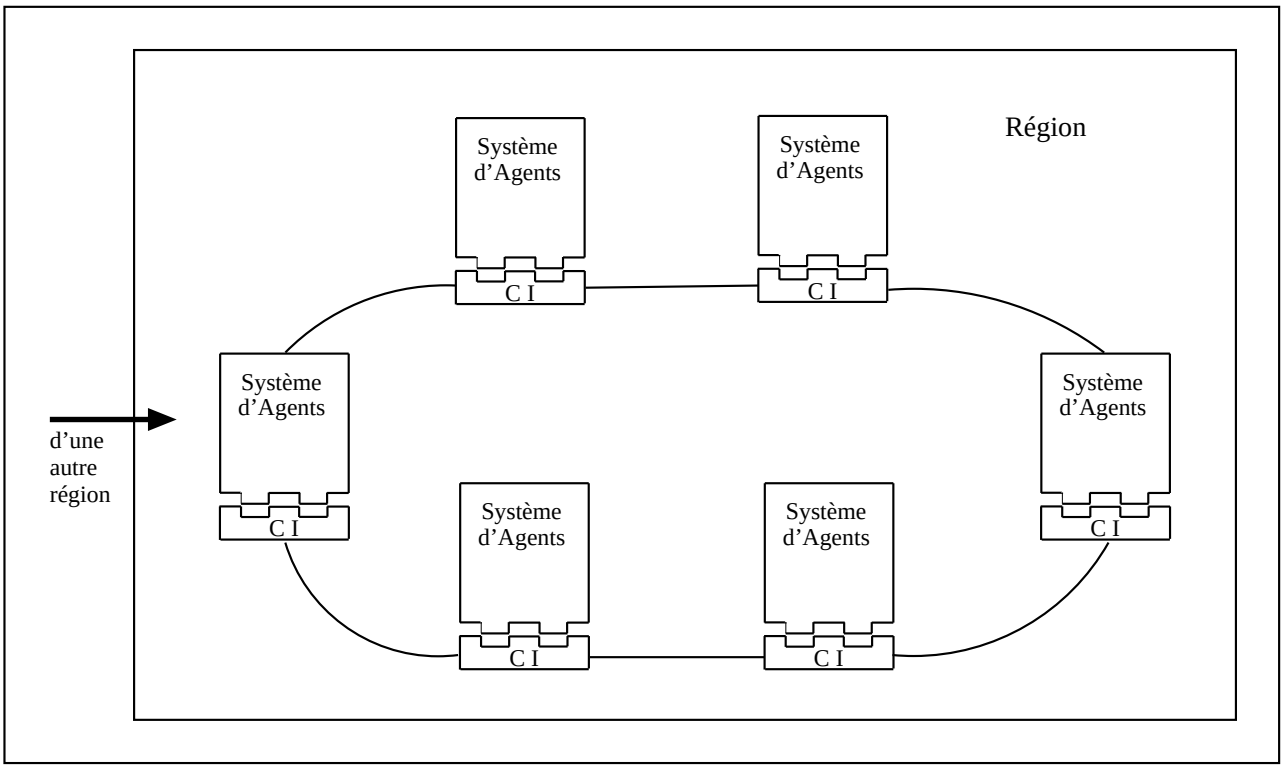


FIG. 4.3 – *Régions*

Certaines reposent en partie sur l'utilisation des spécifications de l'OMG. Il s'agit par exemple de la plate-forme *Grasshopper* (`http://www.infowin.org/ACTS/ANALYSYS/-PRODUCTS/THEMATIC/AGENTS/ch4/ch4.htm`). D'autres plates-formes, telles que MESSENGERS [Fukuda, 1997], reposent sur des approches plus personnelles.

4.3.2 MESSENGERS : une plate-forme d'agents mobiles

Nous présentons ici un exemple de plate-forme d'agents mobiles afin d'avoir un aperçu de la difficulté de mettre en œuvre ce type de notion.

4.3.2.1 Présentation

Le projet MESSENGERS a vu le jour à Irvine dans l'université de Californie. Il s'agit d'un système distribué basé sur le principe de messages autonomes (appelés Messengers). Un Messenger est capable de transporter son propre comportement au travers des nœuds d'un réseau. Sur chaque nœud, il peut s'exécuter de façon spécifique incluant l'invocation de fonctions résidant sur le nœud.

Dans MESSENGERS [Fukuda et al., 1999] [Wicke et al., 1998] [Fukuda, 1997], l'extraction et la restauration de l'état d'exécution d'un agent durant la migration sont essentielles. En règle générale, il est très difficile de faire migrer un agent car les spécifications

de déplacement peuvent être placées n'importe où dans le code. C'est pourquoi, la plupart des systèmes laissent la capture de l'état d'un agent aux soins du programmeur. Une approche intermédiaire consiste à utiliser des spécifications restreintes de développement au plus haut niveau de l'auto-migration de l'exécutif. Il s'agit alors de mettre en place une implémentation efficace de la capture et restauration d'états complètement transparente.

4.3.2.2 Les agents mobiles dans MESSENGERS

La définition utilisée dans MESSENGERS pour la notion d'agent mobile est la suivante : « les agents mobiles sont des entités autonomes qui peuvent se déplacer au travers d'un réseau et accomplir des tâches sur les nœuds qu'elles visitent ». Ce modèle d'agent est basé sur un environnement "multi-threadé" augmenté de capacités de navigation. Un agent est donc composé d'un programme, d'un état, de capacités de communication via des mécanismes de mémoire partagée.

Les capacités de déplacement des agents mobiles leur permettent de se déplacer eux-mêmes ou d'envoyer une copie d'eux-mêmes à travers le réseau de telle façon que leur exécution puisse continuer dans un nouvel environnement. Pour réaliser des capacités de déplacement, une simple copie de programme n'est pas suffisante car cela provoque un redémarrage complet depuis le début de ce programme. Or, dans le cadre d'un agent, son contexte doit être sauvegardé pour être restauré lors de la reprise de son exécution. Il est nécessaire de disposer d'un mécanisme d'extraction de l'état courant d'exécution sur le nœud source et de restauration sur le nœud destination afin de continuer l'exécution comme si l'agent n'avait jamais migré.

4.3.2.3 La capture et la restauration d'états

En mode compilé, il est très difficile de sauvegarder et restaurer l'état d'un programme (état des registres et des flags) qui est du seul fait du processeur. En effet, le processeur connaît l'état d'avancement du programme et l'état de la mémoire. En mode interprété, le contexte d'exécution est maintenu par l'interpréteur ; il est donc plus facile à sauvegarder. En effet, on connaît l'état d'avancement du programme et ainsi on peut connaître quelles sont les variables modifiées depuis le début de l'exécution. L'objectif est de mettre en place des mécanismes d'extraction et de restauration des états de façon transparente pour l'utilisateur. Ceci nécessite d'étendre de façon importante les fonctions des compilateurs et des interpréteurs.

4.3.2.4 Le système MESSENGERS

MESSENGERS est un système supportant le développement et l'utilisation d'applications distribuées structurées en collection d'objets autonomes Messengers. Les Messengers

sont des scripts comportant plusieurs parties plus ou moins indépendantes :

- La partie constituant le corps du programme : elle permet d'effectuer les évaluations générales ; elle est écrite dans un langage C restreint.
- La partie concernant la spécification du déplacement du Messenger : elle permet de doter les agents de capacités de déplacement au travers du réseau.
- La partie concernant les invocations de fonctions externes : elle fournit une interface à l'environnement du système ; ce qui permet le chargement et l'invocation arbitraire de fonctions précompilées en C restreint.
- La partie concernant les spécifications de synchronisation : elle permet d'attendre l'arrivée de certaines conditions (des signaux) ou d'envoyer des signaux.

Dans MESSENGERS, un Messenger peut être préempté uniquement s'il n'effectue pas un déplacement, une opération de synchronisation (bloquante) ou s'il redonne la main volontairement.

Il existe deux types de variables :

- les variables de nœud qui sont stationnaires dans n'importe quel nœud logique,
- les variables messengers qui sont transportées par des Messengers.

La distinction entre le code au niveau script de chaque Messenger et les fonctions C qu'il invoque est une propriété intrinsèque des Messengers.

L'état d'un Messenger est automatiquement sauvegardé dans une structure de données qui peut alors être envoyée au travers du réseau. Extraire les variables locales n'est pas très difficile. Le problème principal est constitué par le déroulement du programme. En effet, il faut être en mesure de poursuivre l'exécution d'un agent après sa migration à l'endroit où il s'était arrêté, de façon à obtenir une continuité dans le déroulement de l'exécution de l'agent.

Capture d'états avec les Messengers compilés. La solution qui a été retenue est de structurer le Messenger en une collection de niveaux de scripts de fonctions de la façon suivante [Wicke et al., 1998] :

- Toutes les étapes du script original deviennent une partie de nouvelles fonctions.
- Toutes les fonctions sont numérotées et chacune connaît ses successeurs éventuels (i.e. les numéros des prochaines fonctions à exécuter).
- Chaque étape qui pourrait causer un changement de contexte doit être exécutée uniquement comme la dernière fonction.

4.4 Domaines d'application

De nombreux domaines d'application peuvent bénéficier de la technologie liée au développement de SMA distribués utilisant la technologie des agents mobiles. Il s'agit par exemple des agents chargés d'aller chercher de l'information sur le WEB ou dans des systèmes distribués. Un autre type d'application concerne l'utilisation de ressources de calcul intensif. En effet, on peut alors provoquer la distribution de l'application lorsqu'elle repose sur l'exploitation du paradigme agent. Ceci permet d'augmenter les capacités de calcul en utilisant une multitude de sites et de machines là où une machine unique même performante n'aurait pas été suffisante. Nous allons donner trois exemples de domaine d'application des systèmes multi-agents distribués dans cette section : les agents du WEB, les applications nécessitant des capacités de calcul importantes et les applications naturellement distribuées.

4.4.1 Les agents mobiles du WEB

De nombreuses applications concernant le WEB ont vu le jour, dont certaines utilisent la technologie des agents mobiles [DeLespinasse et al., 1995]. Il s'agit souvent d'applications ayant recours à des agents pour extraire des informations sur les différents sites qu'ils visitent. Cette technologie permet de réduire les coûts liés aux communications. En effet, les agents qui vont aller chercher les informations demandées vont alors traiter l'information sur place afin de ne ramener que celle qui apparaît correspondre à la demande. De cette façon, les canaux de communications sont beaucoup moins encombrés par des données superflues.

4.4.2 Les applications manipulant des quantités importantes d'agents

Certaines applications, notamment celles concernant la simulation d'écosystèmes (par exemple le projet Manta [Drogoul, 1993]) ou de particules de fluides [Olivier et al., 2000] requièrent l'utilisation d'un grand nombre d'agents. La charge en temps CPU devient très vite trop importante pour pouvoir être gérée par une machine unique ; la distribution physique et non plus uniquement logicielle des agents s'avère une solution adéquate. On répartit alors les agents sur plusieurs machines. La répartition initiale joue un rôle primordial car une mauvaise répartition augmente le nombre de communications entre les agents distants et occasionne une plus grande charge qui nuit à un fonctionnement optimal de l'application. L'utilisation de systèmes multi-agents distribués s'avère donc une solution adéquate permettant de répartir la charge initiale. L'utilisation d'agents mobiles peut permettre d'obtenir une répartition dynamique de cette charge en fonction de son

évolution au cours du temps.

4.4.3 Les applications naturellement distribuées

Certaines applications sont constituées de plusieurs systèmes localisés physiquement sur des machines différentes mais devant interagir entre eux. Il s'agit notamment des systèmes qui font appel à des composants externes (des systèmes qui existent individuellement mais qui peuvent être utilisés par d'autres), mais aussi des systèmes multi-utilisateurs (l'application existe en divers points d'un réseau et entre dans la constitution d'un système plus vaste). Un exemple d'application de ce type pourrait être celui d'un agenda servant à gérer des réunions multi-participants et permettant de trouver des dates de réunion disponibles. Un utilisateur possède cette application en local sur son poste et peut l'utiliser individuellement pour gérer son emploi du temps. Cet agenda peut aussi être utilisé par d'autres agendas pour connaître les disponibilités de l'utilisateur en vue d'une réunion multi-participants. Il s'agit dans ce type d'application de constituer une application à partir de plusieurs "briques" physiquement distantes mais communiquant entre elles.

4.5 Conclusion

Dans ce chapitre, nous avons donné un état de l'art sur le domaine des systèmes multi-agents distribués. Des travaux abordant l'aspect mobilité des agents ont aussi été évoqués et existent en nombre important. Très peu de travaux concernant la distribution concrète d'applications reposant sur le paradigme agent existent dans la littérature. Les travaux de l'OMG [OMG, 1997] ont défini les spécifications pour la conception de systèmes multi-agents distribués, notamment la réalisation de plates-formes interopérables. Certaines plates-formes permettant la conception de systèmes multi-agents distribués prennent d'ailleurs en compte ces spécifications qui ont pour principal objectif l'interopérabilité des systèmes multi-agents. Cette interopérabilité permettra à terme à des agents de se déplacer d'une plate-forme à une autre sans que pour cela les plate-formes aient été construites avec la connaissance préalable des agents qu'elles reçoivent. Ainsi on pourra construire des systèmes informatiques qui pourraient être complètement adaptatifs aux besoins de l'utilisateur. Il suffirait alors de faire appel à des agents possédant les compétences nécessaires pour faire évoluer les systèmes déjà en place. Ces agents pourraient provenir de sources très diverses dans la mesure où leur conception reposerait sur la réalisation des spécifications de l'OMG.

Dans nos travaux, nous allons nous intéresser à la réalisation d'une application de commerce électronique faisant intervenir de multiples utilisateurs. Il s'agit par conséquent d'une application naturellement distribuée qui devra reposer sur un SMA distribué. Nous

présenterons un modèle pour la réalisation de SMA distribués en nous inspirant des travaux que nous avons présentés dans ce chapitre. Cependant, l'aspect mobilité des agents ne sera pas pris en compte dans le cadre des travaux présentés dans ce mémoire. Ces travaux feront l'objet d'une extension future à notre système. Il nous a semblé néanmoins nécessaire de tenir compte de cette extension future pour la conception et la réalisation du modèle de distribution des agents. En effet certains aspects de la mobilité des agents sont déjà présents dans le modèle DISMAS. On peut par exemple citer le système de répérage et celui du nommage des agents.

Chapitre 5

Les Systèmes Multi-Agents temps réel

Résumé

La prise en compte du temps réel dans les Systèmes Multi-Agents consiste à garantir les délais d'exécution sur des systèmes qui ont comme principale caractéristique de ne pas avoir un comportement déterministe. En effet, un système multi-agent est composé de multiples entités qui interagissent afin de résoudre un problème ou du moins afin de produire une solution satisfaisante. Ces interactions se font selon le contexte d'exécution et ne sont jamais prévisibles. On ne peut pas savoir combien d'agents vont intervenir dans une résolution d'un problème car cela dépend du contexte d'exécution. Il s'avère donc d'autant plus difficile de connaître les temps d'exécution nécessaires pour cette résolution et par conséquent garantir un temps d'exécution dans ce contexte devient impossible. Dans ce chapitre, nous allons présenter les différents travaux qui ont été effectués pour prendre en compte l'aspect temps réel dans les systèmes multi-agents ou comme c'est le cas le plus souvent, dans les agents. Ces travaux sont essentiellement basés sur l'utilisation des algorithmes anytime et sur l'approche à base de tableaux noirs.

5.1 Approche anytime

Les techniques anytime sont une des approches utilisées pour la conception des Systèmes Multi-Agents Temps Réel. En effet, puisqu'il est très difficile, dans des applications basées sur des architectures multi-agents, de garantir le respect des contraintes de temps, l'approche anytime semble très appropriée puisqu'elle permet de fournir un résultat à n'importe quel instant de l'exécution du système. Nous avons déjà présenté dans un chapitre précédent les concepts de base concernant les algorithmes anytime. Dans cette section, nous présenterons les travaux effectués par Thierry Salvant [Salvant, 1998] [Salvant and Brunessaux, 1997] et Abdel-Illah Mouaddib [Mouaddib, 1993] [Mouaddib, 1997] [Mouaddib and Zilberstein, 1998] [Mouaddib and Zilberstein, 1997]

pour mettre en œuvre ces concepts dans les systèmes multi-agents.

CAAM : un modèle d'agents anytime

Ces travaux se situent dans le domaine de l'aide à la décision en temps critique. Ils sont focalisés sur la possibilité pour des agents de donner un résultat « à tout moment ». Thierry Salvant présente donc un modèle d'agent anytime nommé CAAM (Cooperative Anytime Agent Model) [Salvant, 1998] [Salvant and Brunessaux, 1997]. Ce modèle est basé sur l'utilisation de techniques anytime pour permettre l'exécution des tâches de l'agent avec des propriétés anytime. L'architecture d'agent utilisée ici repose sur quatre composants :

- les tâches de contrôle, qui possèdent deux rôles :
  1. S'assurer que les tâches anytime ne dépassent pas le temps qui leur a été alloué et créer de nouvelles tâches.
  2. Attendre l'arrivée de nouvelles tâches en provenance d'autres agents ; il s'agit là d'une fonction d'écoute.
- les tâches anytime du domaine : il s'agit en résumé de l'exécution d'un algorithme anytime ;
- l'horloge temps réel : il s'agit de l'élément essentiel permettant à l'agent de prendre en compte le temps dans ses raisonnements ;
- les ressources de calcul : chaque agent possède ses propres ressources de calcul qu'il ne partage avec aucun autre.

La remarque qui peut être faite sur les travaux de Thierry Salvant est la suivante : la prise en compte du temps réel n'est faite qu'au niveau de l'agent et non pas au niveau multi-agent ; ce n'est pas suffisant pour qu'un système basé sur cette architecture puisse fournir un résultat suivant des caractéristiques anytime, du moins dans les systèmes ayant recours à une population importante d'agents.

5.2 Approche à base de tableaux noirs

Cette partie décrit les systèmes reprenant l'approche blackboard ou tableau noir. Le système REAKT [Mouaddib, 1993] mis à part, les systèmes décrits sont en fait des extensions de systèmes à base de tableaux noirs existant pour un fonctionnement en temps réel. BB1-TR et RT1-TR mettent l'accent sur les capacités du système à adapter son cycle de contrôle à l'évolution de son environnement. DVMT-TR [Mouaddib, 1993] (page 69) vise plus particulièrement à intégrer des techniques de raisonnement approximatif. ATOME-TR [Mouaddib, 1993] met l'accent sur la réactivité du système ainsi que sur les capacités

d'adaptation de son raisonnement à la situation en intégrant des techniques de planification réactive. REAKT [Mouaddib, 1993] met l'accent sur les capacités du système à mener un raisonnement temporel, à adapter son raisonnement à la situation et à respecter les échéances. Enfin, RTSOS [Mouaddib, 1993] décrit une approche multiblackboard (sociétés de spécialistes) intégrant un mécanisme de raisonnement en temps contraint anytime.

5.2.1 Généralités sur l'approche à base de tableaux noirs

Dans les agents de type « blackboard », le contrôle et les connaissances permettant de raisonner sont regroupés en modules procéduraux ou à base de règles de production, appelés Source de Connaissances (SCs). Ces SCs communiquent via une structure de données commune appelée tableau noir ou blackboard. Ce tableau noir est la plupart du temps organisé en différentes couches et permet aux SCs de mener des raisonnements concurrents. Un ordonnanceur ou scheduler permet de faire de la résolution de conflit si plus d'une SC se déclenchent au même moment. Il existe principalement trois types de contrôle dans ces architectures :

- procédural (DVMT [Lesser and Corkill, 1983]), lorsqu'il est mis en œuvre par un programme procédural unique,
- hiérarchique (ATOME [Laasri and Laasri, 1988]), lorsque les connaissances de contrôle mises en œuvre sont organisées de façon hiérarchique,
- à base de tableaux noir (BB1 [Hayes-Roth, 1985]), lorsque ces mêmes connaissances sont codées par un ensemble de SCs interagissant à travers un second tableau noir.

5.2.2 BB1 temps réel

La version temps réel de BB1 [Hayes-Roth et al., 1989b] reprend l'approche de BB1 [Hayes-Roth et al., 1989a] et l'enrichit pour permettre un fonctionnement en temps réel. Cette version temps réel de BB1 a en effet été réalisée dans le cadre de l'application Guardian [Hayes-Roth et al., 1989a] visant à développer un système de surveillance de malades après une intervention chirurgicale à l'hôpital.

L'architecture générale BB1 temps réel est la suivante. BB1 est constitué de trois modules principaux : (1) un module de perception de l'environnement, (2) un module d'actions sur cet environnement et (3) un module de raisonnement (Figure 5.1). Ces trois modules communiquent par l'intermédiaire de tampons d'entrée/sortie qu'une interface de communication lit et remplit d'informations respectivement à envoyer ou reçues. Le module de perception reçoit des informations provenant de l'environnement et les filtre en fonction de directives données par le module raisonnement. En fonction de ce filtrage, certaines de ces informations sont mises à disposition de ce dernier ou envoyées directement

au module d'actions. Le module de raisonnement incorpore dans un blackboard les données disponibles dans son tampon d'entrée/sortie et provenant du module de perception. Ce blackboard constitue sa base de faits. Les raisonnements menés sur ces faits peuvent affecter des paramètres de contrôle pilotant l'acquisition de données ou générer des actions à mener sur l'environnement extérieur par le module d'actions. Le module d'actions récupère dans son tampon d'entrée/sortie les directives données par le module raisonnement et effectue les actions correspondantes sur l'environnement à l'aide d'actionneurs.

La communication entre ces modules est donc asynchrone et dépend directement, en termes de performances, de la bonne gestion de leurs tampons d'entrée/sortie respectifs. La taille de ces tampons est limitée mais accessible par l'utilisateur à l'initialisation. La manière d'extraire les données est liée à leur importance respective qui dépend de celle que leur donne l'agent, de la date d'arrivée de la donnée dans le tampon et de l'urgence avec laquelle l'agent souhaite la voir traitée.

5.2.2.1 Le module de raisonnement

Ce module reprend une architecture blackboard. Une mémoire globale comprend le blackboard du domaine, un blackboard de contrôle contenant un plan de contrôle, l'ensemble des Sources de Connaissances (SCs), un agenda des SCs activables, une liste d'événements et les tampons d'entrée/sortie. Ce module reprend et étend le cycle de contrôle de BB1. Il mène trois activités : (1) la gestion de l'agenda pour déterminer les SCs activables en fonction de triggers et préconditions associés et affecter une priorité en fonction des données du blackboard de contrôle, (2) l'ordonnancement des SCs activables en fonction de leur priorité associée et (3) l'exécution de la SC choisie par l'ordonnanceur. L'exécution d'une SC conduit à la modification du blackboard du domaine et/ou de contrôle. La modification consiste à borner le temps de ce cycle et en particulier à borner en temps l'activité complexe et couteuse de l'agenda manager. Elle est en effet complexe et coûteuse en temps car elle peut être représentée par une fonction croissante du nombre de SCs ainsi que du nombre de paramètres et d'événements à considérer pour fixer la priorité de ces SCs. C'est en tout cas au niveau de ce processus que BB1 est limité pour réaliser des raisonnements complexes impliquant de nombreuses SCs avec des contraintes de temps conséquentes. Dans une nouvelle approche, l'agenda manager adopte un comportement incrémental non exhaustif en n'identifiant qu'un nombre restreint de SCs activables. De façon simplifiée, cela consiste pour lui à ne chercher des SCs activables que pendant une durée maximale fixée ou jusqu'à ce qu'une action jugée suffisamment bonne ait été trouvée pour être exécutée, le « suffisamment bon » étant fonction de paramètres de contrôle qui peuvent évoluer au cours du temps. Le scheduler peut alors prendre l'action la plus prioritaire à mener.

L'algorithme adopté par l'agenda manager influence directement le temps nécessaire

pour obtenir une SC suffisamment bonne et est donc prépondérant. Il manipule les SCs et les événements par ordre d'importance et tente à chaque itération d'associer les SCs les plus importantes avec les événements les plus importants.

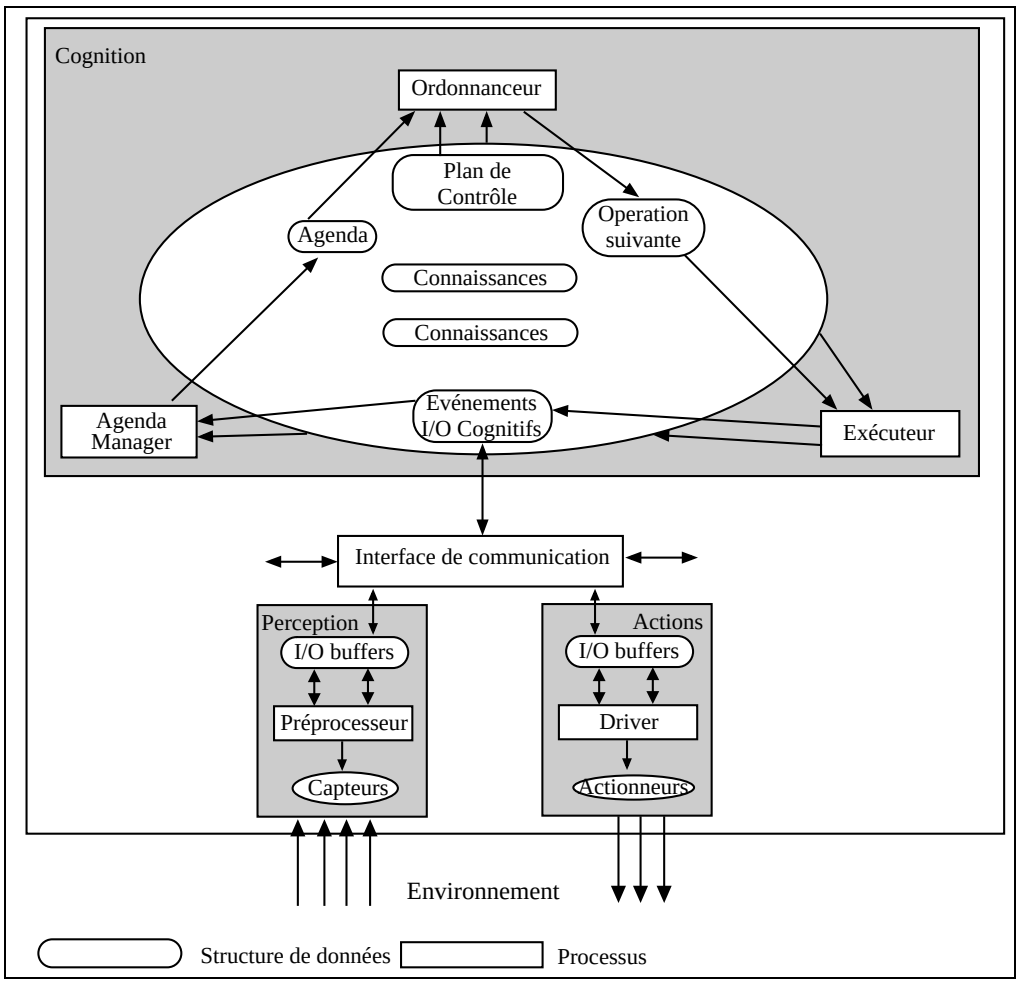


FIG. 5.1 – *Architecture de BB1 temps réel selon [Hayes-Roth et al., 1989b]*

5.2.2.2 Conclusion

Les mécanismes décrits sont destinés à permette au système d'adapter son raisonnement à l'évolution de l'environnement par l'intermédiaire de paramètres de contrôle ajustables, ceci, en particulier, au niveau du module de perception pour adapter son activité aux besoins du module de raisonnement. Mais ils permettent également au système de passer outre des possibilités de raisonnement et donc d'actions en ignorant des SCs activables moins importantes, permettant ainsi de construire des solutions dégradées.

La grande difficulté est cependant dans la définition de l'application pour des problèmes complexes et particulièrement pour arriver à associer correctement l'adaptation des paramètres de contrôle aux données du blackboard de contrôle.

5.2.3 RT1

RT1 [Dodhiawala et al., 1989] a été conçu dès le départ pour mener des raisonnements en temps réel. Il est basé sur une architecture blackboard avec une activité de contrôle adaptée de celle de BB1 [Hayes-Roth, 1985] et fonctionne sur un mode dirigé par les événements.

L'accent a été mis sur l'aspect performances et sur l'adaptation du raisonnement à l'évolution de l'environnement, en particulier sur la vitesse d'exécution des tâches, sur la réactivité du système pour prendre en compte rapidement les événements externes et internes, le respect des échéances et enfin la capacité du système à s'adapter suffisamment rapidement aux situations non prévues (en réévaluant la priorité des tâches en cours par exemple).

Le projet a été financé par la DARPA (Defense Advanced Research Projects Agency) et visait comme application le développement d'un copilote électronique.

5.2.3.1 Architecture générale

L'architecture blackboard sur laquelle repose RT1 met en œuvre des entités de raisonnements physiquement distribuées, fonctionnant de manière asynchrone et communiquant par données échangées par l'intermédiaire du blackboard. Celui-ci intègre à la fois les données de contrôle et du domaine, à la différence de BB1 où ces deux types de données sont stockées dans deux tableaux noirs séparés. Chaque entité de raisonnement est constituée d'un ensemble de SCs dont elle gère les activités en se basant sur les données de contrôle du tableau noir. Comme les entités, les SCs communiquent entre elles via le blackboard.

Les raisonnements sont déclenchés sur réception d'événements générés par le blackboard dès que des modifications significatives y ont été faites par une ou plusieurs SCs. Il existe deux catégories d'évènement : les événements internes à un module de raisonnement et les événements externes engendrés par un module pour un autre. La distribution physique se fait au niveau des entités de raisonnement exécutées sur différents processeurs. Chacune d'entre elles mène trois activités principales sur trois processus distincts : une activité de gestion des événements externes pour leur envoi et réception (processus entrées/sorties), une activité de gestion du tableau noir (en lecture, écriture et modification des objets) et enfin une activité de raisonnement proprement dite mettant en œuvre un cycle de contrôle pour exécuter les SCs sélectionnées. Ce cycle de contrôle prend en compte les événements internes et externes ainsi que des heuristiques de contrôle dynamiques pour la sélection de SCs à activer.

5.2.3.2 Le raisonnement dans RT1

Comme il a été dit, le raisonnement dans RT1 est géré par un cycle de contrôle inspiré de celui mis en œuvre dans BB1 (Figure 5.2). Il s'articule donc autour des mêmes étapes :

- identification, dès réception d'événements, des SCs dont le champ trigger est vérifié, création des instances correspondantes et rangement dans un agenda d'Instances de Source de Connaissances (ISC) sélectionnées ;
- mise à jour d'un agenda d'ISCs exécutables à partir de l'agenda des ISCs sélectionnées en identifiant celles dont les préconditions sont valides. Mise à jour également en éliminant les ISCs dont les conditions de rejet sont vérifiées ;
- attribution de priorités aux ISCs exécutables à l'aide d'heuristiques de contrôle et ordonnancement ;
- exécution de la SC associée à l'ISC prioritaire.

L'adaptation au niveau du cycle de contrôle par rapport à BB1 est faite de deux manières : d'une part en le dupliquant sur différents canaux d'importance prédéfinie et d'autre part en paramétrant le nombre de SCs exécutables à exécuter avant que celui-ci ne soit relancé.

L'objectif des canaux est de pouvoir associer aux événements des priorités de traitement en fonction de leur importance dans le raisonnement et d'associer un canal par niveau d'importance. Tout le raisonnement engendré par un événement d'une certaine importance est alors conduit sur le canal correspondant. Le système réalise en priorité le cycle des canaux de plus haut niveau ; ceci lui permet d'être plus réactif aux événements les plus importants. Le paramètre de contrôle est lui aussi lié à la réactivité du système. En effet, plus le nombre de SCs à exécuter avant que le cycle de contrôle ne soit relancé est petit, plus le système est réactif aux événements mais moins il est efficace, car le coût du contrôle nécessaire devient plus élevé.

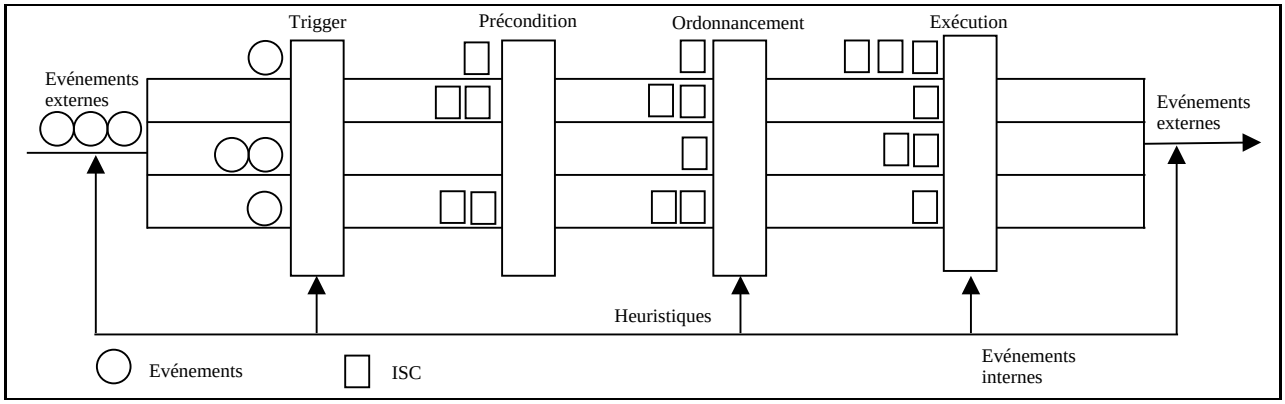


FIG. 5.2 – *Le cycle de contrôle dans RT1 selon [Dodhiawala et al., 1989]*

5.2.3.3 Conclusion

Les atouts de l'approche RT1 pour le temps réel se situent au niveau de ses capacités en terme de réactivité aux événements et d'adaptation de cette réactivité. La réactivité aux événements est faite en introduisant des canaux pour permettre de mener en parallèle différents cycles de contrôle en fonction de l'importance des événements traités. L'adaptation de cette réactivité se fait en paramétrant le nombre de SCs à exécuter par cycle.

On retrouve cependant la difficulté qui apparaît dans BB1 temps réel. En effet, la détermination des paramètres du système rend difficile la conception d'une application RT1. De plus, et à la différence de BB1 temps réel, le cycle de contrôle n'est pas borné en temps.

5.2.4 DVMT temps réel

La version temps réel de DVMT est une extension de DVMT (Distributed Vehicle Monitoring Testbed) [Lesser et al., 1988] et reprend donc l'architecture blackboard de ce dernier (Figure 5.3). On y retrouve les deux blackboards (blackboard de buts et de données), le scheduler ainsi que les trois tables permettant de choisir la SC à exécuter en fonction des événements : une table d'événements à partir de laquelle sont créés des buts, une table de buts à partir de laquelle sont identifiés des sous-buts et enfin une table des sous-buts à partir de laquelle sont activées des SCs.

L'extension au temps réel est faite ici en introduisant des mécanismes permettant de mener des raisonnements approximatifs à l'aide de plans et de prendre en compte des échéances.

Pour cela, le système se base sur une estimation des temps d'exécution des plans pour déterminer les niveaux d'approximation permettant de respecter les échéances fixées.

5.2.4.1 Le raisonnement approximatif dans DVMT

Le raisonnement approximatif dans DVMT vise à adapter de façon optimale la qualité des solutions au temps disponible pour les générer. Ainsi lorsque ce dernier est insuffisant pour obtenir une réponse optimale à un problème donné, ce mécanisme permet d'obtenir une solution dégradée. L'approximation peut se faire dans trois directions :

- la complétude : le système ne va s'intéresser qu'à certains aspects de la solution, et qui sera ainsi incomplète ;
- la précision : la valeur de certains paramètres de la solution sera plus ou moins précise ;

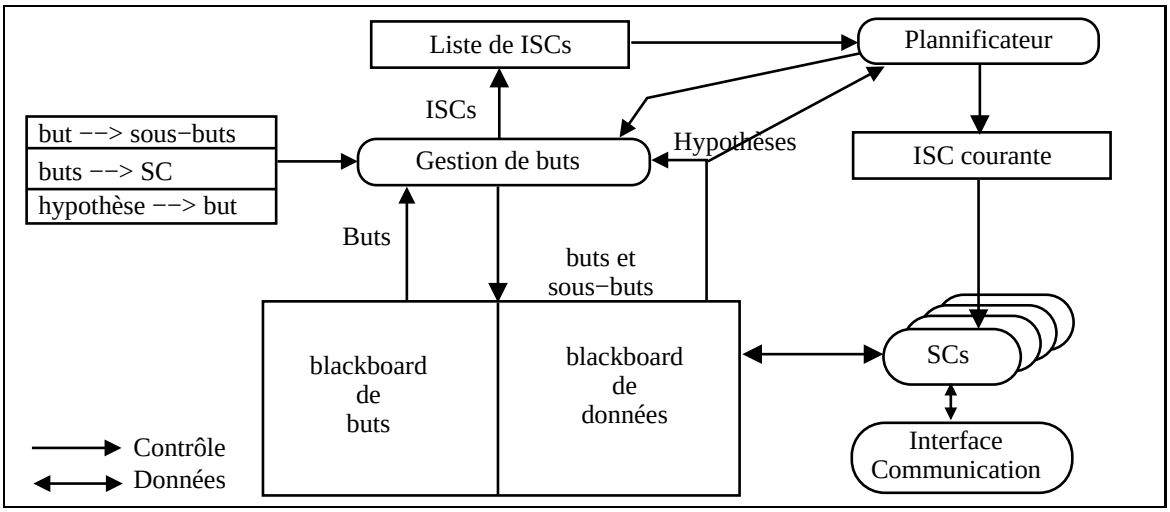


FIG. 5.3 – *Architecture de DVMT*

– la certitude : les traitements visant à vérifier que la solution est correcte ou non sont plus ou moins faits.

Lors du développement d'une application, l'utilisateur doit définir et implémenter des SCs produisant chacune une solution de qualité différente à un même problème mais nécessitant un temps de calcul également différent. La qualité de la solution s'accroît lorsque les temps de calcul sont croissants. Ce sont ces SCs qui définiront les niveaux d'approximation possibles pour résoudre un problème.

5.2.4.2 Fonctionnement

Comme dans la première version de DVMT, DVMT temps réel utilise un planificateur pour raisonner sur des buts lointains (Figure 5.4). Celui-ci a pour but de construire un plan abstrait incluant les principales activités à mener pour atteindre les buts courants. Seules les étapes proches dans le temps sont détaillées, l'affinement des suivantes se faisant de manière incrémentale après que les premières ont été exécutées. C'est le plannificateur qui détermine le niveau d'approximation nécessaire pour atteindre les buts courants avant leurs échéances respectives. Il choisit pour cela parmi les plans élaborés par l'utilisateur.

Parmi ces plans, certains permettent d'atteindre des buts identiques avec des niveaux d'approximation différents. Ce choix est basé sur des mesures statistiques de temps d'exécution et de qualité des solutions de ces plans ou de plans similaires exécutés dans le passé. Ces mesures statistiques ainsi que des modèles de résolution de problèmes permettent de faire des estimations sur les temps de résolution en fonction des niveaux d'approximation et de choisir ainsi le bon niveau pour les échéances données.

Le planificateur met en œuvre trois stratégies d'approximation au niveau du black-

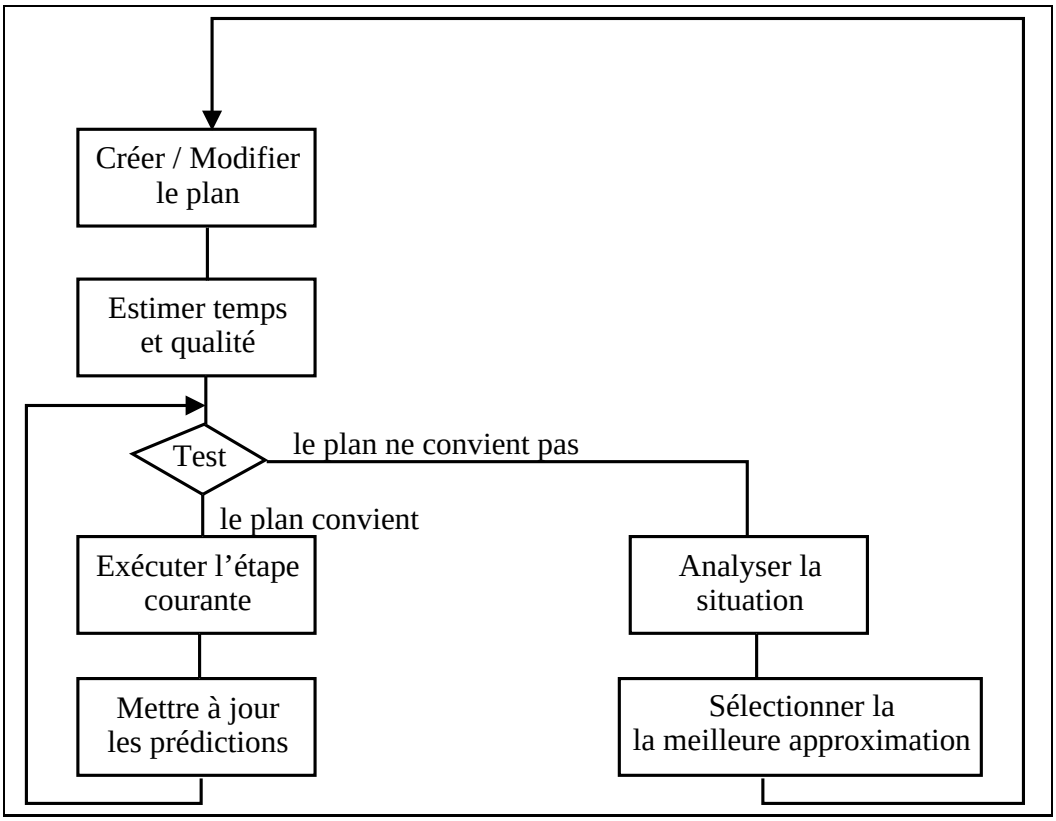


FIG. 5.4 – *La boucle de planification dans DVMT*

board pour arriver à un niveau de qualité de la solution compatible avec le temps disponible pour la résolution :

- une approximation sur les données : certains champs des données ne seront pas remplis ou certaines données seront regroupées ;
- une approximation sur l'espace de recherche : des données provenant de l'extérieur ne seront pas prises en compte et/ou les interprétations compétitives des SCs seront restreintes ;
- une approximation sur les connaissances : certaines SCs pourront être omises au cours de la résolution.

En pratique, l'utilisateur donne les objectifs, les échéances ainsi que l'ordre dans lequel les approximations doivent être employées.

5.2.4.3 Conclusion

Ici, à la différence des raisonnements progressifs ou anytime le degré de qualité de la solution est déterminé avant le raisonnement correspondant. L'avantage du raisonnement approximatif tel qu'il est implémenté dans DVMT temps réel est que le temps est utilisé de façon optimale pour la résolution du problème puisqu'à la fin, une seule solution d'un

niveau de qualité fixé à l'avance aura été générée pour un problème donné.

Cependant, dans des applications complexes, il apparaît que le processus d'identification du niveau d'approximation nécessaire peut être lui aussi très complexe et coûteux en temps. De même, l'évaluation correcte du temps d'exécution des plans peut s'avérer très difficile. On peut facilement imaginer que dans certains cas, l'évaluation soit plus complexe que la résolution proprement dite.

De plus, ce raisonnement n'est pas adapté à des situations dans lesquelles les échéances peuvent varier au cours du raisonnement, ce qui est souvent le cas dans beaucoup d'applications temps réel.

D'autres évolutions de DVMT ont été menées pour exécuter les SCs en parallèle et sur des processeurs différents [Decker et al., 1992a] [Decker et al., 1992b].

5.2.5 ATOME-TR

Le système ATOME-TR [Brunessaux et al., 1992] [Lementec and Brunessaux, 1992] [Haton et al., 1991] est un environnement de prototypage de systèmes multi-experts orientés temps réel. Réalisé dans le cadre de travaux de recherche menés en coopération entre MatraDéfense, Matra Cap Systèmes et le Centre de Recherche en Informatique de Nancy, ATOME-TR est le résultat de deux études supportées par la DGA/DRET et le MRT, ayant pour objet l'extension de l'architecture blackboard ATOME [Laasri and Laasri, 1988] [Haton et al., 1991] pour lui permettre de fonctionner en temps réel. ATOME-TR a été utilisé par MatraDéfense dans le cadre d'un maquettage de SEGCM (Système Expert de Gestion de Contre-Mesures) [Desard et al., 1991].

5.2.5.1 Architecture générale

ATOME-TR repose sur une architecture blackboard multi-processus. Dans ce modèle, trois types de sources de connaissance interagissent (Figure 5.5) :

- la stratégie, source de connaissance de contrôle, responsable de la coordination générale du système ;
- les tâches, sources de connaissance de contrôle local, responsables de la coordination de spécialistes en vue de la résolution d'un sous-problème ;
- les spécialistes, sources de connaissance du domaine, chargées de la résolution d'un aspect du problème.

Un module particulier, appelé l'intégrateur, est chargé de l'acquisition et du filtrage intelligent des données en provenance de l'environnement à contrôler. Des structures de contrôle (résumé de blackboards, buts, sous-buts, événements) sont utilisées pour coordonner les sources de connaissance et adapter leur raisonnement.

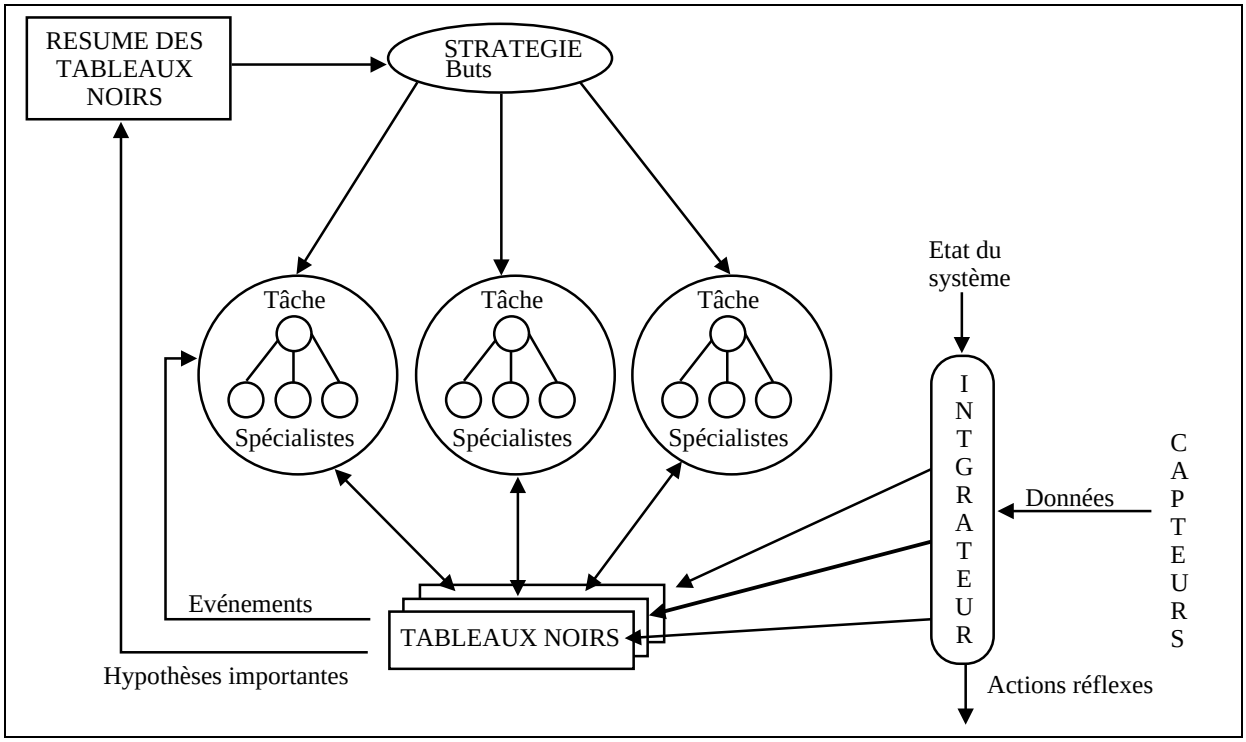


FIG. 5.5 – *Architecture d'Atome-TR d'après [Lementec and Brunessaux, 1992]*

5.2.5.2 Fonctionnement en temps réel

Les caractéristiques d'ATOME-TR pour un fonctionnement en temps réel sont les suivantes :

- Un mécanisme de planification réactive [Firby, 1987][Schoppers, 1987], approche reprise dans PRS [Ingrand and Coutance, 1993] permettant de faire du raisonnement adaptatif. Le raisonnement est représenté sous forme de plans, un plan devant correspondre à la réalisation d'un but et chaque nœud de ce plan correspondant à un sous but. Chaque parcours possible du graphe correspond à une manière d'atteindre le but fixé, le choix du parcours se faisant en temps réel en fonction de la situation et du temps disponible. Ceci permet donc d'adapter en temps réel la manière d'atteindre ce but et donc le raisonnement proprement dit. ATOME-TR permet deux niveaux de planification : un niveau stratégie et un niveau tâche ; un but au niveau stratégie se décomposant en sous-plans au niveau tâche (Figure 5.6).
- Sa capacité à exécuter en parallèle des sources de connaissance de contrôle et de domaine pour assurer la réactivité du système, ce qui revient à pouvoir parcourir plusieurs branches d'un plan en parallèle.
- Des mécanismes d'interruption spécifiques dépendant du domaine d'application.
- Une utilisation combinée des raisonnements dirigés par les buts et par les événements permettant la mise en œuvre de raisonnement approximatif ou progressif. Le

raisonnement dirigé par les buts se fait par la génération de buts de plus haut niveau par la stratégie, ceci en fonction de l'état d'avancement du raisonnement représenté dans le résumé des tableaux noirs. Ces buts sont ensuite décomposés en sous-buts au niveau tâche. Le raisonnement dirigé par les événements se fait lui au niveau des tableaux noirs qui donnent la possibilité d'envoyer des événements à des tâches spécifiques lors de l'arrivée ou de la modification de certains types d'hypothèses.

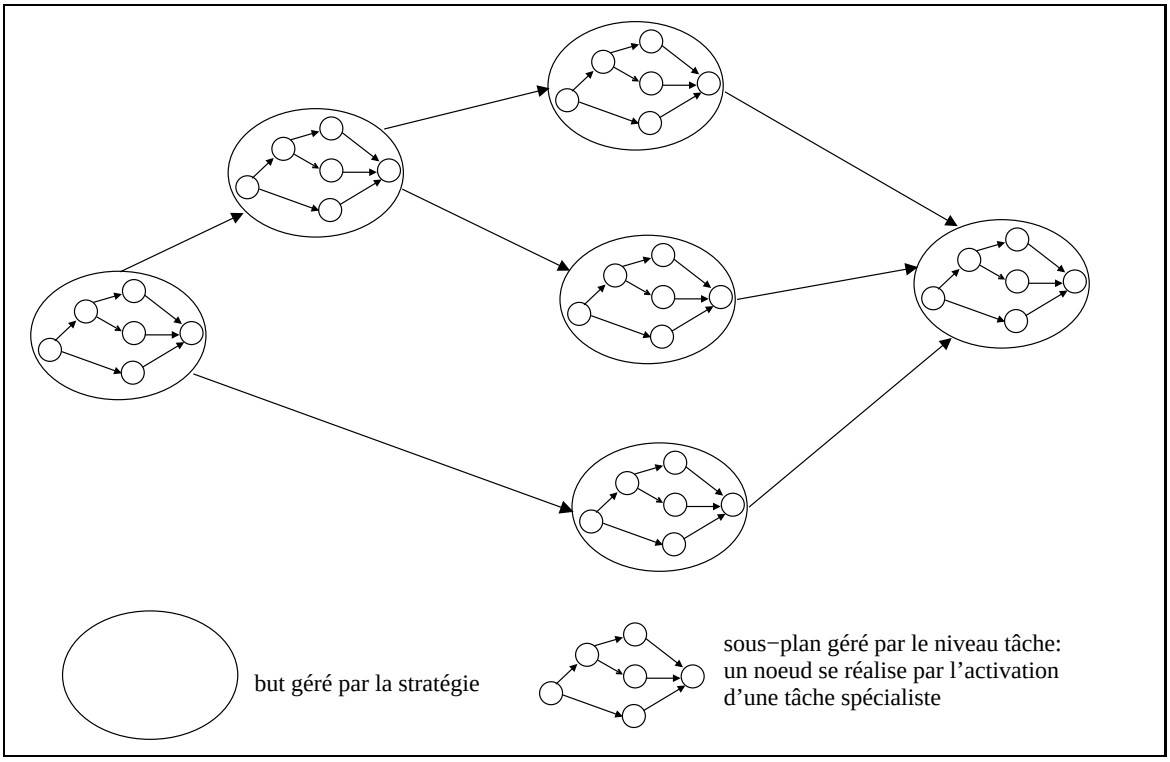


FIG. 5.6 – *Planification réactive dans Atome-TR d'après [Brunessaux et al., 1992]*

5.2.5.3 Conclusion

Les principaux atouts de cette architecture pour le temps réel se situent principalement au niveau de sa réactivité (module intégrateur, raisonnement dirigé par les buts, mécanismes d'interruption des spécialistes) et de ses capacités d'adaptation du raisonnement par l'intégration au niveau de la stratégie et des tâches d'un mécanisme de planification réactive.

Cependant, la mise en œuvre efficace de ces mécanismes peut s'avérer délicate du fait de leur relative complexité. En particulier, la décomposition et l'implémentation de l'expertise en SCs et plans au niveau de la stratégie et des tâches n'est pas triviale. De plus, même si l'architecture permet de les accueillir, elle n'intègre pas de mécanismes

pour raisonner en temps contraint tels que des techniques de raisonnement approximatif ou progressif.

5.2.6 REAKT

REAKT [AFIA, 1994] [Lalanda et al., 1992] [Mouaddib, 1993] est un outil de développement de systèmes multibases de connaissances orienté temps réel et basé sur le modèle du tableau noir.

5.2.6.1 Architecture générale

Le système est constitué des éléments suivants (Figure 5.7) :

- des sources de connaissances qui peuvent être de natures différentes (procédurales, ordre 0+ ou ordre 1 compilées avec RETE) et qui sont toutes interruptibles ;
- un module de contrôle qui a pour rôle essentiel de coordonner les SCs et d'assurer entre elles une bonne coopération pour conduire le raisonnement. Ce module intègre des capacités de planification tout en restant réactif aux modifications de l'environnement ;
- un *timer* qui gère une horloge interne avec des fonctions d'accès mais qui fournit également des fonctions pour gérer des événements et actions dans le temps ;
- un gestionnaire de communications intelligent (ICM : « Intelligent Communication Manager ») chargé d'intégrer de façon intelligente les données provenant de l'extérieur en les filtrant et en ne présentant que celles utiles au système à l'instant considéré ;
- le tableau noir (*blackboard*) qui a la particularité de pouvoir contenir des slots (nœuds) multi-valués. Un slot pourra avoir à un instant donné des valeurs passées, une valeur courante et des valeurs prédites. La manipulation de ces valeurs se fait au niveau des SCs. Dans le cas d'une SC à base de règles, une affectation en partie droite de règle sera prédite si toutes les conditions en partie gauche le sont ou si au moins une l'est et les autres sont courantes. De même, une affectation sera courante si toutes les prémisses en partie gauche le sont. Une valeur sera passée dès qu'une autre valeur deviendra courante et la remplacera. Une valeur affectée à un slot pourra être au cours du temps prédite puis courante puis passée. On peut également affecter à une valeur courante ou prédite un intervalle de temps de validité. On peut dire par exemple que « A=3 » est prédit sur l'intervalle de temps [maintenant+3sec, maintenant+20sec]. À la date « maintenant+20 sec », si « A=3 » n'est toujours pas devenue courante, elle est retirée et n'est même plus prédite. REAKT manipule un graphe temporel pour gérer les dépendances temporelles entre faits prédits et courants et leurs intervalles temporels de validité.

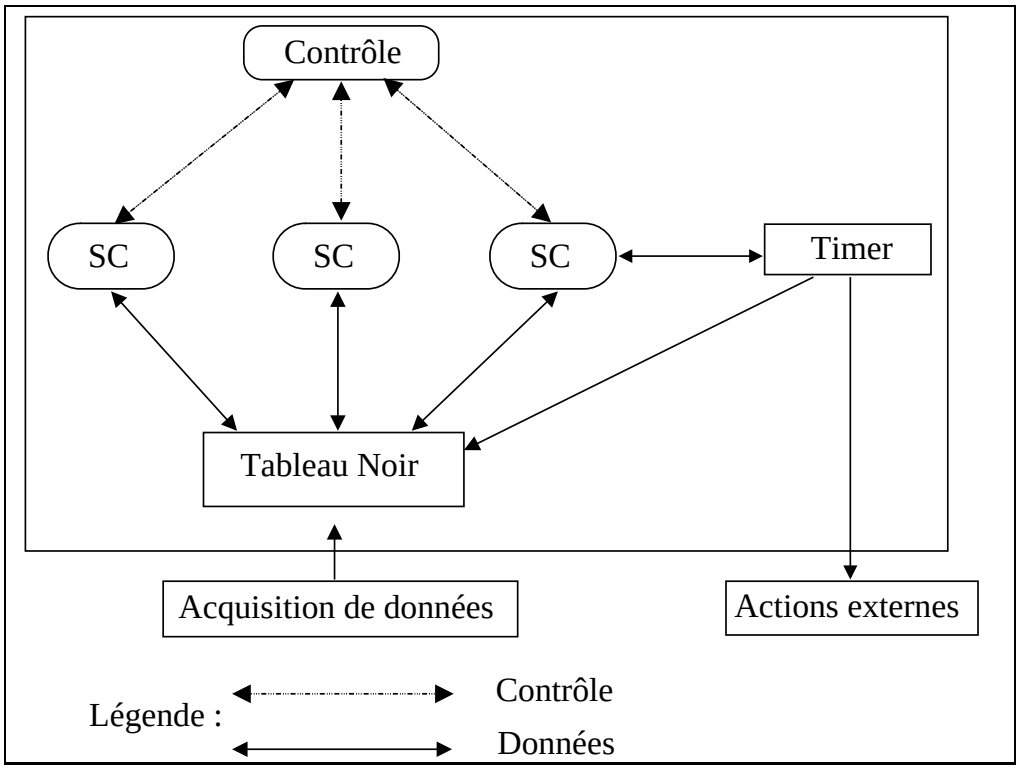


FIG. 5.7 – *REAKT : Architecture générale selon [Lalanda et al., 1992]*

5.2.6.2 La conduite générale du raisonnement

D'une façon générale, une forme de contrôle adapté au temps réel doit permettre au système d'être sélectif, réactif, synchronisé sur l'extérieur, adaptatif, respectueux des échéances et efficace. L'architecture générale du contrôle est la suivante :

- une SC de contrôle : génère les intentions après évaluation de la situation courante ;
- une bibliothèque de plans définis statistiquement : un plan par intention. Un plan est constitué d'enchaînement de SCs avec des conditions d'exécution. Plusieurs chemins sont possibles en fonction de la situation. Une étape d'un plan peut avoir ou pas de date limite ; celle-ci peut être fixée par l'utilisateur à la conception ou être déduite par le système à partir de dates limites d'étapes ultérieures dans le plan ;
- un gestionnaire d'intentions (intention = but) : il gère l'exécution concurrente de plusieurs intentions en assurant le respect des échéances. Il gère également le parcours des plans associés aux intentions en tenant compte de la situation courante. Un algorithme d'ordonnancement est utilisé lorsque l'exécution en parallèle des intentions courantes ne permet pas de respecter certaines échéances. Si toutes les intentions en cours sont sans échéances, elles sont exécutées en parallèle. Si une intention avec échéance arrive, les intentions sans échéances sont interrompues et celle-ci est exécutée seule. Si une autre intention avec échéance arrive, les deux intentions sont

alors exécutées en parallèle si leurs dates limites peuvent être respectées, sinon, le gestionnaire d'intention les ordonne en fonction des priorités ;

- un ensemble d'agents : chacun est chargé de l'exécution d'une SC. Un agent peut être actif (il exécute sa SC), en attente (il lui manque des données pour continuer) ou être suspendu par le gestionnaire d'intentions (si l'arrivée d'une intention le nécessite pour assurer le respect des échéances).

5.2.6.3 Conclusion

Les atouts de REAKT se situent essentiellement au niveau de la réactivité du système et des possibilités de raisonnement temporels. La réactivité est liée à la présence du *timer* et à la possibilité d'implémenter des Sources de Connaissances interruptibles. Le formalisme de représentation des données (slots multi-valués) permet à ces mêmes SCs de mener des raisonnements temporels. Le raisonnement temporel peut également être mis en œuvre au niveau du langage de plans.

Les mécanismes d'ordonnancement permettent dans une certaine mesure de tenir compte de contraintes de temps. Cependant, ces mécanismes se limitent à l'ordonnancement de tâches dont il faut connaître a priori le temps d'exécution. De plus, lorsqu'il n'est pas possible d'ordonner toutes les tâches avec date limite, le système est obligé de faire un choix parmi celles-ci. Enfin, le raisonnement temporel à l'aide du formalisme de représentation des données est délicat à maîtriser (chaque donnée ayant une valeur passée, courante et prédite).

5.2.7 RT-SOS/GREAT

Le système RT-SOS (Real-Time Society Of Specialists) [Mouaddib et al., 1992], [Mouaddib et al., 1993] reprend une approche blackboard un peu particulière appelée *société de spécialistes* que l'on peut considérer comme une approche multi-blackboards. RT-SOS l'adapte pour prendre en compte des contraintes de temps au cours du raisonnement. L'adaptation est faite par la mise en œuvre d'un fonctionnement asynchrone, l'introduction du parallélisme simulé et surtout par la mise en œuvre au niveau des spécialistes de raisonnement progressif basé sur le modèle GREAT (Guaranteed REAsoning Time) [Mouaddib, 1993].

5.2.7.1 Architecture générale

L'architecture reprend le principe de structures hiérarchiques du modèle de société de spécialistes. Une société de spécialistes est composée d'une administration, de différentes associations et d'un intégrateur.

L'administration (1) détermine les buts à traiter du système et la charge des plans associés (Figure 5.8), (2) fait le suivi de l'exécution de chacun de ces plans et (3) donne à traiter aux associations (Figure 5.9) les buts qui y sont associés. Elle est constituée de trois entités :

- un scheduler chargé de gérer et d'ordonner les messages de la boîte à lettres globale. Cette boîte à lettres contient des messages soit internes (au système) et provenant des associations, soit externes provenant des capteurs. Les tâches principales du scheduler sont alors d'identifier la nature des messages (internes ou externes), de leur attacher une priorité et de les transmettre soit au planificateur (messages internes) soit au gestionnaire d'interruptions (manager d'IT) (messages externes) ;
- un planificateur chargé de mettre à jour les plans actifs, de désigner les buts à traiter et les associations qui les traiteront ;
- un gestionnaire d'interruptions chargé en fonction des messages externes de déterminer soit les nouveaux plans à activer et à livrer au planificateur, soit les plans à interrompre.

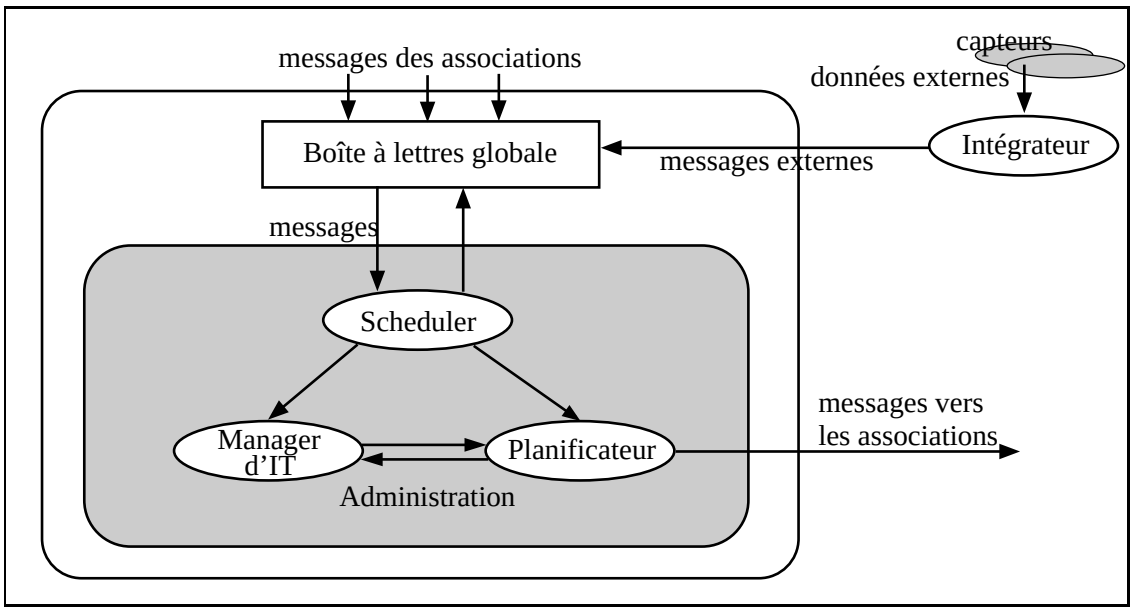


FIG. 5.8 – *L'administration dans RT-SOS selon [Mouaddib, 1993]*

L'intégrateur est chargé de faire le lien entre l'environnement extérieur et le système. Il génère pour cela des messages vers l'administration dès l'arrivée de données extérieures. Chaque association est constituée de différents modules (Figure 5.9) :

- un directeur d'association chargé de recevoir les buts donnés par l'administration, de les décomposer en sous-buts, de donner les sous-buts qui sont dans les compétences de l'association aux spécialistes concernés et de renvoyer ceux qui ne le sont pas à l'administration ;

- un gestionnaire de messages qui scrute la conférence en attente de modifications importantes et envoie des messages pour rendre compte au directeur dès que cela se produit ;
- des spécialistes qui exécutent les buts donnés par le directeur de leur association respective ; ces buts sont transmis par messages que chaque spécialiste est capable d'interpréter pour y associer une méthode à exécuter.

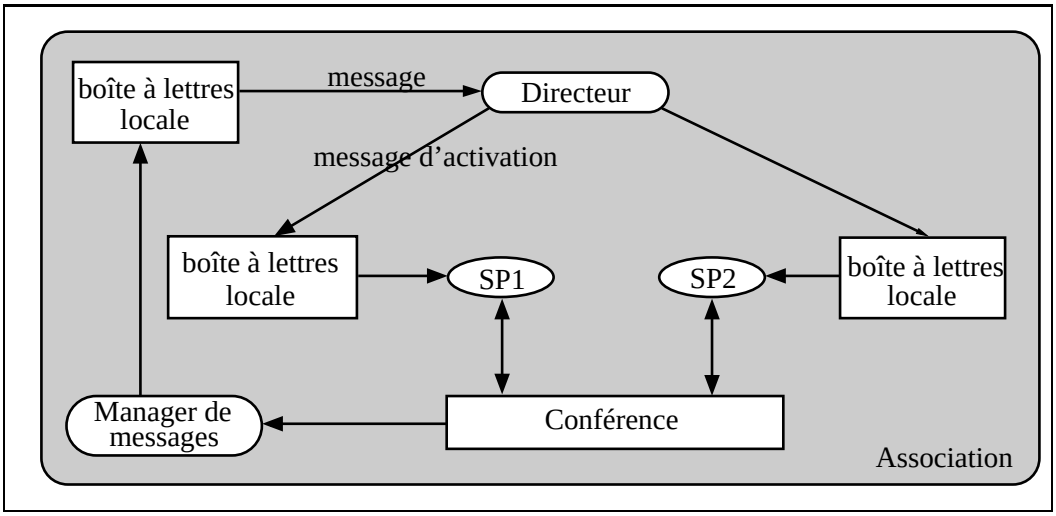


FIG. 5.9 – *Structure d'une association selon [Mouaddib, 1993]*

Les communications se font via des structures de données et des messages: au sein d'une même association les spécialistes communiquent entre eux via une structure de données communes appelée conférence (une conférence joue en fait le rôle d'un blackboard local à une association) et reçoivent des message de leur directeur d'association. Le directeur reçoit des messages indirectement des spécialistes, messages qui sont générés par la conférence lorsqu'un ou plusieurs spécialistes y ont fait des modifications significatives. Les associations communiquent par envoi de messages. Ce sont leurs directeurs respectifs qui s'occupent de cette tâche. L'administration ainsi que chaque directeur et spécialiste dans une association disposent d'une boîte à lettres propre.

5.2.7.2 La réactivité dans RT-SOS

Elle est assurée à différents niveaux:

- au niveau du scheduler par l'attribution d'une priorité à chaque message de la boîte à lettre globale. Cette priorité est fonction d'une part d'une date limite attachée au message et d'autre part, d'une importance calculée elle aussi par le scheduler. La date limite indique que les données contenues dans ce message ont une certaine durée de validité. L'importance quant à elle est une fonction (1) de la charge du receveur, qui est directement liée au nombre de messages que sa boîte à lettre contient, (2) du

nombre de buts à long terme dont dépend ce message, (une heuristique discriminatoire dépendante de l'application) et enfin (3) de sa complexité, liée au nombre de sous-buts qui peuvent être identifiés après décomposition du but transporté par le message. Le premier élément permet d'éviter d'engorger des associations, le second favorise les messages qui peuvent apporter des informations à des buts en attente, le troisième favorise les buts simples à résoudre ;

- au niveau du manager d'IT qui peut interrompre des raisonnements ;
- au niveau des associations par l'utilisation d'une conférence qui, surveillée par un gestionnaire de messages, permet au directeur d'être réactif aux évolutions importantes du raisonnement au sein de son association.

5.2.7.3 Introduction du raisonnement progressif dans GREAT

Le raisonnement progressif dans GREAT est fait au niveau des spécialistes qui, lorsqu'ils reçoivent du directeur dont ils dépendent un but à atteindre, peuvent fonctionner soit en mode normal si le temps imparti est suffisant soit en mode progressif dans le cas contraire. Ce choix est fait par le directeur de l'association qui est capable de juger si ce temps est suffisant en fonction de l'échéance associée au but et de la tâche correspondante à réaliser.

Lors du passage en mode progressif, les données locales ainsi que les méthodes du spécialiste concerné reçoivent un niveau de granularité. Elles sont alors ordonnées en régions pour les données et en paquets de règles pour les méthodes, chacune est associée à un de ces niveaux selon le modèle GREAT [Mouaddib, 1993]. Ces différents niveaux correspondent alors à différents niveaux de raisonnement de qualité croissante. La solution au premier niveau sera d'abord générée, puis la deuxième et ainsi de suite jusqu'à ce que le temps imparti soit écoulé.

Pour juger du mode à utiliser, le directeur dispose d'une heuristique qui lui permet de prendre sa décision à partir de trois éléments : d'une estimation du temps de raisonnement minimum (niveau 1 en mode progressif) du spécialiste considéré pour atteindre le but, du temps maximum et de la complexité du but à atteindre, c'est-à-dire de sa "décomposabilité" en sous-buts.

5.2.7.4 Conclusion

Les capacités de RT-SOS pour le temps réel se situent au niveau des communications (interruptions, focalisation) pour assurer une certaine réactivité au système et au niveau des spécialistes qui peuvent mettre en œuvre des raisonnements progressifs avec GREAT. C'est essentiellement ce dernier aspect qui fait l'originalité de l'approche.

On voit cependant rapidement la difficulté qu'il y a à définir un contrôle efficace pour

mettre en œuvre ce type de mécanismes dans un raisonnement distribué, en particulier au niveau des heuristiques pour définir les priorités et choisir le passage en mode progressif des spécialistes.

5.2.8 L'approche temps réel d'Occello

Cette architecture d'agent temps réel [Occello and Demazeau, 1994] est également basée sur la notion de tableaux noirs. Elle est constituée d'un contrôleur de processus impliquant la plannification et la décision. L'architecture comporte des capacités cognitives et réactives. Chaque agent peut à la fois percevoir et communiquer pendant qu'il décide de l'action à réaliser.

Dans ce modèle, les auteurs mettent en avant le fait que le concepteur doit à la fois prendre en compte l'aspect temps réel au niveau mascroscopique et au niveau microscopique. Le niveau macroscopique définit les capacités de l'agent. Le niveau microscopique (l'implémentation de l'agent) concerne le niveau du système d'exploitation, de l'architecture matérielle et logicielle.

5.3 Comparaison et récapitulatif des modèles présentés

Nous dressons dans ce paragraphe un tableau comparatif des différents modèles que nous avons présentés tout au long de ce chapitre sur les systèmes multi-agents temps réel. Nos éléments de comparaison seront les suivants :

- le niveau de traitement : agent ou SMA ;
- le type de raisonnement employé : inférence, approximatif, progressif, ... ;
- s'agit-il de l'extension d'un système existant non temps réel ;
- la présence de raisonnement progressif ;
- la structure des agents : léger ou non ;
- les atouts du système ;
- les faiblesses du système.

L'analyse du tableau 5.1 met en évidence les points faibles qui reviennent régulièrement dans la conception de systèmes multi-agents temps réel. L'un de ceux qui nous apparaît comme étant important concerne l'absence de prise en compte du temps réel au niveau SMA. En effet, les traitements temps réel sont quasiment tout le temps fait au niveau de l'agent. De plus, ces approches, surtout en ce qui concerne les approches à base de tableau noir, sont très lourdes en terme de fonctionnement, en termes d'utilisation et de mise en œuvre.

5.4 Limites des travaux existants

L'approche à base de tableaux noirs est sûrement la plus répandue pour concevoir des architectures multi-agents temps réel. Il y a une raison « historique » car comme on l'a vu, la plupart des systèmes à base de tableaux noir temps réel sont des extensions d'architectures développées précédemment auxquelles sont rajoutés des mécanismes spécifiques. Mais on peut trouver des raisons propres à ce type d'architecture par rapport aux autres approches multi-agents. Cette approche permet en effet d'implémenter relativement facilement des raisonnements opportunistes et des mécanismes de focalisation. On remarque que les mécanismes introduits ont, dans la plupart des cas, visé à améliorer la réactivité de ces systèmes à l'évolution de leur environnement et à pouvoir leur permettre d'adapter leur raisonnement. A part RT-SOS et REAKT dans une certaine mesure, aucun d'eux n'introduit de mécanismes complets et vraiment efficaces permettant de raisonner en temps contraint. De plus, ces architectures sont en général très complexes et leurs mécanismes très difficiles à mettre en œuvre, ce qui les rend difficilement prédictibles.

Les architectures d'agents temps réel utilisant l'approche à base de tableaux noirs sont par trop restrictives en comparaison des possibilités des architectures d'agents telles qu'elles sont décrites dans [Huget and Pinson, 1999] [Wooldridge and Jennings, 1994] [Ferber, 1995].

Nous avons fait dans ce chapitre un tour d'horizon des différents travaux effectués dans le domaine des systèmes multi-agents temps réel. De cet examen, il ressort que les approches utilisées jusqu'à maintenant ne sont pas encore réellement utilisables. Elles comportent de nombreuses lacunes dont l'absence de prise en compte de l'aspect temps réel au niveau organisation du SMA pour un grand nombre d'entre-elles. De plus, l'approche utilisant les tableaux noirs est très lourde et ne s'applique pas à des architectures utilisant des agents légers.

5.4.1 Une restriction au niveau de l'agent

Dans la plupart des SMA temps réel, on ne tient compte de l'aspect temps réel que lors de la conception des agents. Il apparaît même que certains estiment que cela est suffisant pour que le SMA soit temps réel ou anytime. Or, il s'avère que la complexité des interactions entre agents rend en réalité très difficile la prédiction les temps d'exécution pour une tâche. En effet, cela dépend du nombre d'agents qui vont intervenir pour le traitement de cette tâche. Cette partie n'est pas déterministe dans un SMA.

5.4.2 La difficulté du changement de niveau

Concevoir des agents temps réel ne constitue pas en soi une difficulté plus importante que celle de la conception d'applications temps réel classiques. On peut simplifier le processus en encapsulant une application temps réel ou, pour l'approche anytime, un algorithme anytime dans un agent. Le plus difficile lors de la conception de SMA temps réel ou anytime consiste à passer du niveau agent au niveau SMA. On peut se demander que faut-il faire pour qu'un SMA devienne temps réel connaissant la complexité naturelle des SMA (cf. chapitre 3). Nous tenterons d'apporter des réponses et quelques solutions à ce problème dans la seconde partie de ce mémoire.

L'ensemble des travaux n'a pas beaucoup progressé ces dernières années. Quelques travaux ont été présentés à la communauté des chercheurs mais il s'agit essentiellement d'extensions mineures des travaux existants. Ceci démontre la difficulté d'introduire des extensions temps réel au niveau des systèmes multi-agents. Notre objectif est d'apporter une solution pour la conception d'applications reposant sur le paradigme multi-agents et nécessitant à la fois de prendre en compte la notion de temps réel et la notion de distribution physique.

Modèle	Niveau de traitement	Raisonnement	Extension à un système existant	Anytime/ Tableau noir	Agent léger	Atouts	Faiblesses
CAAM	agent	progressif	non	anytime	oui	un vrai modèle agent	pas de niveau SMA
BB1 temps réel	agent	inférence à partir des SCs	oui	tableau noir	non	adaptabilité à l'environnement	difficulté pour l'application à des problèmes complexes
RT1	agent	inférence à partir des SCs	non	tableau noir	non	réactivité adaptation de cele-ci	difficulté pour déterminer les paramètres
DVMT-TR	agent	approximatif	oui	tableau noir	non	temps utilisé de façon optimale	raisonnement inadapté au changement d'échéances au cours du raisonnement
ATOME-TR	agent	plannification et adaptatif	oui	tableau noir	non	réactivité et adaptabilité du raisonnement	complexité de la mise en œuvre
REAKT	agent	inférence	non	tableau noir	non	réactivité et raisonnement temporel	limité à l'ordonnancement de tâches connaître a priori les temps d'exécution
RT-SOS	agent	progressif	oui	tableau noir	non	réactivité	difficultés pour mettre en œuvre ces mécanismes en mode distribué

TAB. 5.1 – *Comparaison de différents SMA temps réel*

Chapitre 6

ANYMAS : un modèle pour la réalisation de SMA anytime

Résumé

La conception de systèmes multi-agents, basée sur l'utilisation des techniques anytime, est une nouvelle approche pour la prise en compte de l'aspect temps réel dans un SMA. Elle signifie que le comportement du système devra respecter les principes de ces techniques : être en mesure de fournir un résultat dont la qualité est fonction du temps imparti au système. La mise en œuvre d'algorithmes anytime au niveau agent consiste à modéliser le comportement des agents suivant les techniques anytime. Nous présentons dans ce chapitre la structure de notre agent anytime. Cependant, dans un système multi-agent anytime, la présence d'agents anytime n'est pas suffisante pour que le comportement du SMA soit anytime. Il est en effet nécessaire de contraindre le comportement du SMA d'une façon plus globale. C'est pour cela que nous considérons deux niveaux dans notre modèle : le niveau agent (dit niveau local) et le niveau SMA (dit niveau global). Nous allons présenter dans ce chapitre le modèle ANYMAS, un modèle de système multi-agent anytime.

6.1 Introduction

Par nature, un SMA est composé de multiples entités coopérantes appelées agents. Les agents d'un SMA interagissent et coopèrent à la résolution de problèmes complexes. Le schéma d'interaction pour la résolution d'un problème particulier est quasiment imprévisible : on ne peut pas savoir combien d'agents vont intervenir, quels vont être ces agents, combien de fois ils vont communiquer et s'échanger des messages. Par conséquent, il apparaît très difficile de prévoir le temps d'exécution nécessaire à la résolution d'un problème particulier. Nous allons présenter dans ce chapitre une nouvelle approche pour la prise en compte du temps réel dans un SMA : le modèle ANYMAS, un modèle de système

multi-agent basé sur l'approche anytime.

Dans un système multi-agent anytime, il va être 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'algorithme. A cette fin nous allons présenter un modèle d'agent anytime sur lequel les SMA devront 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. Ce niveau repose très fortement sur le niveau précédent. Nous allons introduire des agents qui seront chargés de coordonner les agents anytime suivant des groupes fortements 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.

Notre objectif est d'enrichir les SMA classiques suffisamment pour qu'ils puissent fournir des résultats dans des temps « contrôlés ». Nous allons à cet effet nous baser sur les techniques anytime qui nous permettent d'obtenir des résultats partiels à des pas d'exécution intermédiaires. Notre modèle considérera comme acquises les propriétés classiques d'un agent et d'un SMA [Ferber, 1995] [Wooldridge and Jennings, 1994]. C'est aussi pour cette raison que lors de la réalisation de ce modèle nous avons par la suite choisi d'utiliser une plate-forme de développement existante et possédant ces propriétés minimales.

6.2 Les composants du modèle ANYMAS

Le modèle ANYMAS permet la conception de systèmes multi-agents anytime et repose sur des agents anytime et des *agents de coordination temporelle* que nous allons brièvement présenter dans cette section avant d'en donner une description plus détaillée par la suite.

6.2.1 Les agents anytime du modèle ANYMAS

Pour la modélisation des agents anytime, nous avons choisi de nous inspirer des architectures modulaires comme DIMA [Guessoum and Briot, 1997] [Guessoum, 1996] (voir figure 6.1), et de nous baser sur une architecture d'agents légers. Un agent léger est un agent comportant des fonctions de communication et un comportement autonome. Nous allons à partir de cette architecture nous attacher à l'augmenter de modules nécessaires à la réalisation d'une architecture d'agents anytime. L'architecture de base de l'agent étant minimale, nous pourrons adapter notre modèle d'agent anytime sur n'importe quelle plate-forme existante en l'augmentant.

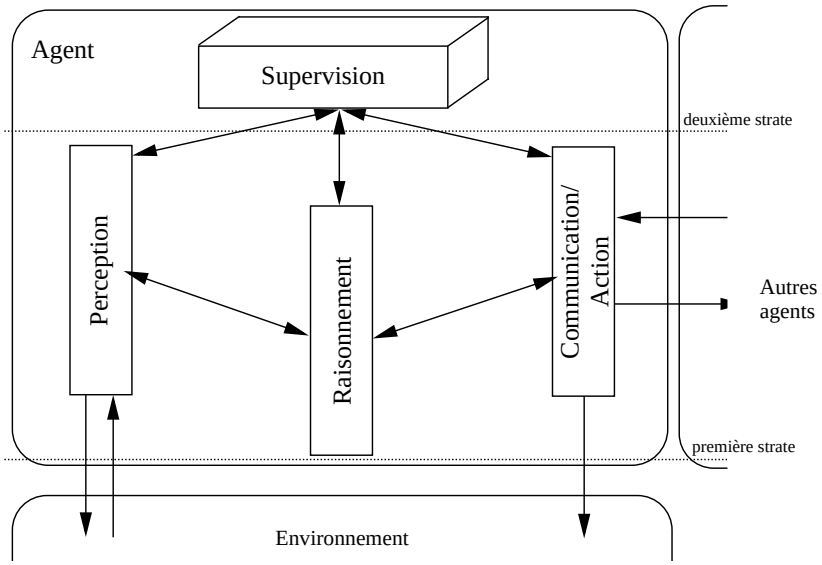


FIG. 6.1 – *Architecture d'un agent dans DIMA*

Les agents anytime devront être en mesure de :

- fournir des résultats intermédiaires de qualité croissante à chaque pas d'exécution,
- prédire le temps nécessaire pour l'obtention du prochain résultat intermédiaire.

Nous décrirons plus en détails l'architecture de cet agent dans la section 6.4.

6.2.2 Les agents de coordination temporelle

Ces agents vont intervenir au niveau global afin d'infléchir le comportement du SMA pour qu'il fournisse des résultats intermédiaires de qualité croissante en fonction des temps d'exécution alloués. Les agents de ce type vont intervenir auprès de petits groupes d'agents afin de contrôler leur comportement. Ils vont coopérer entre eux afin de savoir s'ils doivent réduire ou augmenter l'activité de leur groupe d'agents et ainsi favoriser ou contraindre l'activité d'autres groupes d'agents. Ces groupes d'agents seront formés en fonction de leurs réseaux d'accointances et de l'importance des communications à l'intérieur de ces réseaux.

Nous décrirons plus en détails l'architecture de ces agents dans la section 6.5.

6.3 Fonctionnement général du système

Nous allons introduire dans notre modèle un fonctionnement en deux temps :

1. un mode d'apprentissage : durant cette première phase, l'objectif du système sera

de s'auto-calibrer en mesurant les temps d'exécution et en mettant en place les différentes organisations d'agents nécessaires par la suite au bon fonctionnement du système. Cette phase est surtout nécessaire pour l'auto-évaluation du système.

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

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 fait appel à 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. Dans ce cadre, la qualité du résultat est indirectement une fonction de la qualité des éléments que l'on a fournis à cette fonction en entrée.

Lors d'un processus de résolution de problème ou de calcul d'une solution, on va voir intervenir des phénomènes tels que celui cité 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 donc sont soumis au phénomène de composition d'algorithmes anytime dont nous avons précédemment parlé.

La formation des groupes d'agents, qui aboutira à la naissance des agents de coordination temporelle, 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 figure 6.2 illustre la naissance des agents de coordination temporelle.

6.4 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.

Dans le modèle ANYMAS, un ATN (Augmented Transition Network) est utilisé pour stabiliser les agents dans différents états. Cette notion d'ATN est héritée de l'achitecture d'agent utilisée dans DIMA [Guessoum and Briot, 1997] (cf. figure 6.1). 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 à la fin du mode

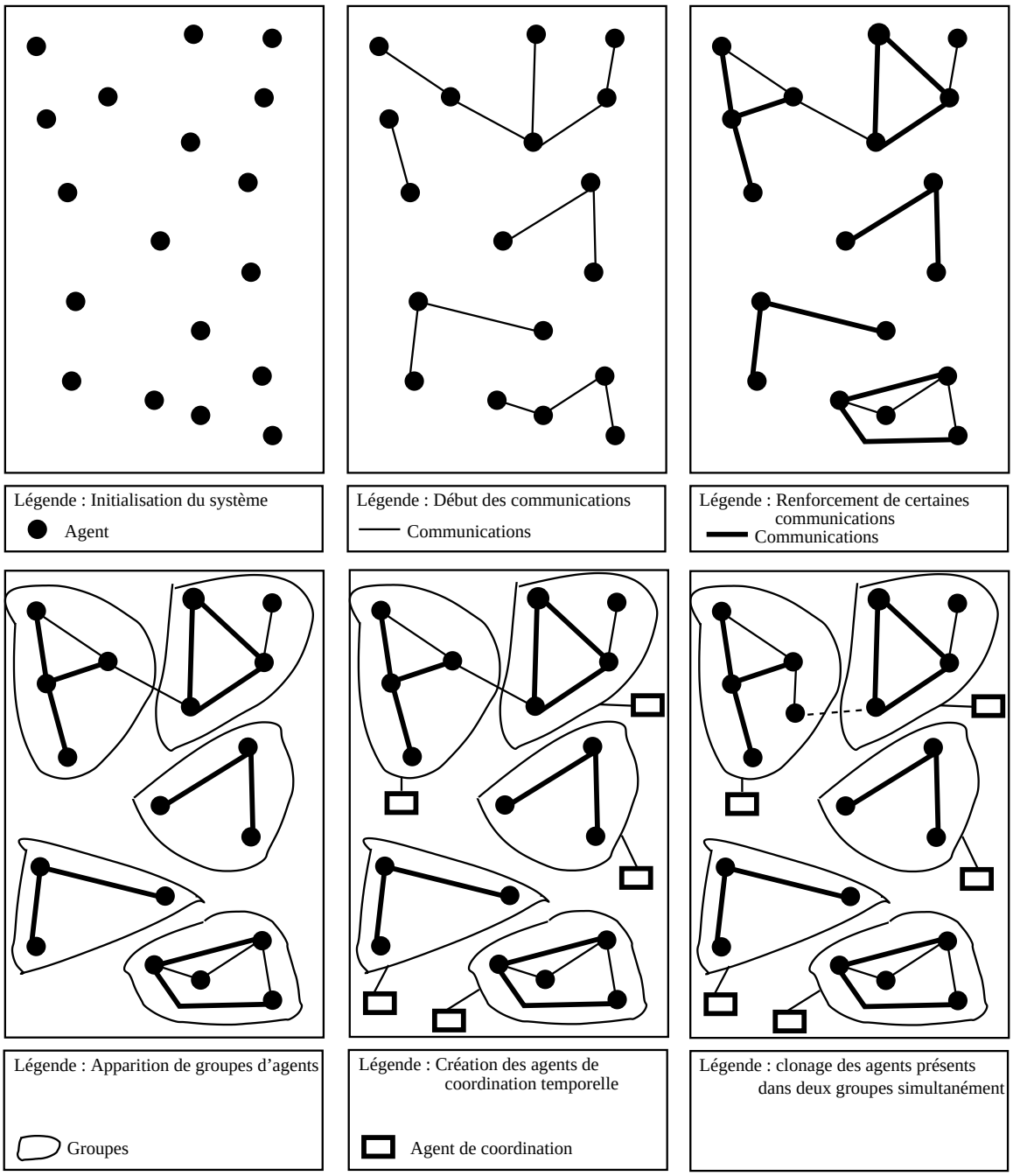


FIG. 6.2 – *Création des groupes*

d'apprentissage. Cette unité de temps est appelée ΔT dans la suite.

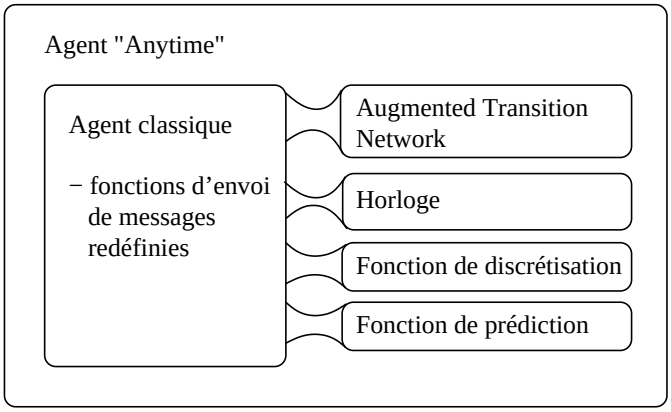


FIG. 6.3 – *Agent anytime*

Pour le fonctionnement d'un agent anytime (voir figure 6.3), nous allons donc avoir besoin des composants suivants :

- un ATN, qui va permettre de contrôler le comportement d'un agent durant sa phase active,
- une horloge temps réel, qui va permettre de mesurer le temps écoulé entre deux états successifs de l'ATN,
- un module de réification des agents de coordination temporelle, dont nous parlerons plus en détails dans la section 6.5,
- 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. Cette fonction est appelée fonction de prédiction temporelle.

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

6.4.1 L'ATN

L'utilisation d'un ATN nous permet de modéliser le comportement de l'agent anytime. Chaque franchissement d'état dans l'ATN correspond à l'obtention d'un résultat dont la qualité est meilleure que celle du précédent. La figure 6.4 présente un premier exemple d'ATN similaire à celui que nous utiliserons dans notre modèle.

L'ATN est utilisé pour 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,

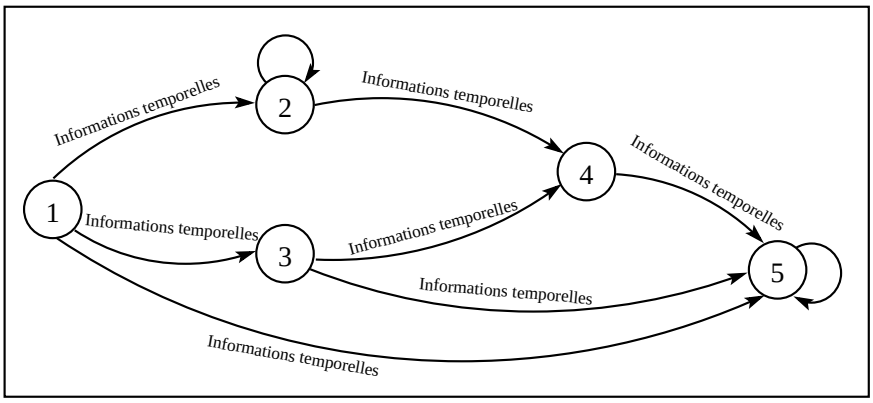


FIG. 6.4 – *Un premier exemple d'ATN*

– le nombre de communications,

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

Toutes ces informations sont nécessaires pour le fonctionnement de la fonction de discrétisation et de la fonction de prédiction temporelle.

Lors de la conception d'un agent anytime, l'ATN est initialisé pour modéliser le comportement de l'agent selon un algorithme anytime.

6.4.2 Le réseau d'accointances

Dans tout système multi-agent, 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 [Drogoul, 1993] ou par communications directes comme c'est beaucoup plus souvent le cas.

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. On peut notamment reprendre le cas du fonctionnement du modèle Aalaadin [Gutknecht and Ferber, 1999] et plus techniquement de la plate-forme MadKit qui implémente ce modèle.

En nous basant sur le modèle Aalaadin, ce réseau d'accointances est réifié 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 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.

6.4.3 La fonction de discrétisation

Le but de cette fonction est de trouver une unité de temps qui puisse servir à mesurer le temps écoulé entre deux états de l'ATN et qui soit plus grande que l'unité de temps de base (en général la milliseconde). Ces nouvelles unités de temps sont ensuite utilisées 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 illustrer notre modèle, nous nous basons volontairement sur un ATN simplifié (sous forme linéaire). La discrétisation va se faire de la façon suivante :

Etant donné un ATN avec n états (par exemple la figure 6.5, où n est égal à 5), $d_{i,i+1}$ $(1 \leq i \leq n-1)$ 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 et i+1.

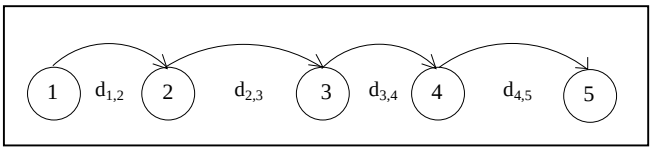


FIG. 6.5 – *Exemple d'ATN linéaire*

Alors, on obtient l'équation suivante :

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

Dans l'exemple de la figure 6.5, nous obtenons alors :

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

Le nombre de sous-divisions pour le premier intervalle étant initialement fixé, un algorithme (voir figure 6.6) 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. ε est une variable permettant d'allouer la précision du calcul effectué pour déterminer ΔT.

Après application de l'algorithme nous obtenons un ATN qui aura l'aspect présenté sur la figure 6.7.

L'algorithme présenté ici a été implémenté en langage JAVA et une simulation de cet algorithme est présentée (tableau 6.4.3) afin d'illustrer son fonctionnement. Dans les première et seconde colonnes 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

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 deux états i et i+1
L'algorithme :
$n_{1,2} \leftarrow 1$
$i \leftarrow 1$
$\Delta T \leftarrow 0$
TANT QUE (($|d_{i,i+1}$ - ΔT * $n_{i,i+1}| > \varepsilon$)
ET ((i+1) $\neq$ Dernier Etat))
i $\leftarrow$ 1
$n_{1,2} \leftarrow n_{1,2}$ + 1
ΔT = arroundi ($d_{i,i+1}$ / $n_{i,i+1}$)
TANT QUE ($|d_{i,i+1}$ - ΔT * $n_{i,i+1}| < \varepsilon$
ET (i+1) $\neq$ Dernier Etat))
i $\leftarrow$ i + 1
$d_{i,i+1} = d_{i,i+1} + d_{i-1,i}$ - ΔT * $n_{i-1,i}$
$n_{i,i+1}$ = arroundi ($d_{i,i+1}$ / ΔT)
FIN TANT QUE
FIN TANT QUE

FIG. 6.6 – *Algorithme de discrétisation du temps sur un ATN linéaire*

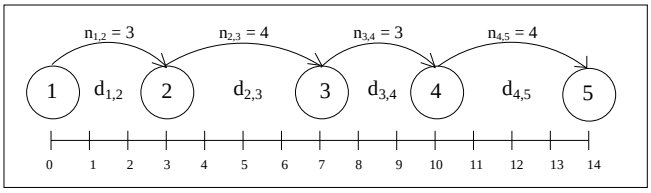


FIG. 6.7 – *Exemple de discrétisation du temps sur un ATN linéaire*

quatrième colonne représente les résultats obtenus après l'exécution de l'algorithme. Dans notre exemple, pour une valeur $\varepsilon = 5$, la valeur calculée de ΔT est 4. En résumé, basé sur une estimation moyenne des temps d'exécution, le but de cet algorithme est de permettre à l'agent de disposer d'un outil nécessaire 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. Cette partie de nos travaux 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 6.8) car les ATN que l'on est amené à utiliser sont rarement linéaires.

Il va s'agir de discrétiser le temps sur un ATN qui est assimilable à un graphe. Nous allons utiliser à la base le même principe : on commence par initialiser à *un* le premier nombre d'unité de temps pour la première transition, puis on calcule la première valeur

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
9	10	13255 ms	3314 ΔT
10	11	34256 ms	8564 ΔT
11	12	11284 ms	2821 ΔT
12	13	34975 ms	8744 ΔT
13	14	45975 ms	11493 ΔT
14	15	18975 ms	4744 ΔT
15	16	13975 ms	3494 ΔT
16	17	3495 ms	874 ΔT
Valeur de ε : 5			
Valeur calculée de ΔT : 4			

TAB. 6.1 – *Simulation de l'algorithme*

de temps discret (ΔT). Ensuite, on continue selon le même principe que pour l'algorithme précédent mais on va avoir recours cette fois-ci à un algorithme récursif. On obtient alors l'algorithme illustré sur la figure 6.9.

6.4.4 La fonction de prédiction temporelle

Dans un système multi-agent anytime, l'objectif est de fournir des résultats dont la qualité varie avec le temps alloué. Pour mettre en œuvre ce principe, la méthode que nous préconisons est 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 6.5. 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 respecter ce critère, 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 avons introduit 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

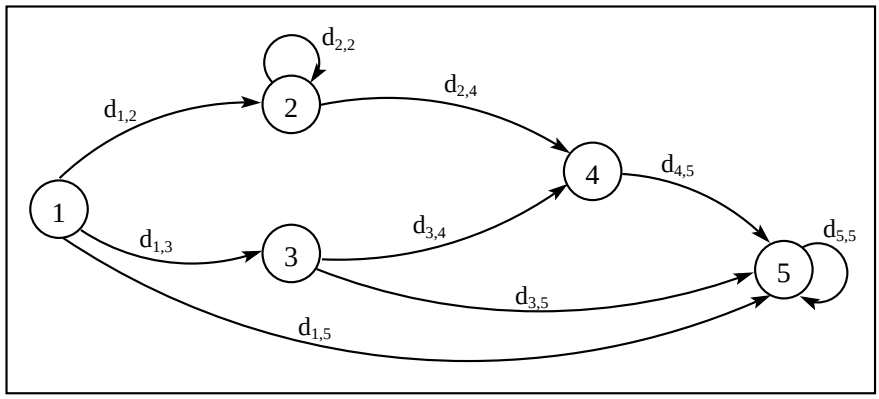


FIG. 6.8 – *Un exemple d'ATN avec des temps d'exécution*

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, nécessaire habituellement 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 jusquà l'état Y connaissant la durée moyenne qu'avait nécessité ce franchissement d'état par le passé et la durée courante d'exécution nécessaire pour franchir l'état précédent. On obtient l'algorithme précédent (figure 6.9).

6.5 Agents de coordination temporelle

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 dépasse un seuil et doit être contrôlé afin d'éviter une surcharge inutile du système par un nombre trop important de communications. 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 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 alors 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) réifier un nouvel agent de coordination temporelle et ainsi fonder un nouveau groupe, sinon.

L'agent de coordination temporelle est un agent léger [Wooldridge and Jennings, 1994]

L'algorithme :

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

FirstState $\leftarrow$ *Etat initial de l'ATN*
CurrentState $\leftarrow$ *Premier successeur de 'FirstState'*
$n_{FirstState,CurrentState} \leftarrow 1$
$\Delta T \leftarrow d_{FirstState,CurrentState} \ / \ n_{FirstState,CurrentState}$
TANT QUE CalculerAPartir (FisrtState, 0) est **Faux**
$n_{FirstState,CurrentState} \leftarrow n_{FirstState,CurrentState} + 1$
$\Delta T \leftarrow d_{FirstState,CurrentState} \ / \ n_{FirstState,CurrentState}$
FIN TANT QUE

La fonction *CalculerAPartir (state, error) :*
En sortie : une valeur booléenne

POUR TOUT *succ* successeur de *state* **FAIRE**
trans $\leftarrow$ la transition entre *state* et *succ*
SI *trans* n'a pas été visité
Mettre *trans* à visité
time $\leftarrow d_{state,succ}$ + error
$n \leftarrow$ arrondi (*time* / ΔT)
error $\leftarrow$ *time* - ΔT * n
SI error > ε **RETOURNER Faux**
SI CalculerAPartir (succ, error) est **Faux RETOURNER Faux**
FIN SI
FIN POUR
RETOURNER Vrai

FIG. 6.9 – *Algorithme de discrétisation du temps sur un ATN complexe*

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 pour cela doit coopérer avec d'autres agents de coordination temporelle afin de négocier l'importance relative de son groupe. Ceci se fait au moyen de priorités fixées sur les tâches effectuées par les agents. Réduire l'activité de certains groupes permet d'augmenter les performances d'autres groupes afin qu'ils aboutissent plus rapidement à une solution. De cette façon une solution partielle et progressive peut être construite.

6.6 Mise en œuvre du modèle ANYMAS

Pour la mise en œuvre du modèle ANYMAS, la plate-forme MadKit a été utilisée et plus particulièrement sa classe de base, la classe *Agent*, qui permet de créer des agents. L'agent anytime du modèle ANYMAS hérite directement de cette classe. Pour donner un bref aperçu de l'implémentation, on peut se reporter au diagramme de relation concernant l'agent anytime (voir figure 6.10). Les différents composants présentés au cours de ce chapitre se retrouvent dans cette implémentation, très souvent sous forme de classes.

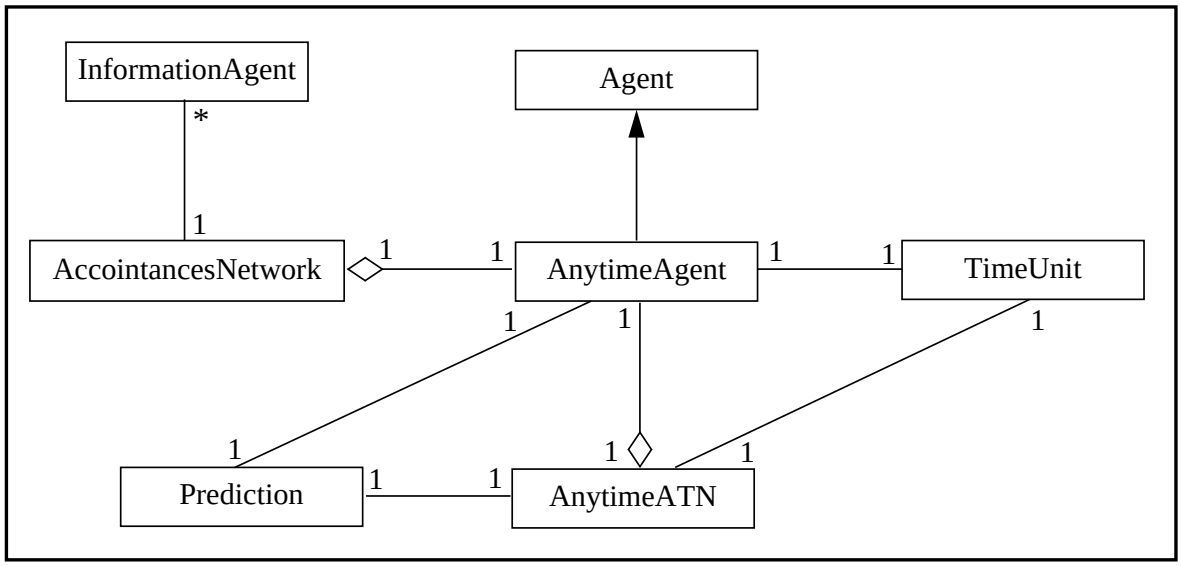


FIG. 6.10 – *Diagramme de relations pour l'agent anytime*

La classe *TimeUnit* est la classe qui permet de mettre en œuvre la fonction de discrétisation décrite précédemment. La classe *Prediction* permet de mettre en œuvre les fonctions de prédiction temporelle de l'agent anytime. La classe *AnytimeATN* permet de doter l'agent d'un ATN dédié à l'utilisation particulière de l'agent anytime. L'implémentation de l'ATN constitue une part importante de l'agent anytime. Son diagramme de relations est présenté plus en détails dans la figure 6.11.

Nous avons décrit dans ce paragraphe la mise en œuvre du modèle ANYMAS, essentiellement au travers de son composant de base : l'agent anytime. Nous ne nous attarderons pas sur l'implémentation de l'agent de coordination temporelle. Les propriétés à mettre en œuvre ont été décrites dans les paragraphes précédents et ne présentent pas de difficultés particulières.

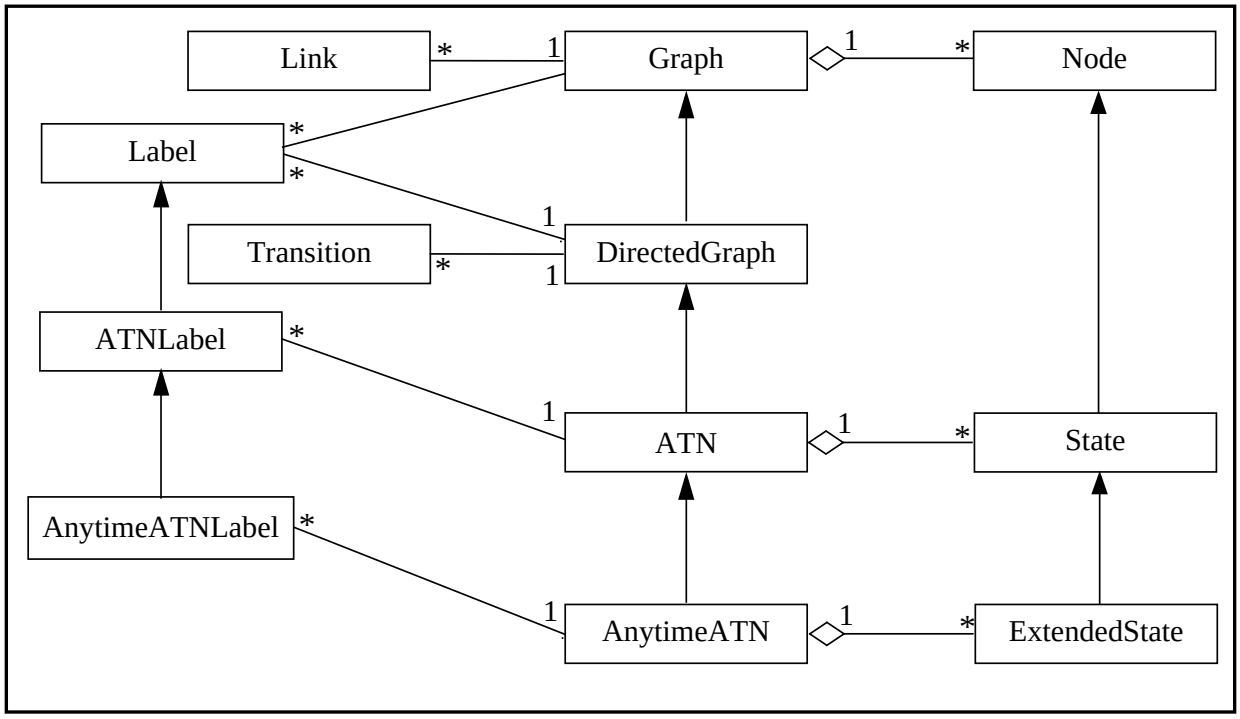


FIG. 6.11 – *Diagramme de relations pour l'ATN*

6.7 Conclusion

L'architecture ANYMAS est composée d'agents anytime reposant partiellement sur une architecture d'agent à base de composants [Guessoum and Briot, 1997] à laquelle des augmentations ont été faites. Nous avons ainsi ajouté un module d'introspection à l'architecture initiale de l'agent afin de permettre de prédire le temps nécessaire pour exécuter les tâches. Ce module peut modifier dynamiquement le comportement en fonction des prédictions effectuées précédemment si cela est nécessaire (c'est-à-dire si le temps restant pour terminer la tâche n'est pas suffisant). Une tâche étant composée de sous-tâches, la célérité d'un agent permet de mesurer le temps nécessaire à un agent pour effectuer une sous-tâche en mode apprentissage. Ces temps seront utilisés en mode fonctionnement réel pour effectuer des prévisions sur les temps d'exécution. Afin d'avoir un contrôle au niveau du Système Multi-Agent, les agents anytime peuvent créer des agents de coordination temporelle au moyen d'un module de réification d'agents temporels. Ces agents sont utilisés pour réduire le nombre de communications entre agents. Chacun de ces agents gère un groupe d'agents fortement liés par les communications échangées. En effet, l'exécution d'un SMA est très dépendante des communications entre agents. Selon le temps disponible, il est intéressant d'influencer le comportement des agents afin d'obtenir un résultat partiel exploitable. De plus, dans le modèle ANYMAS, nous encourageons des groupes d'agents afin d'obtenir des résultats en priorité et d'écarter d'autres groupes d'agents.

En résumé, le contrôle du temps d'exécution du SMA dépend du contrôle des interactions entre les groupes d'agents. Ce contrôle est effectué par les agents de coordination temporelle.

Chapitre 7

Réalisation d'un modèle de SMA distribué

Résumé

La réalisation d'un système multi-agent (SMA) lors de la modélisation d'un système informatique distribué nécessite de tenir compte de la distribution physique du système et non plus uniquement de la distribution logique du système informatique. Dans les SMA, nous tenions compte jusqu'à très récemment, uniquement de la distribution logicielle sous forme d'entités logicielles appelées agents. Pour prendre en considération la distribution physique des SMA, il est nécessaire de revoir leur conception classique. Nous allons présenter dans ce chapitre la réalisation d'un modèle de système multi-agent distribué au moyen de l'utilisation de la norme CORBA. Ce choix de la distribution selon CORBA repose en partie sur l'existence de spécifications de l'OMG (Object Management Group) pour la distribution des SMA ainsi que sur les facilités offertes par les ORB (Object Request Broker) pour la distribution d'objets. Dans ce chapitre, nous commençons par décrire la problématique de la distribution des systèmes multi-agents. Puis, pour répondre aux interrogations posées par cette problématique, nous présenterons la réalisation du modèle DISMAS. Ce modèle a été implémenté en utilisant une plate-forme de développement de SMA existante : MadKit [MADKIT, 2001], ainsi que le "broker" d'objets ORBacus.

7.1 Introduction

De nombreux systèmes informatiques sont physiquement distribués sur plusieurs machines. Dès lors que ces systèmes sont conçus suivant le paradigme agent, il devient nécessaire d'avoir recours à un système multi-agent physiquement distribué [Halpern, 1995] [Fagin et al., 1995]. Dans ce chapitre, il ne sera pas question de l'aspect temps réel. DISMAS ne possède pas de caractéristiques temps réel en propre et seul son couplage avec le

modèle ANYMAS nous permettra de concevoir des applications temps réel reposant sur les techniques anytime.

Les Systèmes Multi-Agents [Ferber, 1995] [Wooldridge and Jennings, 1994] physiquement Distribués (SMAD) doivent permettre à des agents présents sur des machines distantes de travailler ensemble de façon similaire à des agents regroupés sur une même machine. La seule restriction que l'on doit admettre est celle du coût des communications entre agents. En effet, lorsque les agents sont éloignés physiquement les uns des autres le coût des communications s'accroît rapidement.

La distribution physique des SMA se justifie essentiellement dans deux types d'applications :

1. Les applications où il est nécessaire d'avoir recours à une répartition de la charge. Il s'agit souvent d'applications qui fonctionnent avec un très grand nombre d'agents. On peut citer les applications de simulations d'écoulement de fluides ou de particules [Bertelle et al., 2000] mais aussi les simulations d'écosystèmes (fourmilières [Drogoul, 1993], insectes divers, bancs de poissons, etc.).
2. Les applications qui sont naturellement distribuées. Ces applications englobent les systèmes multi-utilisateurs où des acteurs interviennent depuis des endroits physiquement distants sur une même application [Durand, 1999] [Lesage, 2000]. Par exemple, il s'agit des applications d'aide à la décision où des acteurs utilisent des systèmes informatiques pour faciliter leur prise de décision [Boukachour et al., 1998].

Dans la suite de ce chapitre, nous présentons un modèle de SMA distribué (SMAD) que nous appelons DISMAS, abbréviation de DIStributed Multi-Agent System. L'objectif est de concevoir un modèle de distribution de SMA qui soit indépendant de la plate-forme et de montrer que l'on peut rendre « distribuable » une plate-forme qui ne l'était pas au départ. Nous avons choisi d'implémenter ce modèle en nous basant sur une plate-forme de développement de SMA existante : Madkit [MADKIT, 2001]. Ce modèle, de par sa généricité, pourrait être appliqué à d'autres plate-formes non distribuées.

7.2 Problèmatique de la distribution de SMA

La principale différence entre les SMA distribués et non-distribués réside dans la forme des communications qui, dans le premier cas, doivent passer par un réseau physique. En outre, la distribution d'un SMA nécessite de tenir compte de différents aspects qui interviennent lors du fonctionnement de celui-ci. Il s'agit par exemple de la nature de la distribution du SMA. Deux points de vue devront être abordés pour concevoir un système multi-agent distribué :

1. Le point de vue du SMA.
2. Le point de vue de l'agent.

7.2.1 La nature de la distribution

On peut, en effet, envisager plusieurs types de distribution de SMA qui correspondent aux différents types d'applications, évoqués en introduction :

- un SMA de SMA (voir figure 7.1) : plusieurs SMA sont répartis sur différentes machines et coopèrent pour former un SMA global ; typiquement ce sont les applications multi-utilisateurs ;
- un SMA réparti (voir figure 7.2) : les agents d'un même SMA sont répartis sur plusieurs machines. Typiquement, ce sont les applications à très forte charge qu'il est nécessaire de répartir.

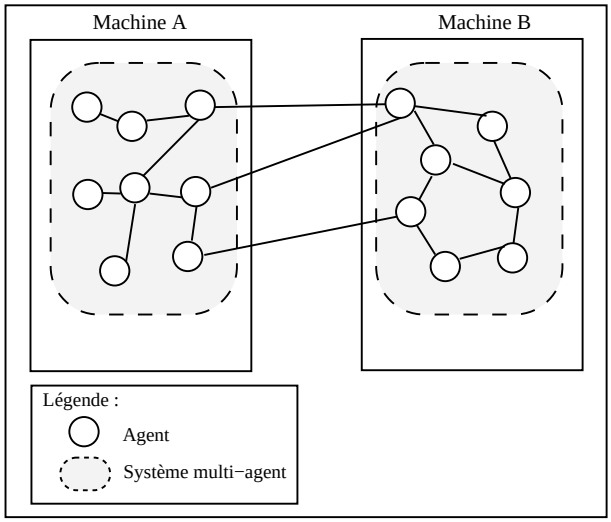


FIG. 7.1 – *Un SMA de SMA*

Selon la nature de la distribution, les difficultés rencontrées ne seront pas toujours les mêmes. Néanmoins, l'un des points qui apparaît essentiel est de prévoir l'interaction entre les agents dans le cadre particulier d'un SMA distribué. Cette interaction, pour qu'elle puisse avoir lieu, nécessite la mise en place de mécanismes permettant aux agents de connaître les agents présents sur les autres machines et par conséquent de connaître les machines disposant de SMA. Elle nécessite également d'étendre la communication en mode distribué.

7.2.2 La communication dans les SMA distribués

Les SMA sont composés d'agents autonomes qui coopèrent le plus souvent à l'aide de la communication [Ferber, 1995]. La communication est la base de la résolution coopérative des problèmes. Elle permet de synchroniser les actions des agents et de résoudre les

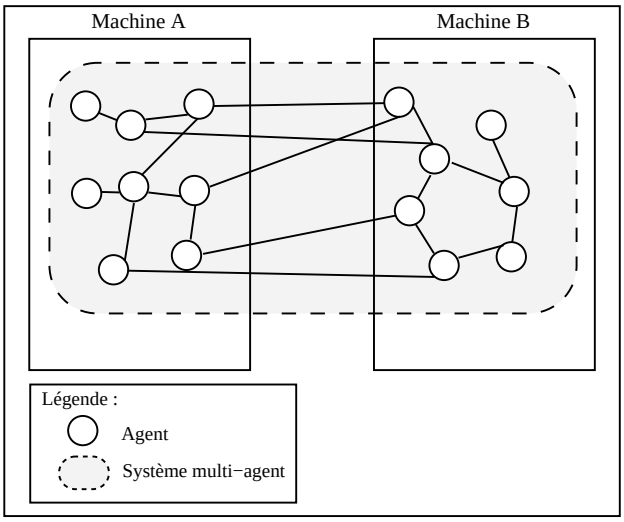


FIG. 7.2 – *Un SMA réparti*

conflits de ressources et de buts par la négociation. La communication entre les agents repose sur des mécanismes d'envoi de messages, de réception et de synchronisation. Nous sommes de ce fait en présence d'un système distribué qui peut être soit localisé sur une même machine, soit réparti sur plusieurs machines. Il faut alors prendre en compte cette répartition physique du SMA.

La communication dans un système multi-agent distribué est plus complète par rapport à celle présente dans les systèmes multi-agents classiques car elle fait intervenir la notion de réseau. L'envoi de messages à travers le réseau depuis un agent vers un autre se fait selon un protocole permettant de connaître l'origine physique du message ainsi que sa destination.

7.2.3 La distribution au niveau SMA

Dans un SMA distribué (SMAD), on peut considérer deux types de distribution :

- distribution dynamique, qui consiste à prendre en compte la mobilité des agents et éventuellement l'optimisation de cette distribution. L'optimisation consiste à trouver la meilleure façon de placer des agents sur les différentes machines disponibles de manière à répartir la charge et à réduire le coût des communications. En effet, on sait que, par nature, les communications distantes sont coûteuses en temps pour les agents.
- distribution statique, qui consiste à ne pas prendre en compte la mobilité des agents. En effet, on estime alors que le système est physiquement distribué lors de la conception. Cette distribution est plus simple que la précédente. Les problèmes auxquels on va devoir s'intéresser sont alors essentiellement des problèmes de communications

distantes entre agents localisés sur plusieurs machines.

Dans nos travaux, nous nous intéressons uniquement à la distribution statique des systèmes multi-agents. Les principales extensions que nécessitent ces systèmes multi-agents concernent les communications entre agents distants, la structure du réseau d'accointances des agents et la notion de groupe d'agents répartis. Cette notion de groupe est celle présentée dans le modèle Aalaadin [Gutknecht and Ferber, 1999] et implémentée dans la plateforme MadKit [Gutknecht and Ferber, 1997]. La répartition de cette notion de groupe sur des systèmes présents sur plusieurs sites ne pose pas de problèmes particuliers. Seul l'envoi de messages à un groupe réparti doit être revu.

7.2.4 La distribution au niveau agent

Un agent, dans le cadre d'un SMA distribué (SMAD), doit disposer de capacités accrues. Il doit notamment être en mesure de faire appel à des agents présents sur d'autres machines. Pour cela, il faut qu'il soit capable de demander des services sur d'autres machines et que la représentation de son réseau d'accointances [Ferber, 1995] lui permette de lier un agent à une machine distante. Il doit pouvoir disposer de capacités à envoyer des messages à des agents distants.

Des extensions par rapport à l'architecture de base d'un agent (voir figure 7.3) devront permettre à un agent de fonctionner dans un environnement distribué. Ces extensions seront fonction de l'architecture de base de l'agent et du type d'agent concerné.

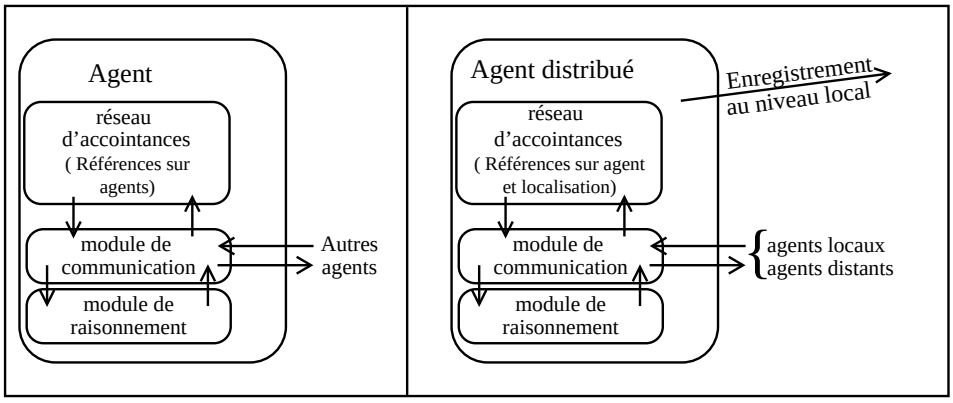


FIG. 7.3 – *Agent/Agent distribué*

7.3 Un modèle pour la distribution de SMA : DISMAS

Nous avons défini la problématique de la conception de SMAD. Nous allons donner un modèle satisfaisant tant au niveau SMA qu'au niveau agent. Dans un SMAD, il est

nécessaire que chaque agent puisse communiquer avec des agents situés sur des machines distantes. Il est donc indispensable que les agents disposent de fonctions de communication distante. De plus, une structure permettant de faire le lien entre les différents SMA présents sur des machines distantes doit être mise en place. Nous proposons à cet effet, un modèle agent basé sur l'utilisation d'objets CORBA (voir figure 7.4).

7.3.1 Un modèle agent pour DISMAS

Dans un système où des agents sont présents sur plusieurs machines distantes, il est nécessaire de disposer d'informations sur ces machines. Par exemple :

- Quelles sont les machines sur lesquelles se trouvent des agents avec lesquels on pourrait éventuellement communiquer ?
- Sur quelle machine se trouve tel ou tel agent ?

Le principal aspect des SMAD qui nous intéresse est l'envoi et la réception de messages à distance. Afin de gérer la disponibilité des machines, nous introduisons un objet CORBA qui contient la liste des machines sur lesquelles se trouvent des SMA. Cet objet sera géré par un agent qui sera chargé de mettre à jour cette liste. Un autre agent permettant de connaître la liste des agents présents sur la machine est introduit afin de pouvoir localiser les agents. Pour l'envoi ou la réception de messages sur d'autres machines, on a recours à deux agents qui se chargent de la réception et de l'émission des messages au moyen d'objets CORBA. Nous allons décrire en détail dans la section 7.4 chacun des agents que nous venons d'évoquer. Cet ensemble d'agents forme la première partie de notre modèle représenté dans la figure 7.4. La seconde partie de ce modèle repose sur la conception d'agents distribués et donc la prise en compte de la distribution au niveau des agents qui serviront à concevoir les applications basées sur des SMAD (voir paragraphe 7.3.2).

7.3.2 La distribution au niveau agent

Les SMA étant constitués d'agents capables de communiquer, les capacités de ces agents doivent être augmentées pour les agents d'un SMAD. Nous allons présenter ici un modèle d'agent distribué nommé « DistributedAgent ».

Les principales extensions d'un agent pour qu'il puisse fonctionner dans un environnement distribué sont les suivantes :

- extension des fonctions de communication : un agent doit pouvoir envoyer et recevoir des messages en provenance d'un agent situé sur une autre machine ;
- extension du réseau d'accointances : il doit contenir la localisation des agents ;
- enregistrement des agents au niveau local afin que les agents distants puissent connaître la liste des agents présents sur une machine.

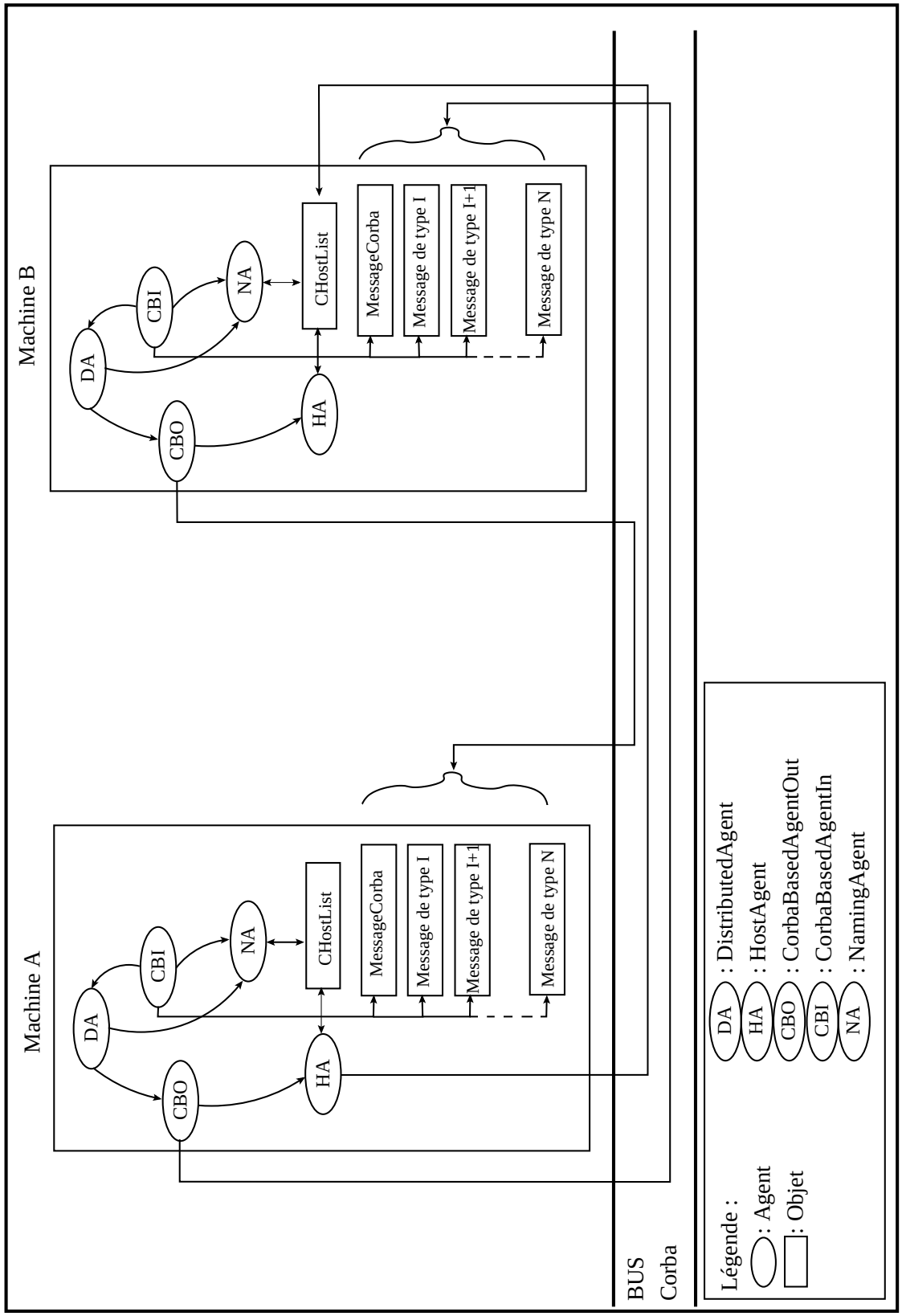


FIG. 7.4 – *DISMAS : modèle de distribution de SMA*

7.4 Mise en œuvre du modèle DISMAS

Nous présentons dans cette section l'implémentation du modèle de système multi-agent distribué présenté dans la figure 7.5. Une modélisation en UML (Unified Modelization Language) de notre implémentation se trouve en figure 7.6. L'implémentation du modèle DISMAS repose sur l'utilisation de la plateforme MadKit [MADKIT, 2001] [Gutknecht et al., 2000]. Le modèle DISMAS repose par conséquent beaucoup sur l'utilisation du modèle Aalaadin [Gutknecht and Ferber, 1999].

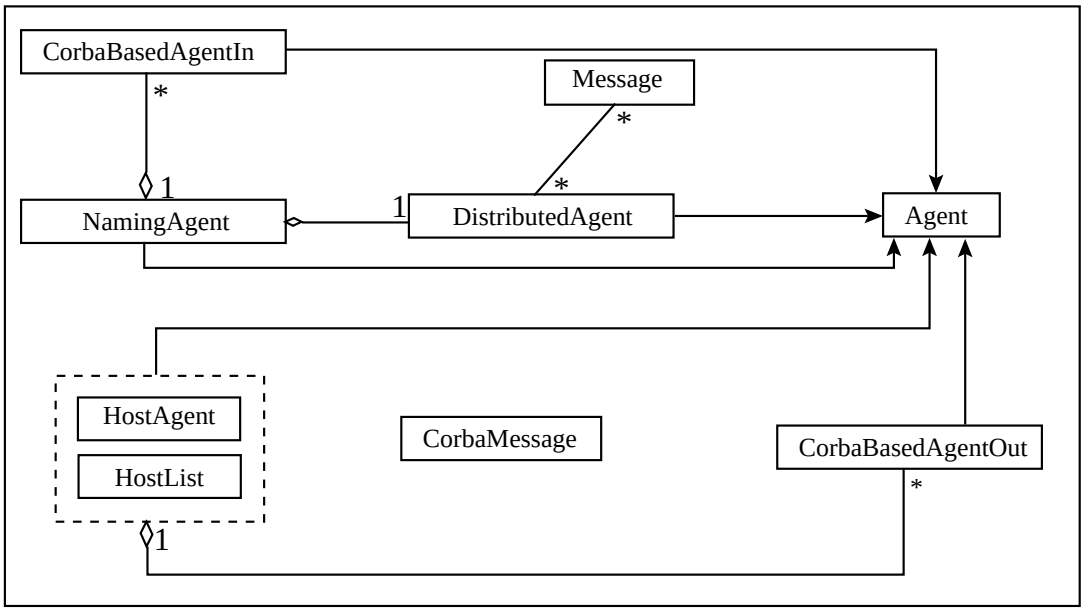


FIG. 7.5 – *Diagramme de relations*

7.4.1 Les différents composants du modèle

Dans ce paragraphe, nous allons décrire les agents et les objets présents dans le modèle DISMAS.

7.4.1.1 L'agent « DistributedAgent »

Cet agent constitue l'extension principale de l'agent de base d'un SMA classique. Il permet de disposer d'agents capables d'évoluer dans un système physiquement distribué, notamment grâce à des fonctions de communications étendues et à la possibilité d'interroger l'agent « NamingAgent ». Il doit se faire enregistrer auprès de celui-ci.

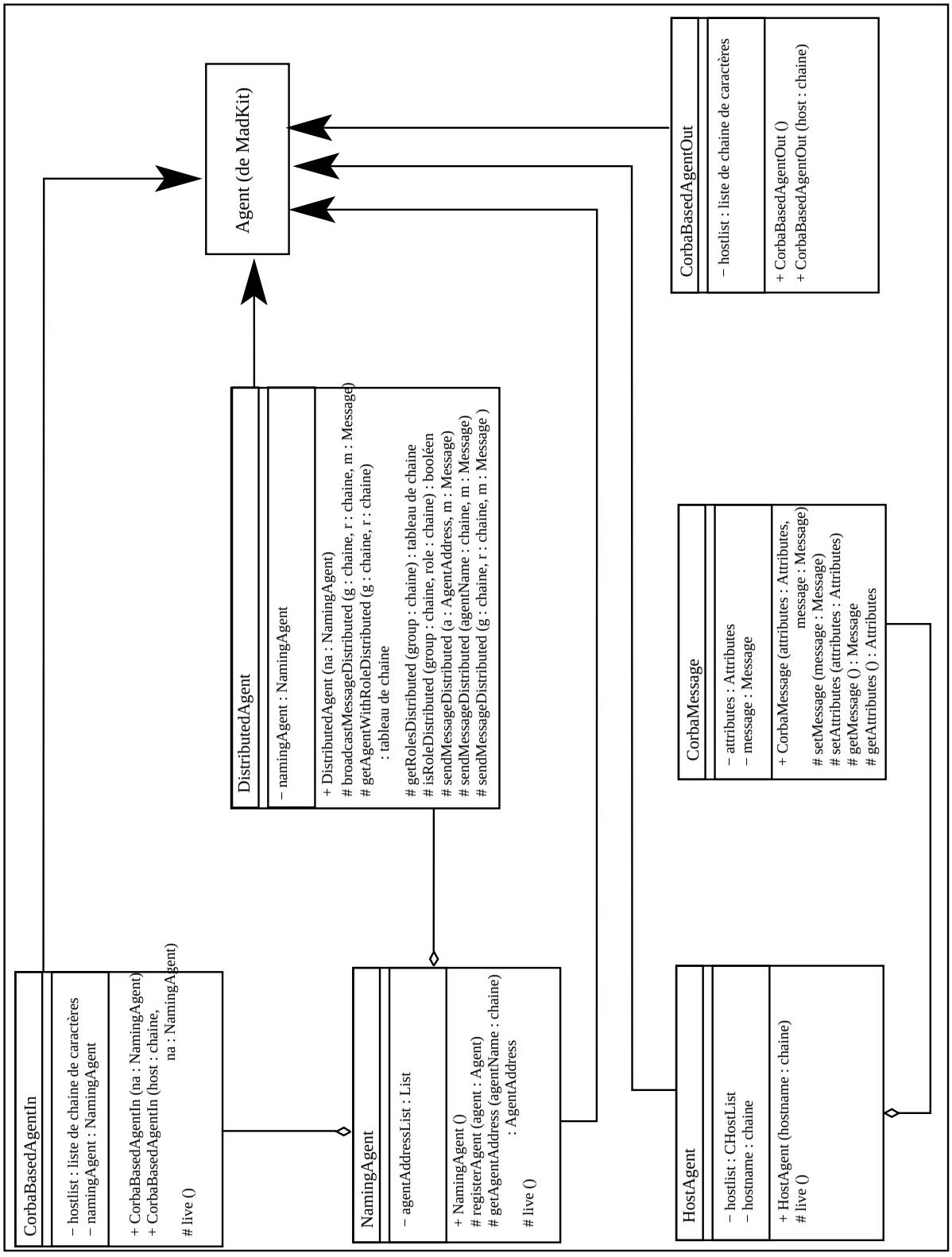


FIG. 7.6 – *Diagramme de classes du système DISMAS*

7.4.1.2 L'objet « CHostList »

Il s'agit d'un objet CORBA présent sur chaque machine comportant un SMA distribué. Il permet de connaître la liste des machines distantes (noms ou numéros IP) connues par la machine locale.

7.4.1.3 L'agent « HostAgent »

Cet agent doit au moyen des objets « CHostList » gérer la liste des machines qui sont actuellement disponibles. Pour cela, il interroge constamment les objets « CHosList » des machines distantes qu'il connaît déjà et met à jour l'objet « CHostList » local.

7.4.1.4 L'agent « NamingAgent »

Il est chargé de gérer les agents en indiquant leur emplacement physique à partir de leur adresse (AgentAddress) ou de leur nom. Il peut être interrogé par d'autres agents afin de fournir ces informations.

Les agents de type « DistributedAgent » doivent se faire enregistrer auprès de cet agent pour pouvoir ensuite être repérés par des agents situés sur des machines distantes. Cet agent met en œuvre une partie des propriétés de repérage d'agents décrite dans les travaux de l'OMG [OMG, 1997].

7.4.1.5 L'objet « CorbaMessage »

Il s'agit là d'un message spécial permettant de gérer les messages à envoyer sur le bus CORBA. Il hérite de la classe « Message » définie dans MadKit. Cet objet permet de véhiculer des informations supplémentaires concernant les messages devant être envoyés à distance. Par exemple, il s'agit d'informations concernant l'émetteur du message (machine sur laquelle il est situé), type de diffusion du message, etc.

7.4.1.6 Les objets « CorbaMessage(Type) »

Il s'agit d'objets CORBA permettant de gérer l'émission/réception de messages distants selon le type initial de messages reçus. Ces types peuvent être les suivants : (Type) = { ACL | KQML | String | ...}. Il faut créer autant de messages/d'objets de ce type qu'il existe de type de messages pouvant être échangés par les agents. Ces objets encapsulent les messages des agents lors de leur envoi à distance.

7.4.1.7 L'agent « CorbaBasedAgentIn »

Cet agent gère les messages en entrée de la plate-forme. Il reçoit les messages qui proviennent de machines distantes, y effectue un traitement et les réachemine. Il possède

donc un module de traitement des messages qui lui permet de rendre son aspect (type[3]) initial au message. Il s'agit d'un des deux agents qui serviront d'intermédiaires entre les agents des différentes machines.

7.4.1.8 L'agent « CorbaBasedAgentOut »

Il gére les messages en sortie de la plate-forme, c'est-à-dire qu'il envoie sur le bus CORBA les messages des agents après les avoir transformés en objets CORBA. Il s'agit là du second agent servant d'intermédiaire entre les agents des différentes machines.

7.4.2 La gestion des messages

La gestion des messages constitue un des points essentiels du modèle DISMAS. On considère deux cas :

- l'émission/réception de messages entre des agents situés sur la même machine ;
- l'émission/réception de messages entre des agents situés sur des machines différentes.

Le premier cas est résolu par les moyens classiques employés dans les SMA, alors que le second cas va être traité dans le modèle DISMAS grâce à des agents et des objets CORBA. En effet, comme nous venons de le décrire précédemment (composants du modèle), nous avons recours à des agents (*CorbaBasedAgentIn, CorbaBasedAgentOut*) pour le traitement en entrée et en sortie des messages qui sont échangés entre les machines au moyen d'objet CORBA (*CorbaMessage*). Les agents auront surtout un travail d'interprétation des objets CORBA afin d'en extraire le message d'origine et sa destination pour le réacheminer correctement.

7.5 Application

Le modèle DISMAS permet de réaliser des applications en mode distribué basées sur l'utilisation du paradigme multi-agent. Comme nous l'avons déjà évoqué, nous considèrons essentiellement deux types d'applications :

- les applications qui nécessitent une répartition de la charge sur plusieurs machines ;
- les applications qui sont naturellement distribuées.

Dans le cadre des premières, on peut citer comme exemple les applications de simulation d'ecosystèmes telles que les fourmilières [Drogoul, 1993], de simulation d'écoulement de fluide [Bertelle et al., 2000], etc. Il s'agit donc d'applications ayant recours à un nombre très important d'agents. Dans ces applications, la répartition de la charge initiale n'est pas toujours une répartition optimale et cette charge peut varier au cours du temps.

3. Il s'agit de la forme que possède le message lorsque les agents communiquent en mode non distribué.

Par conséquent, il est nécessaire d'avoir recours à la migration des agents entre les différentes machines afin d'optimiser cette répartition de la charge. Des politiques de migration doivent être mises en place pour accompagner ces migrations. La migration évoquée ici ajoute un degré de difficulté supplémentaire à la distribution du système. Nous n'avons pas traité la migration au sein de notre modèle DISMAS. Nous ne considérerons donc pas les applications de ce type pour l'instant.

Les applications naturellement distribuées sont des applications qui ont un fonctionnement nécessitant d'avoir recours à d'autres applications mais qui possèdent néanmoins une certaine autonomie. Il s'agit par exemple des applications multi-utilisateurs. Chaque utilisateur possède une application qui repose sur un SMA et fonctionne de façon autonome. Néanmoins, elle ne constitue qu'un point d'entrée dans un système plus vaste qui fera intervenir d'autres utilisateurs et d'autres applications. On peut éventuellement considérer des applications sans utilisateur comme les serveurs d'information chargés de répondre aux demandes effectuées par d'autres applications.

Pour illustrer notre propos, nous avons utilisé deux applications de ce type :

- une application de gestion d'agendas multiples pour l'organisation de réunions qui sera présentée brièvement dans le paragraphe suivant ;
- une application de gestion de marché pour l'aide à la décision qui sera présentée en détails dans les deux derniers chapitres de ce document.

La réalisation repose sur l'utilisation du modèle DISMAS.

7.5.1 Gestion d'agendas multiples

Nous nous plaçons dans le cadre d'un ensemble d'utilisateurs disposant d'un agenda électronique leur permettant de gérer leurs rendez-vous et leurs réunions. L'application *agenda électronique* sera donc présente sur chaque poste utilisateur. Elle devra permettre à des groupes d'utilisateurs souhaitant se réunir d'organiser leurs réunions et donc de trouver des créneaux horaires communs. Le principe de fonctionnement est le suivant :

1. Un utilisateur fait une demande de réunion.
2. La demande est examinée par l'agenda électronique des utilisateurs concernés.
3. Chaque agenda fait une ou des propositions de date à son utilisateur.
4. Chaque utilisateur valide une ou plusieurs propositions.
5. Lorsque toutes les propositions reviennent à l'émetteur de la demande, un traitement est effectué afin de trouver une date commune.
6. Cette date est ensuite renvoyée vers chaque utilisateur pour confirmation.
7. S'il y a refus de la date par un utilisateur, on doit reprendre à l'étape 5 afin de trouver une nouvelle date possible sinon on valide définitivement la date.

Chaque utilisateur se voit doté d'une priorité hiérarchique qui lui permettra d'imposer plus ou moins facilement ses dates de réunion. Il pourra aussi définir un niveau d'importance par rapport à ses réunions de façon à permettre éventuellement de reporter une réunion.

L'ensemble du problème sera traité au moyen du paradigme agent et, du fait de sa distribution naturelle, il reposera sur l'utilisation du modèle DISMAS.

7.5.1.1 La notion de rendez-vous

L'un des aspects essentiels du système se situe dans la notion de rendez-vous. Un rendez-vous définit les éléments qui vont permettre au système d'aller chercher des informations sur d'autres machines afin de déterminer les possibilités de réunion, les dates qui sont déjà bloquées, etc. Dans notre système, l'agenda d'un utilisateur sera constitué d'une liste de rendez-vous. L'organisation d'une réunion se traduira par une demande de rendez-vous à effectuer.

Les caractéristiques d'un rendez-vous sont les suivantes :

- un sujet/titre ;
- une date et une heure ;
- une durée ;
- une liste de personnes ;
- une priorité relative.

Effectuer une demande de réunion va donc consister à envoyer une proposition de rendez-vous. La priorité fixée ne sera que relative à l'utilisateur effectuant la demande, c'est-à-dire à sa position hiérarchique. En effet, cette priorité sera recalculée en fonction de la position hiérarchique des personnes impliquées dans la réunion. La date et l'heure constitueront une base pour la détermination des dates possibles. Dans le meilleur des cas, cette date conviendra dès le départ et par conséquent aucun travail réel du système ne sera nécessaire autre que l'envoi des demandes vers les autres utilisateurs et la réception de l'acceptation de la date de réunion.

7.5.1.2 Agentification

L'agentification consiste à décrire les agents qui vont composer le système. Ce système repose sur le modèle DISMAS et il va, par conséquent, s'agir de déterminer quels seront les agents permettant la gestion propre de l'agenda. Ces agents seront classés dans deux catégories :

- les agents qui agiront uniquement de façon locale,
- les agents qui interagiront avec des agents présents sur d'autres machines.

Les agents de cette deuxième catégorie reposeront sur l'utilisation du modèle d'agent distribué DISMAS. Parmi ces agents distribués, nous allons retrouver un agent de gestion de profil utilisateur qui est décrit dans le chapitre 10 concernant l'application de gestion de marché. Cet agent possède comme principal objectif de maintenir une liste des utilisateurs connectés afin que l'utilisateur courant puisse connaître les utilisateurs avec qui il peut dialoguer par le biais de l'application. Cet agent va utiliser la notion de groupe d'agents répartis en s'intégrant dans le groupe « ProfileAgent ». De cette façon, il pourra envoyer des messages aux agents du même groupe que lui pour demander des informations sur les utilisateurs connectés sur les autres machines. Pour connaitre les machines qui sont connectées, il utilisera la liste mise à disposition dans l'objet CORBA « CHostList ». Cet agent reçoit aussi des messages des autres agents de son groupe ; il s'agit soit de demandes d'information, soit de réponses aux questions qu'il avait lui-même posées.

D'autres agents vont être chargés de la gestion propre de l'agenda et vont devoir gérer les demandes de réunion effectuées par l'utilisateur et celles effectuées par les autres agents présents sur d'autres machines. Ces agents de gestion d'agendas auront en charge de trouver (calculer) les disponibilités dans l'agenda de l'utilisateur et de les soumettre aux autres agents concernés par la réunion. En effet, contrairement aux agents de gestion de profils, il ne s'agira pas d'envoyer les demandes à toutes les machines connectées mais uniquement aux machines dont l'utilisateur est concerné par la réunion.

7.5.1.3 Réalisation du système

Pour des raisons de commodité et de portabilité, le système global est réalisé dans le langage JAVA. En effet, le système DISMAS utilisé avait été développé dans ce langage. Une interface utilisateur permet d'effectuer des demandes de réunions et de recevoir des réponses. Elle permet également d'avoir accès à un agenda comportant la liste des rendez-vous existants avec le niveau de priorité relatif.

La partie concernant la gestion de profil utilisateur nous a permis de valider le modèle DISMAS et l'implémentation qui en a été faite au dessus de la plate-forme MadKit.

7.6 Conclusion et perspectives

Dans ce chapitre, est présenté DISMAS, un modèle de SMA distribué reposant sur l'utilisation d'objets CORBA. Une implémentation de ce modèle a été réalisée au dessus de la plateforme MadKit [MADKIT, 2001]. Nous aurions pu tenter d'implémenter notre modèle DISMAS au dessus d'autres plate-formes. Ceci ne nous a pas été matériellement possible.

Pour donner un aperçu du domaine d'application de nos travaux, nous avons présenté une application dont la réalisation s'appuie sur le modèle DISMAS. Une seconde

application est présentée dans la suite de ce document.

L'extension prochaine du modèle DISMAS concernera la prise en compte de la mobilité des agents. Cette extension est nécessaire pour étendre les domaines d'application, notamment aux applications nécessitant une répartition de charge dynamique.

Si l'on envisage de faire travailler des plate-formes SMA de différentes natures, il sera aussi nécessaire de prendre en compte l'interopérabilité des SMA. Dans ce cas, les recommandations de l'OMG [OMG, 1997] sont une base de travail nécessaire.

Le modèle de distribution DISMAS est utile pour servir de base à la réalisation d'applications reposant sur des SMA distribués et anytime.

Chapitre 8

Interrogation de systèmes d'information distribués par des SMA anytime et distribués

Résumé

L'emploi de systèmes d'information (SI) distribués dans les systèmes d'aide à la décision (SAD) nous a amenés à nous poser le problème de l'interrogation de ces SI. Ces SI sont très souvent distribués physiquement sur plusieurs sites distants. Les SAD sont eux-mêmes multi-utilisateurs et répartis sur plusieurs sites. Comme nous l'avons déjà évoqué dans ce document, nous utilisons le paradigme multi-agent pour concevoir des systèmes d'aide à la décision. En outre, les transactions effectuées par les décideurs doivent être traitées en temps contraint. Nous sommes donc amenés à concevoir un SMA anytime et distribué qui nous permettra de développer des SAD capables d'effectuer des interrogations sur des SI dans un environnement distribué et en temps contraint. Nous allons présenter, dans ce chapitre, un modèle pour l'interrogation de SI par des SMA anytime et distribués.

8.1 Introduction

Les systèmes d'aide à la décision (SAD) doivent être en mesure de fournir des informations plus ou moins précises en fonction du temps dont disposent les décideurs pour prendre une décision. Cela se traduit par une interrogation sous forme anytime des systèmes d'information sur lesquels repose le SAD. Les différents utilisateurs qui interviennent dans un SAD peuvent être détenteurs d'informations qu'un autre utilisateur pourra utiliser. L'interrogation devra donc être en mesure de recueillir l'information depuis plusieurs sites. Il s'agit d'un des aspects qui rendent nécessaire la distribution du système.

Les différents niveaux de précision d'une réponse à une interrogation vont dépendre de la forme globale du système d'information. Par conséquent, il sera nécessaire pour le système de connaître l'architecture générale de la globalité des systèmes d'information.

Dans ce chapitre, nous commençons par décrire un SMA anytime et distribué (voir section 8.2) basé sur les modèles ANYMAS et DISMAS. Puis, nous décrivons la structuration globlale du système d'information (voir section 8.3) pour permettre par la suite l'interrogation anytime de SI (voir section 8.4). Nous concluons ce chapitre sur les exploitations possibles du modèle que nous avons présenté.

8.2 Un SMA anytime et distribué

Pour la réalisation d'un modèle de SMA qui soit à la fois anytime et distribué, nous allons nous appuyer sur nos travaux concernant la réalisation du modèle ANYMAS et du modèle DISMAS. Dans ces deux modèles, nous avions traité séparément les deux aspects qui nous intéressent. Nous allons donc étudier les contraintes nécessaires à la fusion des deux modèles.

Dans le modèle ANYMAS, on peut extraire les caractéristiques suivantes :

- un modèle d'agents anytime ;
- des agents de coordination temporelle pour le contrôle des groupes d'agents anytime.

La conception d'un SMA basé sur le modèle ANYMAS repose donc sur l'utilisation des agents anytime issus du modèle ANYMAS.

Dans le modèle DISMAS, on extrait les caractéristiques suivantes :

- un modèle d'agents distribués avec des fonctions de communication étendues ;
- des agents assurant les fonctions de communications distantes, le repérage des agents et le nommage des agents.

Un système informatique utilisant DISMAS doit donc reposer sur des agents distribués et une instanciation des agents nécessaires aux fonctions de communications distantes, au repérage des agents ainsi qu'à leur nommage.

Par conséquent la conception d'un SMA à la fois anytime et distribué va essentiellement reposer sur la fusion du modèle d'agent anytime et du modèle d'agent distribué ainsi que la reprise des caractéristiques de ces deux modèles d'agent.

8.2.1 Un modèle d'agent anytime et distribué

Dans notre modèle de SMA anytime et distribué, les agents qui constituent le système informatique possèdent à la fois des comportements anytime et des capacités à fonctionner en mode distribué. Nous présentons ici un modèle d'agent qui constitue la brique de base

des systèmes informatiques que nous souhaitons concevoir. Les caractéristiques présentes dans ces agents sont donc les suivantes :

- un comportement anytime ;
- un automate (un ATN simplifié) permettant de modéliser les points d'arrêt de l'algorithme anytime ;
- une horloge permettant la mesure du temps écoulé entre les états de l'automate ;
- des fonctions de communications permettant :
  - la mesure du nombre de communications échangées entre les agents ;
  - l'envoi de messages à des agents situés sur une autre machine.
- une fonction de « discrétisation » du temps ;
- une fonction de prédiction du comportement temporel.

Dans les systèmes d'aide à la décision, les propriétés anytime seront utilisées au niveau de l'extraction progressive des informations. Au début de l'extraction, on aura des informations superficielles, générales et peu précises. Plus on avancera dans le temps, plus l'information extraite sera précise et importante. Ces extractions anytime de l'information nécessitent une mise en place particulière de la structure des systèmes d'information utilisés.

8.3 Structure d'un système d'information pour les SAD

On souhaite mettre en place des interrogations qui permettent d'extraire de façon progressive de l'information afin de disposer très rapidement d'informations exploitables. Puis, au fur et à mesure de l'écoulement du temps, l'extraction d'information va se faire de plus en plus précise. On va donc chercher l'information plus loin dans les SI en procédant à des recoupements, en analysant les domaines connexes, etc.

Les systèmes d'information utilisés interviennent dans un schéma global qui permettra de structurer l'information au sein du SAD. Ce schéma permet de classifier l'information suivant son niveau de précision et les relations entre les différentes sources d'information. En effet, la sémantique des différentes informations peut être la même mais se situer sur différents niveaux de précision. Prenons un exemple permettant d'illustrer notre propos : on considère le classement par catégories d'âge de la population d'un pays donné. L'information que l'on possède peut alors exister sous différentes formes exprimant chacune un niveau de précision : le pourcentage d'individus, le nombre exact d'individus, le nombre d'individus par sexe, le classement par région, etc. Suivant le niveau de précsion que l'on souhaite atteindre, on ne fait pas intervenir les mêmes systèmes d'information. C'est pourquoi, il est absolument nécessaire de structurer les systèmes d'information afin d'obtenir

un schéma global qui permettra par la suite de mettre en place les interrogations anytime effectuées par le SAD.

La structuration des systèmes d'information pour leur exploitation par des SAD consiste essentiellement à établir des relations permettant de préciser les équivalences sémantiques et le degré de précision entre ces systèmes d'information.

8.4 Interrogation anytime de SI

Dans les SAD, un utilisateur cherche à recueillir un maximum d'éléments pertinents lui permettant de prendre sa décision dans les meilleures conditions possibles ; c'est-à-dire une décision pertinente dans les meilleurs délais. Pour effectuer ce recueil d'information, il est amené à poser des questions au SAD. Ces questions sont alors traduites sous forme d'interrogations qui sont effectuées sur les systèmes d'information. Dans un SAD, il peut être nécessaire pour le décideur de prendre une décision dans l'urgence. Il faut donc que les interrogations fournissent des résultats dans les meilleurs délais mais avec un maximum de précision. En outre, le décideur peut à tout moment décider de réduire le temps alloué pour l'interrogation car la situation exige qu'il prenne sa décision plus tôt qu'il ne l'avait initialement prévu. Nous avons donc recours aux techniques anytime pour effectuer ces interrogations.

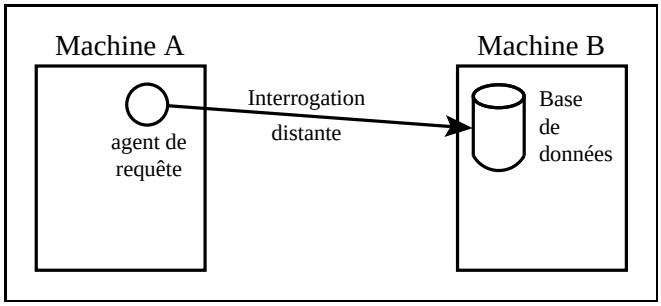
Les interrogations étant très souvent multi-critères, nous avons donc eu recours à des agents ayant des caractéristiques anytime pour effectuer ces interrogations. Lorsqu'un agent reçoit une demande, il peut effectuer la requête correspondante et éventuellement faire appel à d'autres agents pour compléter le résultat par rapport aux différents critères connexes à la question initiale.

A partir du schéma global reliant les systèmes d'information, il est nécessaire de déterminer les requêtes élémentaires qu'il est possible d'effectuer et de les mettre sous forme anytime. Pour mettre en œuvre ces interrogations anytime, nous allons présenter le modèle d'agent de requête, chargé d'effectuer les requêtes sur les SI, que nous avons conçu. Il repose sur le modèle d'agent anytime et distribué que nous avons présenté dans le chapitre précédent.

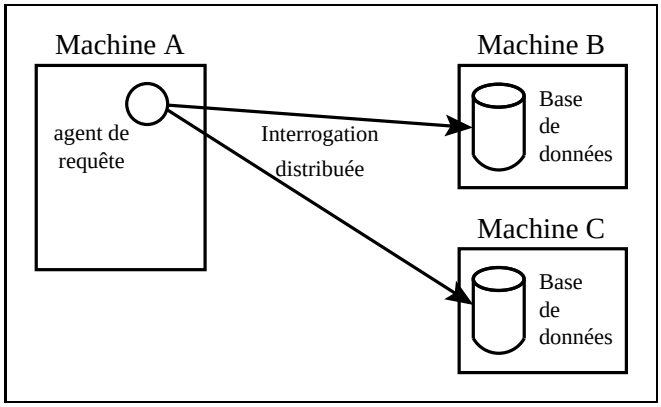
8.5 Agents de requête

Un agent de requête est un agent qui possède des capacités à effectuer des interrogations sur des systèmes d'information distants (voir figure 8.1(a)) et distribués (voir figure 8.1(b)). Une interrogation consiste à extraire de l'information depuis des SI au moyen de requêtes SQL (on suppose que les SI reposent sur des bases de données relationnelles). Les agents reçoivent ces interrogations sous forme de messages, qui seront alors traités pour

donner des requêtes simples ou complexes. Les requêtes complexes seront décomposées en requêtes plus simples. Ces requêtes peuvent être sérialisées ou traitées en parallèle. Chaque interrogation correspond à un traitement intelligent par un agent. Ce traitement peut nécessiter de faire appel à d'autres agents de requête qui ont alors en charge le traitement d'une partie de l'interrogation initiale. Cela peut s'avérer nécessaire lorsque l'interrogation initiale est complexe et nécessite d'avoir recours à différentes catégories d'informations.



(a) Interrogation distante



(b) Interrogation distribuée

FIG. 8.1 – *Interrogation de SI par des agents de requête*

La modélisation du comportement d'un agent de requête est effectuée au moyen d'un ATN (Augmented Transition Network). Chaque état de l'ATN correspond à l'exécution d'une requête SQL ou à l'envoi d'un message à un autre agent. On effectue l'exécution des requêtes SQL au moyen d'un objet « threadé » et on utilise des points de rencontre lorsque l'attente du résultat est nécessaire pour continuer le processus de traitement de l'interrogation.

Dans certains cas d'utilisation particuliers, on a recours à des interrogations périodiques permettant d'anticiper les demandes éventuelles et d'augmenter ainsi les perfor-

mances du système pour certaines interrogations récurrentes. Pour effectuer ces anticipations, on a construit des bases de cas qui permettent de juger des interrogations qui sont effectuées dans des situations analogues ou encore à des intervalles de temps précis (même question posée toutes les heures, etc.). Des résultats sont alors disponibles au moment de l'interrogation. Dans ce cas, on peut décider d'utiliser directement le résultat disponible sans effectuer d'autres calculs ou encore d'attendre l'occurence du prochain résultat.

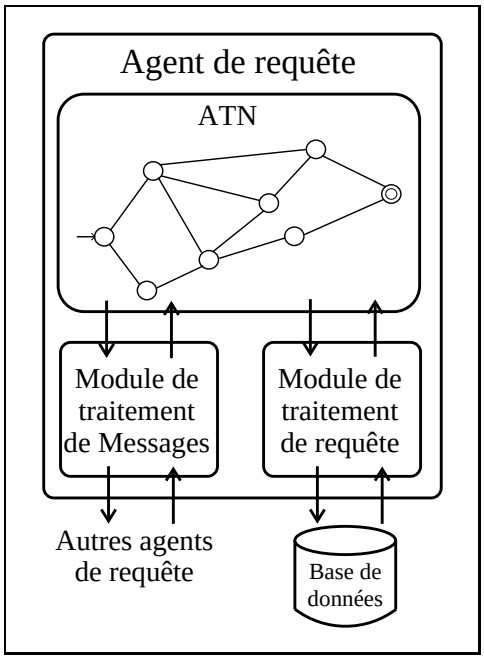


FIG. 8.2 – *Agent de requête*

Pour résumer, un agent de requête a les caractéristiques suivantes (voir figure 8.2) :

- un ATN permettant de modéliser son comportement ;
- la capacité à traiter les interrogations, c'est-à-dire le traitement des messages reçus par l'agent ;
- la capacité à situer physiquement les systèmes d'information disponibles (module de traitement des requêtes) ;
- la capacité à réenvoyer des interrogations à d'autres agents de requête.

8.6 Conclusion et perspectives

Dans ce chapitre, nous avons présenté la structure générale des interrogations de systèmes d'information par des SMA anytime et distribués dans le cadre d'un système d'aide à la décision. Il vient terminer la seconde partie de ce document consacrée à la réalisation d'un modèle de système multi-agent anytime et d'un modèle de système multi-agent distribué. Ces deux modèles sont complémentaires et sont utilisés pour la conception de SAD

dans un contexte multi-utilisateurs. La problématique principale des SAD concerne l'exploration et l'extraction d'informations depuis des systèmes d'information distants pour fournir aux décideurs les éléments pertinents qui leur permettront de prendre une décision adéquate face à une situation donnée.

Chapitre 9

Présentation et modélisation d'une application de gestion de marché

Résumé

La gestion de marchandises en ville reste un problème très difficile à résoudre. Il s'agit, dans ce type d'applications, de passer des commandes (des transactions) vers des fournisseurs qui doivent ensuite livrer les marchandises à leurs clients. Dû au fonctionnement en flux tendu (un minimum de stocks), ces transactions doivent être effectuées dans des délais suffisament raisonnables pour que les transactions soient bénéfiques. De plus, les clients peuvent avoir recours à différents fournisseurs pour un même produit et doivent donc choisir celui qui a le meilleur rapport prix/délais de livraison. La première phase de cette étude concerne la commande des produits et le respect des délais de livraison. Une seconde phase concernant la gestion du transport de ces marchandises en ville ne sera pas traitée pour l'instant. Notre travail consiste à 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. Pour modéliser le problème, nous nous sommes basés sur un ensemble de systèmes d'information distants représentant l'information disponible sur les fournisseurs, les produits, les clients, etc. L'approche choisie pour concevoir ce système d'aide à la décision est celle des Systèmes Multi-Agents. Pour la réalisation du système, nous nous sommes appuyés sur la plate-forme de développement MadKit et sur l'outil ORBacus. Les systèmes d'information sont mis œuvre au moyen du SGBD Oracle.

9.1 Introduction

Les travaux abordés dans ce chapitre concernent la mise en œuvre d'un système d'aide à la décision pour la gestion de marchandises en milieu urbain. Il s'agit de modéliser et

de concevoir un système informatique qui doit permettre à des utilisateurs d'interroger des Systèmes d'Information (SI) et, en fonction des résultats, de prendre les meilleures décisions possibles concernant la gestion de marchandises en ville.

Nous allons procéder en deux temps :

1. Présentation générale du système de gestion de marché (section 9.2).
2. Modélisation du système (section 9.3).

9.2 Présentation générale

Ce système repose en grande partie sur des systèmes d'information (voir paragraphe 9.2.1). Le système d'aide à la décision doit effectuer des interrogations sur des systèmes d'information distants afin de fournir aux utilisateurs les informations dont ils ont besoin pour prendre une décision (voir paragraphe 9.2.2). Le système est multi-utilisateurs et accède à des systèmes d'information distants. Il devra donc être « distribuable » sur plusieurs sites (voir paragraphe 9.2.3). Nous avons développé un prototype d'application de gestion de marchandises en milieu urbain qui permet de valider les travaux effectués (voir paragraphe 9.2.4). Nous allons maintenant décrire chacun de ces aspects plus en détails.

9.2.1 Conception d'un système d'information

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

Certaines informations disponibles sont privées, c'est-à-dire 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 publiques (un fournisseur met à disposition les produits qu'il est susceptible de fournir avec tous les renseignements nécessaires).

9.2.2 Interrogation de systèmes d'information par des agents

Les fonctionnalités principales considérées dans le système concernent l'interrogation des systèmes d'information afin de répondre aux questions 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 9.1.

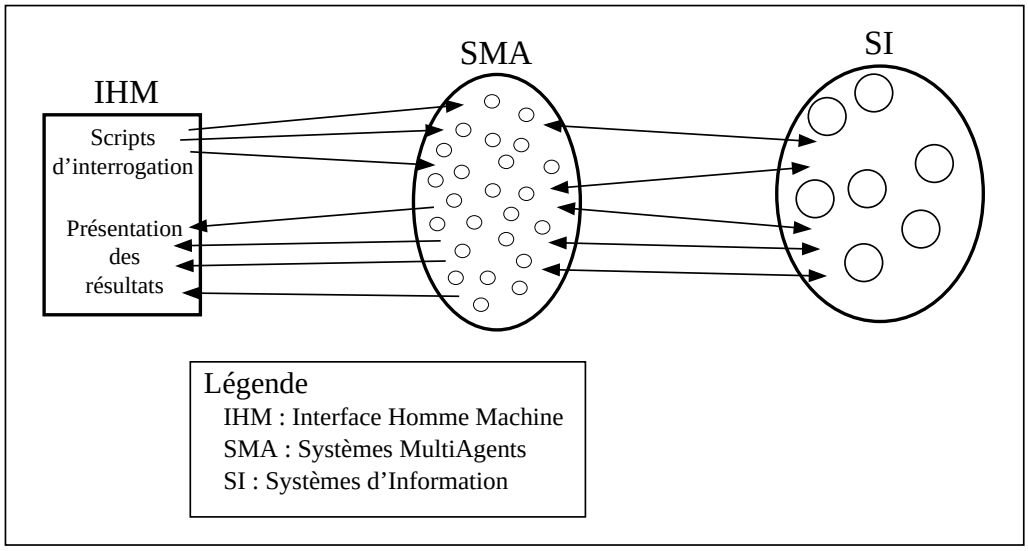


FIG. 9.1 – *Schéma d'interaction de l'application entre l'utilisateur et le système*

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, on s'appuie sur le modèle ANYMAS (conception d'un SMA anytime, chapitre 6).

9.2.3 Conception d'un système distribué

Le cadre de conception du système de gestion de marchandises est le suivant. Il existe un ensemble de clients qui souhaitent passer des commandes auprès d'un ensemble de fournisseurs. L'ensemble des acteurs (clients et fournisseurs) sont physiquement distribués. Il est donc nécessaire que le système soit capable de fonctionner en mode distribué. Pour mettre en œuvre cet aspect, on s'est appuyé sur le modèle DISMAS (distribution de SMA, chapitre 7).

9.2.4 Réalisation d'un prototype

Pour valider l'ensemble de cette étude, nous avons réalisé un prototype d'application mettant en œuvre un système d'aide à la décision pour la gestion de marchandises en milieu urbain (Voir figure 9.2).

Ce système est réalisé dans le langage Java, utilisé tout au long de nos travaux. Pour la conception des systèmes d'information, nous avons utilisé les services offerts par le SGBD Oracle. Les SI sont matérialisés donc sous forme de bases de données relationnelles stockées sous Oracle. Pour la partie multi-agent, nous avons utilisé la plate-forme de développement MadKit et plus particulièrement ses extensions concernant l'implémentation du modèle

ANYMAS ainsi que du SMA distribué DISMAS.

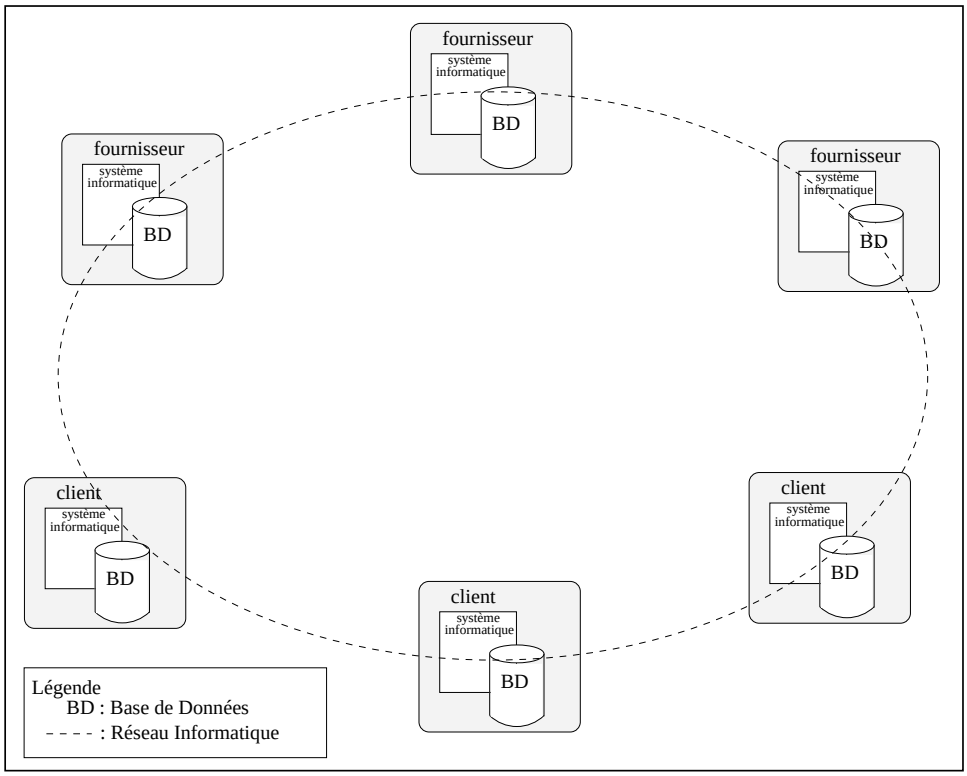


FIG. 9.2 – *Application de gestion de marchandises en milieu urbain*

9.3 Modélisation de l'application

Dans cette section, nous allons présenter les différents éléments qui composent la modélisation de notre système de gestion de marché. Dans un premier temps, nous identifions les acteurs du système : les fournisseurs et les clients. Dans un second temps, nous identifions les objets manipulés par le système. Puis dans un troisième et dernier temps nous identifions les interrogations qui peuvent y être effectuées.

9.3.1 Les acteurs du système de gestion de marché

Comme nous l'avions déjà évoqué, il existe deux types d'acteurs dans notre système : les clients et les fournisseurs. Nous allons donner ici une modélisation de ces acteurs.

9.3.1.1 Les fournisseurs

Les fournisseurs ont pour objectif de vendre certains produits aux clients. Cela suppose qu'ils possèdent des stocks de marchandises à leur disposition. Ces stocks sont modélisés

par un système d'information. Un fournisseur va avoir pour principal objectif de gérer les commandes effectuées par ses clients et de répondre aux questions diverses 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 atteindront 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 ré-alimentera 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 de modifier ce seuil en fonction de la fréquence des commandes effectuées par les clients.

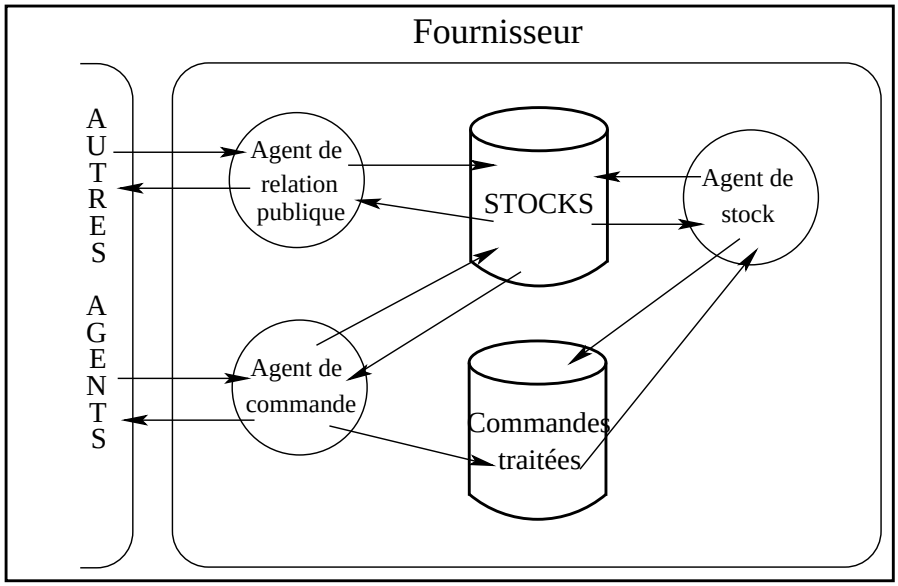


FIG. 9.3 – *Modélisation du fournisseur*

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 et doit par conséquent être capable de fonctionner en environnement distribué. Pour chaque commande traitée, il renvoit un message vers le client pour lui fournir 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 le paragraphe 9.3.3. Il doit aussi être en mesure de communiquer avec les agents situés au niveau des clients.

Vu la nature des agents constituant le fournisseur (voir figure 9.3), on s'est basé sur

le modèle DISMAS pour sa réalisation.

9.3.1.2 Les clients

Les clients ont deux objectifs :

- récupérer suffisament d'information 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 et se traduit par la possibilité d'effectuer des interrogations sur les systèmes d'information mis à leur disposition. Ces interrogations seront décrites dans le paragraphe 9.3.3. Nous avons modélisé ces interrogations par des agents de requête (voir chapitre 8), selon un comportement anytime et avec la possibilité d'évoluer en environnement distribué. Chaque interrogation est effectuée au travers d'une interface et contrôlée par un agent d'interface qui la re-dirige 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 coté fournisseur. Cette commande est ensuite enregistrée dans le système d'information du client avec les informations qui lui sont liées (nom du fournisseur, délai de livraison, etc.).

La modélisation du client (voir figure 9.4) repose sur l'utilisation d'une interface Homme-Machine (IHM) permettant à l'utilisateur d'interagir avec le système informatique et avec un SMA permettant l'interaction avec les autres composantes du système (fournisseurs, systèmes d'information, etc.).

9.3.2 Les objets de l'applications

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 entités manipulées, 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 en détails ces différentes entités.

9.3.2.1 Les produits

Les produits constituent les éléments de base du système de gestion de marché. Un produit possède un certain nombre de caractéristiques : nom, prix, quantité.

Ces données servent à alimenter les bases de données « stocks » présentes au niveau de chaque fournisseur. Lorsqu'une commande est effectuée, elle précisera le produit concerné, la quantité commandée et le prix de commande.

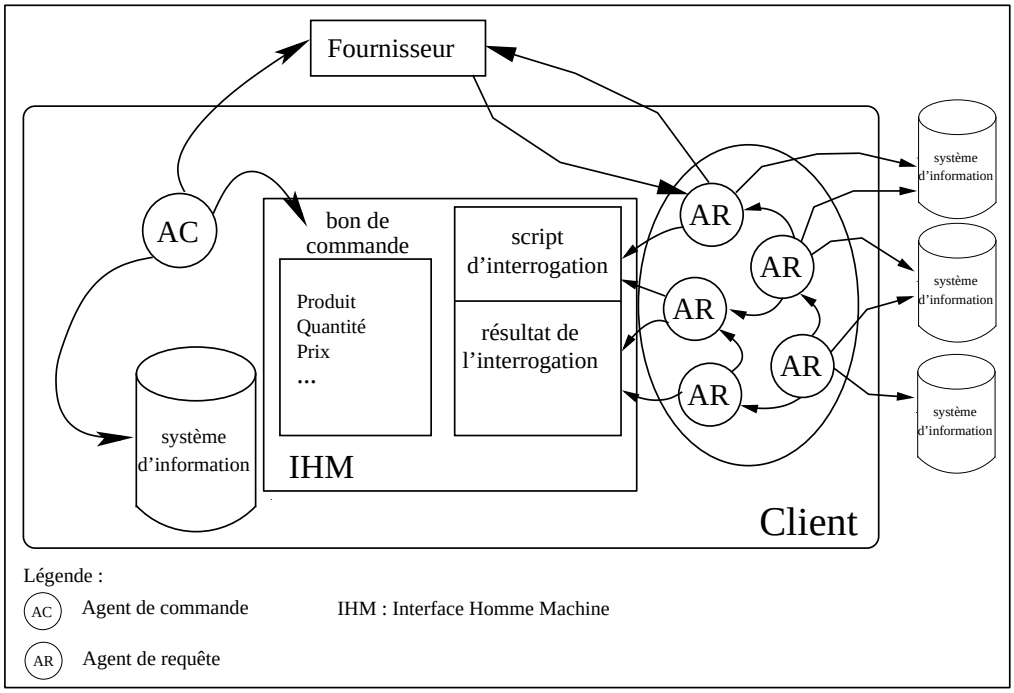


FIG. 9.4 – *Modélisation du client*

9.3.2.2 Les commandes

Les commandes sont des éléments échangés entre clients et fournisseurs. Ces échanges se font au moyen d'*agents de commandes* qui échangent des messages et traitent ces commandes. Le traitement d'une commande par un agent situé du côté fournisseur consiste à vérifier la disponibilité du produit en stock puis 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 enlever des stocks les produits vendus (décrémenter les quantités). Mettre à jour la base de données *Commandes* consiste à reprendre 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.

9.3.3 Les interrogations

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

- les interrogations directes : ce sont les interrogations qui consistent à poser des questions directement à un fournisseur (par exemple, quel prix vendez-vous le produit X) ;
- 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).

Nous allons maintenant décrire les interrogations exploitées dans notre système suivant.

9.3.3.1 Les interrogations directes

Nous décrivons ici les interrogations permettant d'obtenir des informations simples et précises. La liste de ces interrogations pourra être augmentée. Nous en donnons quelques unes :

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

9.3.3.2 Les interrogations générales

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 :

- quel est le cours moyen du produit X sur les six derniers mois ?
- quels sont les fournisseurs susceptibles de fournir le produit X ? avec quel délai moyen ?
- quelle quantité d'un produit X, un fournisseur F peut-il me fournir ?
- etc.

9.4 Conclusion

Nous avons présenté dans ce chapitre une application de gestion de marchés. La description des principaux composants de l'applications nous a permis de dégager une modélisation suivant le paradigme multi-agents. Cette application va permet de valider nos travaux concernant les systèmes multi-agents anytime et les systèmes multi-agents distribués. La réalisation effective de cette application est présentée dans le chapitre 10 qui constituera une description plus technique de l'application.

Chapitre 10

Réalisation de l'application de gestion de marché

Résumé

Dans ce chapitre, nous allons présenter la réalisation de l'application de gestion de marché que nous avons implantée pour valider nos travaux. Cette application est décomposée en plusieurs composants que nous présentons d'abord séparément. Nous illustrons la description de l'application à l'aide de captures d'écran. Nous justifions nos choix techniques lorsque cela s'avère nécessaire. Nous présenterons également des alternatives de développement que nous avons envisagées mais que nous n'avons pas pris en compte. Notons cependant que la réalisation de cette application n'est qu'au stade d'un premier prototype. Pour sa réalisation, nous avons choisi de travailler dans le domaine du commerce en gros de fruits et légumes. Le développement et la réalisation complète de l'application pour illuster l'ensemble de nos travaux sont en cours.

10.1 Mise en œuvre de l'aspect multi-utilisateurs

L'application de gestion de marché permet à des clients et des fournisseurs d'échanger des informations et de communiquer au travers d'un réseau informatique global. Il s'agit donc d'une application multi-utilisateurs permettant à des utilisateurs de travailler dans un même environnement logiciel tout en étant éloignés physiquement. Nous avons choisi de créer cet environnement logiciel au moyen d'une procédure de connexion et déconnexion des utilisateurs. Chaque utilisateur dispose d'une interface homme-machine lui permettant d'avoir accès à cet environnement. Cette connexion au système permet aussi à un utilisateur de se faire connaître des autres utilisateurs. Nous allons présenter la mise en œuvre de tous ces aspects techniques. Dans un premier temps, nous présentons la mise en œuvre de la procédure de connexion au réseau. Dans un second temps, nous présen-

tons la mise en œuvre de la procédure de gestion des connexions et représentation des utilisateurs.

10.1.1 Connexion et déconnexion des utilisateurs

Chaque utilisateur se connecte au moyen d'un identifiant et d'un mot de passe. Ces informations sont stockées dans une table qui est actuellement centralisée dans une base de données Oracle. Cette table permet de disposer d'informations sur les utilisateurs enregistrés. Elle se présente sous la forme suivante lors de sa création en langage SQL :

```
CREATE TABLE Profiles (
    idProfile VARCHAR (28) NOT NULL,
    password VARCHAR (28) NOT NULL,
    Nom VARCHAR (28) NOT NULL,
    Prenom VARCHAR (28) NOT NULL,
    Type VARCHAR (12) NOT NULL,
    Societe VARCHAR (28) NOT NULL,
    PRIMARY KEY (idProfile),
    CONSTRAINT CHK_TYPE CHECK (Type IN ('Client', 'Fournisseur'))
);
```

L'ensemble des informations qui concernent l'utilisateur connecté sont chargées par l'agent de gestion de profils que nous décrirons dans le paragraphe suivant.

10.1.2 Agent de gestion de profil

Cet agent sera appelé par la suite *agent de profil*. Il a pour objectif de gérer les informations liées aux utilisateurs qui se connectent et se déconnectent. Il interroge ses homologues sur les autres machines connectées en réseau afin de connaître tous les utilisateurs connectés. Il s'agit donc d'un agent ayant des capacités à communiquer et à travailler en mode distribué. Il hérite de l'agent *DistributedAgent*. L'application de gestion de marché repose sur l'utilisation du modèle DISMAS (voir chapitre 7).

Le comportement de l'agent est le suivant :

1. Envoi d'un message en mode diffusion multiple pour demander à tous les agents de profils de donner les noms et prénoms des utilisateurs connectés.
2. Lecture du premier message disponible dans la boîte aux lettres de l'agent.
3. Traitement du message et envoi d'une réponse à l'émetteur s'il agit d'une demande d'information.
4. Envoi d'une demande d'information à l'émetteur.
5. Retour en 2.

10.2 Implémentation du système d'information

Dans cette application de gestion de marché, les utilisateurs sont, soit des clients soit des fournisseurs, qui achètent et vendent des produits. Ces produits vont constituer la partie principale du système d'information. La table *Produit* est créée de la manière suivante :

```
CREATE TABLE Produit (
    idProduit VARCHAR (28) NOT NULL,
    Nom VARCHAR (28) NOT NULL,
    Prix FLOAT (8)  NOT NULL,
    Quantite INTEGER NOT NULL,
    Type VARCHAR (30) NOT NULL,
    Commentaires VARCHAR (50),
    PRIMARY KEY (idProduit),
    CONSTRAINT CHK_PRIX CHECK (Prix>0)
);
```

Cette table est présente au niveau de chaque fournisseur et permet de connaître les produits disponibles (si quantité >0), leurs prix et éventuellement d'autres informations supplémentaires.

Les changements de prix sont stockés dans une table permettant d'historiser les prix des produits afin d'extraire ces informations par la suite et de modéliser le comportement du marché. La description de cette table est la suivante :

```
CREATE TABLE Prix (
    idProduit VARCHAR (28) NOT NULL,
    Prix FLOAT (8) NOT NULL,
    Validité DATE NOT NULL,
    CONSTRAINT FK_IdProduit FOREIGN KEY (idProduit)
    REFERENCES Produit (idProduit)
);
```

Il s'agit d'une table présente au niveau de chaque fournisseur. Le champs *Validité* permet de spécifier la date du début d'entrée en vigueur de ce prix. De cette façon, il est possible de modéliser l'évolution du prix d'un produit dans le temps. Une table historisant les produits vendus permet de stocker les prix pratiqués et les quantités vendues au cours du temps.

10.3 Implémentation de l'interface

L'interface homme-machine de la gestion de marché permet aux utilisateurs d'être identifiés, soit en tant que fournisseurs de produits, soit en tant que clients. Cette interface présente les utilisateurs connectés et donc atteignables, pour commercer. On considère dans un premier temps que l'ensemble des fournisseurs disponibles sur le marché restent connectés en permanence.

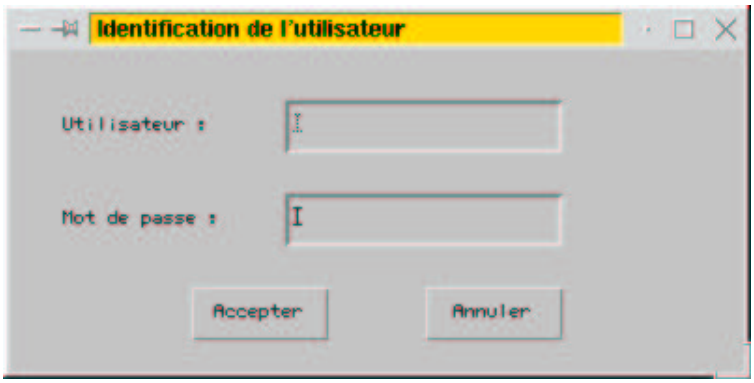


FIG. 10.1 – *Fenêtre d'identification des utilisateurs*

La distinction entre les utilisateurs et l'interface qui leur est liée se fera au moment de leur connexion (voir capture d'écran en figure 10.1). Deux types d'interface seront disponibles :

- l'interface pour les utilisateurs de type fournisseur ;
- l'interface pour les utilisateurs de type client.

10.3.1 L'interface du côté fournisseur

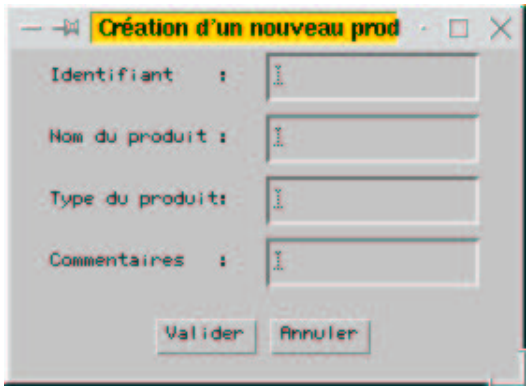


FIG. 10.2 – *Boîte de saisie pour la création d'un nouveau produit*

Les moyens d'action des fournisseurs sont réduits. Le fournisseur aura en charge de valider les commandes, contrôler la gestion des stocks faite par l'agent de stock (valider

les choix de l'agent de stock) et décider des produits dont il sera le fournisseur ou pas. Il pourra créer de nouveaux types de produits dont il sera le fournisseur. Pour décider des produits dont il sera le fournisseur, il s'agira de cocher une case parmi la liste des produits répertoriés. Il validera la quantité de produits que l'agent de stock a décidé de réapprovisionner. Pour l'ajout d'un nouveau produit, il utilise un formulaire de saisie qui lui permet de mettre à jour une nomenclature des produits répertoriés (voir capture d'écran en figure 10.2). Cette nomenclature correspond à une table *Produit* simplifiée (sans notion de prix et quantité), qui doit donc être mise à jour.

La mise à jour de la table des produits répertoriés ne peut avoir lieu que si le produit ne l'a pas déjà été.

FIG. 10.3 – *Boîte de choix pour les produits qu'un fournisseur décide de vendre*

10.3.2 L'interface du côté client

Un client doit être en mesure d'effectuer des commandes ou des demandes d'informations sur des produits. Toutes ces demandes se font au moyen d'écrans de saisie normalisés. Par exemple, la saisie d'une commande se fait au moyen de la boîte de saisie présentée en figure 10.4.

Ces demandes seront récupérées par des agents d'interfaçage avant d'être transformées en requêtes et traitées par des agents de requête spécifiques. Chaque agent de requête est en mesure de traiter un type d'interrogation particulier. Le traitement de cette interrogation se fait selon des techniques anytime que nous avons mises en œuvre spécifiquement pour chaque interrogation. Le traitement immédiat dans la première phase de l'interrogation anytime donne en général un résultat assez brut. Ce résultat peut-être complété

FIG. 10.4 – *Boîte de saisie des commandes*

par des interrogations supplémentaires. Une autre phase est l'analyse des résultats avec les heuristiques qui ont été indiquées pour chaque interrogation (par exemple, les effets de saison sur la vente de certains produits et donc de leur prix). Une dernière phase de l'interrogation anytime est constituée par la présentation des résultats qui peut être plus ou moins élaborée. Chacune de ces phases peut être décomposée selon la complexité des interrogations effectuées.

Il y a donc autant d'écrans d'interrogations qu'il y a de types d'interrogations. Les types d'interrogations sont suffisamment génériques pour couvrir un grand panel d'interrogations possibles.

10.4 Choix techniques effectués

Le premier choix technique que nous avons eu à effectuer concerne l'agentification de notre système de gestion de marché. Pour cela, nous avons utilisé la plate-forme de développement MadKit, présentée dans ses grandes lignes en annexe. Cette plate-forme a l'avantage d'être très légère et générique. Ceci nous a permis de développer les extensions nécessaires à la mise en œuvre des modèles DISMAS et ANYMAS sans trop de difficultés. Ces extensions ont essentiellement concernées l'adjonction de fonctionnalités aux agents de MadKit : ajout de méthodes d'envoi de messages (pour intercepter les communications, les comptabiliser ou les envoyer à distance), ajout de capacités de calcul (pour ANYMAS notamment), ajout d'une fonction d'horloge, définition de nouveaux types de messages, etc. La seule difficulté que nous avons rencontrée concerne l'impossibilité de surcharger les méthodes d'envoi de messages dans MadKit. En effet, la communication en mode distribué, mais aussi le contrôle des communications dans ANYMAS nous ont obligé à redéfinir de nouvelles méthodes pour l'envoi de messages.

L'ensemble de la plate-forme MadKit a été réalisé dans le langage JAVA. Nous avons

donc réalisé l'ensemble de nos développements dans ce langage. Pour la partie CORBA, nous avons choisi d'utiliser un ORB disponible gratuitement et qui permet de développer en langage JAVA (l'ORB ORBacus, version 3.X.). Au début de nos développements, la version 4.0 de ORBacus n'existait pas encore, c'est pourquoi nous avons continué d'exploiter les versions 3.X. De plus, dans la version 4.0, de nouveaux services ont été mis en place mais ils ne nous auraient pas été utiles.

L'implantation des systèmes d'information a nécessité l'emploi d'un système de gestion de bases de données (SGBD) relationnel. Nous avons opté pour le SGBD Oracle car (1) il est puissant et dispose de beaucoup d'outils, (2) c'est un des SGBD les plus utilisés au monde, (3) nous en disposons d'une version sous le système d'exploitation Linux et (4) il nous est le plus familier. Oracle permet de mettre en place des connexions distantes facilement en JAVA au moyen des librairies JDBC (Java DataBase Connectivity). L'implémentation de notre application sous d'autres SGBD aurait facilement pu se faire à condition que ces derniers disposent de moyens d'interfaçage avec le langage Java. Ce qui est le cas de beaucoup de SGBD du marché.

10.5 Conclusions et perspectives

Nous avons présenté dans ce chapitre une partie de l'implémentation de notre application de gestion de marché. Il est à préciser que cette impémentation n'est pas encore complètement terminée au moment de la rédaction de ce document. Cette application repose sur les modèles DISMAS et ANYMAS. Elle est par conséquent distribuée au moyen de la technologie CORBA, base du modèle DISMAS. Des algorithmes anytime sont utilisés pour l'extraction d'informations depuis des systèmes d'information distants. Ces systèmes sont implantés au moyen de tables situées dans des bases de données Oracle. Nous envisageons la possibilité de mettre en œuvre un éditeur de transactions d'interrogation sous la forme d'un assistant qui permettrait l'ajout d'interrogations ainsi que des accès à des systèmes d'information supplémentaires. Cela est déjà en partie modélisé. Il ne reste plus qu'à l'implémenter.

Bien que notre prototype ne soit pas encore complèment opérationel, il apparaît comme un outil permettant de faire la synthèse de nos travaux sur les systèmes d'aide à la décision que nous considérons. Nous avons mis en œuvre les modèles de systèmes multi-agents distribués et anytime dans une application de gestion de marché car elle remplissait tous les critères nécessaires à la démonstration de nos travaux. Néanmoins, certains points de nos travaux n'ont pas pu être pleinement exploités au sein de cette application.

Pour une exploitation plus complète de nos travaux, il sera sans doute nécessaire de concevoir plusieurs types d'applications. De plus, pour pouvoir rendre beaucoup plus significative la démonstration faite au moyen de cette application, il serait sans doute

nécessaire d'avoir recours à une expertise du domaine d'application mis en œuvre.

Chapitre 11

Conclusion et perspectives

11.1 Bilan

Dans ce document, nous avons décrit nos travaux de thèse qui consistent en un modèle de conception pour les systèmes d'aide à la décision. Ce modèle repose sur l'utilisation du paradigme multi-agent. Dans ce contexte, nous avons proposé deux principales contributions à l'étude des systèmes multi-agents :

- la prise en compte du temps réel dans les SMA au moyen de l'utilisation des techniques anytime, concrétisée par le modèle ANYMAS
- et la distribution physique des SMA au moyen de l'utilisation d'objets CORBA, concrétisée par le modèle DISMAS.

Ces deux modèles utilisés conjointement permettent de concevoir des systèmes d'aide à la décision basés sur des systèmes multi-agents anytime et distribués. Les modèles ainsi obtenus sont suffisamment génériques pour pouvoir être utilisés dans une classe d'applications assez large. Néanmoins, ils peuvent également être utilisés sur des applications particulières, en les utilisant de façon restreinte. Le modèle ANYMAS, par exemple, peut être employé sans la prise en compte des agents de coordination temporelle lorsque la complexité de l'application ne le justifie pas.

Les deux modèles présentés dans ce document (ANYMAS et DISMAS) ont été implémenté sous forme de packages JAVA et peuvent être utilisés comme des extensions de la plate-forme multi-agents MadKit. Cependant, il est tout à fait envisageable, moyennant de légères modifications, d'implémenter de façon différente ces modèles afin d'utiliser des outils ou langages de programmation autres que ceux nous avons utilisé.

Pour mettre en œuvre nos travaux sur les systèmes d'aide à la décision, nous avons choisi d'étudier une application de gestion de marché (voir chapitre 9). Cette application repose sur une utilisation partielle du modèle ANYMAS et l'utilisation du modèle DISMAS complet. Cette application nous a permis de mettre en évidence la simplicité d'utilisation des modèles ANYMAS et DISMAS, ainsi que la validité de nos travaux.

L'applicabilité de nos travaux de recherche étaient un des aspects essentiels que nous souhaitions mettre en évidence dans cette thèse. Elle permet en effet de montrer notre contribution dans le domaine de la conception de systèmes d'aide à la décision fonctionnant en environnement distribué et temps réel.

11.2 Perspectives de recherche

Ces travaux sont à la base de la conception de modèles très génériques pour la réalisation de systèmes d'aide à la décision complexes, comme l'application de gestion de marchés. Cependant, ils peuvent également être utilisés dans la réalisation d'applications plus complexes que celle que nous avons utilisée pour illustrer nos travaux. Un projet de conception d'un système d'aide à la décision relatif à la surveillance de sites industriels à hauts risques est actuellement à l'étude dans notre laboratoire. Dans ce type d'applications, le respect des contraintes temporelles est en effet très strict.

Dans nos travaux concernant la distribution physique des SMA, nous n'avons pas tenu compte de la répartition dynamique de charge et de la mobilité des agents. Une des extensions à court terme de nos travaux concerne la prise en compte de la mobilité des agents lors de la distribution des SMA. Une autre extension que nous envisageons est la prise en compte de l'interopérabilité suivant les travaux de l'OMG [OMG, 1997], c'est-à-dire le fonctionnement du système sur des plates-formes hétérogènes.

Un apport envisageable mais qui nécessiterait une étude approfondie est la mise en œuvre automatique des algorithmes anytime au sein des agents du modèle ANYMAS. En effet, à l'heure actuelle pour concevoir des agents ayant un comportement anytime, il est nécessaire de modéliser le comportement de chaque agent suivant les techniques anytime. Une phase ultime serait de mettre en place un modèle générique d'agent qui permette à chaque agent héritant d'un agent anytime d'être lui aussi anytime. La faisabilité de cette approche reste à étudier.

L'interrogation de systèmes d'information par des systèmes multi-agents a été abordée au cours de nos travaux. On a eu recours à des agents capables d'effectuer des requêtes sur des bases de données. Cet aspect de nos travaux pourrait être développé pour traiter des problèmes de gestion dynamique des sources d'information. En effet, que se passe-t-il lorsqu'un système d'information n'est plus disponible ou alors lorsqu'au contraire on veut prendre en compte un nouveau. Comment le SMA peut-il s'adapter à ces changements de situation ? C'est-à-dire est-il en mesure de détecter la disparition de systèmes d'information et d'aller chercher les informations dans un autre ? Lorsqu'on rend disponible un nouveau système d'information, que doit-on faire ? Des pistes pour l'exploitation de ces nouveaux systèmes d'information commencent à être exploitées telles que la mise en place de systèmes d'information descriptifs qu'il faut mettre à jour lors de l'adjonction

de systèmes d'information à l'application. Ceci nécessite de mettre en place des tables d'équivalence et des ontologies pour la description des informations contenues dans les systèmes d'information.

Un autre aspect de couplage SMA/Base de Données concerne la forme des interrogations. Dans nos travaux, les requêtes étaient définies lors de la conception de l'application. Une plus grande souplesse dans les interrogations nécessiterait de mettre en place un méta-langage permettant aux agents de générer des requêtes en fonction des systèmes d'information qui sont à leur disposition et en fonction du temps qui leur est alloué pour effectuer ces mêmes interrogations.

D'autres perspectives concernent l'implémentation. Il serait intéressant de mettre en œuvre nos travaux sur d'autres plates-formes de développement de SMA et d'utiliser d'autres outils, notamment pour l'aspect distribué, comme par exemple l'utilisation d'un autre ORB.

Annexe A

MadKit : une plate-forme de développement de systèmes multi-agents

MadKit est une plate-forme de développement ultra-légère qui permet de faire du développement de SMA de manière simple et rapide. Le langage de programmation utilisé est JAVA. Ce sont ces caractéristiques qui nous ont amené pour nos développements à utiliser MadKit et à l'étendre lorsque cela a été nécessaire. Dans cette annexe, nous allons présenter cette plate-forme afin de donner des éclaircissiments sur les différents développements que nous avons effectués au cours de nos travaux thèse.

A.1 Présentation générale et cadre de réalisation

MadKit (pour MultiAgent Development Kit) est une plate-forme générique de conception et d'exécution de systèmes multi-agents [Gutknecht and Ferber, 1997]. Elle a été développée par Olivier Gutknecht au sein du Laboratoire LIRMN. L'ensemble du développement a été fait en java. Cette plate-forme s'appuie pour la réalisation de SMA sur le modèle conceptuel Aalaadin [Gutknecht and Ferber, 1999] que nous allons présenter dans la suite. L'originalité de cette plate-forme est qu'elle est basée sur un modèle organisationnel plutôt que sur une architecture d'agent ou un modèle d'interaction spécifique.

Cette plate-forme fournit un ensemble de classes java permettant la réalisation d'agents par héritage de classes. Il est aussi possible d'utiliser une interface graphique, nommée G-Box, qui permet de visualiser les agents sous forme de composants graphiques sur lesquels il est possible d'agir.

Dans cette section, nous allons présenter le cadre de réalisation de cette plate-forme, les concepts principaux et son architecture générale.

A.1.1 Cadre de réalisation

Le modèle conceptuel Aalaadin a été mis en œuvre par Olivier Gutknecht dont nous allons présenter les travaux de Olivier Gutknecht.

A.1.1.1 Les travaux de Olivier Gutknecht

Les travaux de Olivier Gutknecht concernent l'étude et la réalisation d'un méta-modèle organisationnel réflexif pour l'analyse et la conception des systèmes multi-agents. Ces travaux ont abouti principalement à la conception d'un méta-modèle, Aalaadin, et à la réalisation d'une plate-forme générique de développement de SMA, MadKit.

A.1.1.2 L'implémentation d'un modèle conceptuel

La réalisation de la plate-forme MadKit a été fortement motivée par le désir de mettre en œuvre un nouveau modèle conceptuel pour la réalisation de SMA. L'ensemble des SMA qui seront alors réalisés à partir de la plate-forme MadKit seront basés sur l'utilisation de ce modèle conceptuel.

A.1.2 Les concepts principaux

Trois concepts principaux ont été introduits dans la réalisation de la plate-forme MadKit :

1. la présence d'un micro-noyau,
2. une agentification des services,
3. une interface componentielle.

Nous allons présenter en détail chacun de ses concepts.

A.1.2.1 Un micro-noyau agent

Il s'agit d'un moteur d'exécution d'agents qui a été réduit au minimum. Ce micro-noyau est encapsulé dans un agent. Il assure un certain nombre de tâches :

- Il mémorise et gère les groupes et les rôles.
- Il se charge de lancer et de contrôler l'exécution concurente des agents locaux. Il possède donc la possibilité de contrôler le cycle de vie des agents.
- Il permet l'échange de messages entre les agents. Pour les échanges de messages, l'émetteur et le récepteur sont identifiés au moyen d'un objet « AgentAdress » grâce auquel le noyau peut rediriger et redistribuer les messages.

A.1.2.2 Une agentification des services

Dans MadKit, un service est agentifié dans la mesure où on y accède selon une interaction agent/agent qui correspond à un rôle. Le fonction du système est alors décrite en termes de groupes: groupe « System », groupe « Mobility », etc.

Les services généraux sont décrits en agents/groupe/rôle. Il s'agit par exemple des fonctions de communication à distance.

A.1.2.3 Une interface componentielle

Le modèle graphique est basé sur des composants indépendants à la « ActiveX/OpenDoc ». En effet, chaque agent est seul responsable de son interface graphique qui est représentée au moyen d'un « Java Bean ». Des fonctionnalités incluses dans l'agent permettent aux développeurs de SMA basés sur MadKit de concevoir des interfaces graphiques sous forme de « Java Bean » pour leurs agents. Ces interfaces graphiques sont notamment utilisées pour la représentation des agents dans la « G-Box » que nous allons présenter plus loin.

A.1.3 L'architecture générale

Pour résumer, l'architecture générale se présente sous la forme suivante (voir figure A.1):

- un micro-noyau qui sert de base à l'exécution des agents,
- les agents,
- les « beans », interface graphique des agents,
- une gestion de groupes et de rôles,
- un environnement graphique: la G-Box.

Cette architecture générale peut être réduite pour le développement de SMA basés sur MadKit. On peut très bien se passer de l'environnement de développement graphique (la G-Box) et de la partie graphique des agents (Bean).

A.2 Le modèle conceptuel Aalaadin

Aalaadin [Gutknecht et al., 2000] est un modèle organisationel qui repose sur les notions d'agent, de groupe et de rôle. Le lien entre ces trois composants de base est représenté dans la figure A.2.

Nous allons maintenant nous attacher à décrire les trois composants de base de ce modèle organisationel.

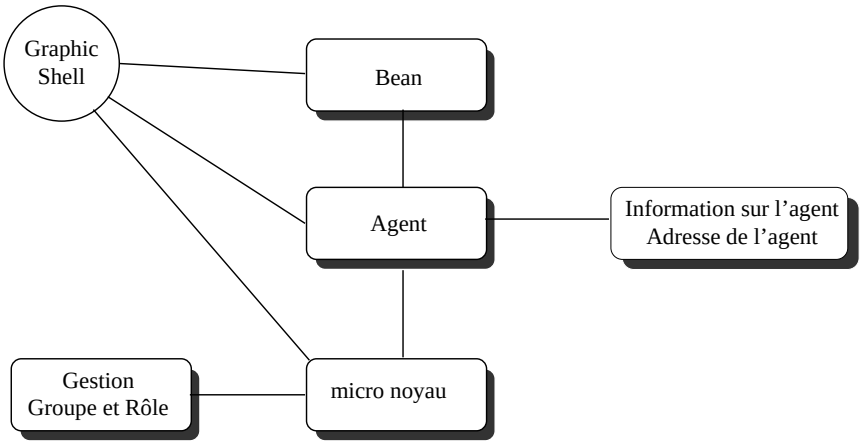


FIG. A.1 – *Architecture générale de MadKit*

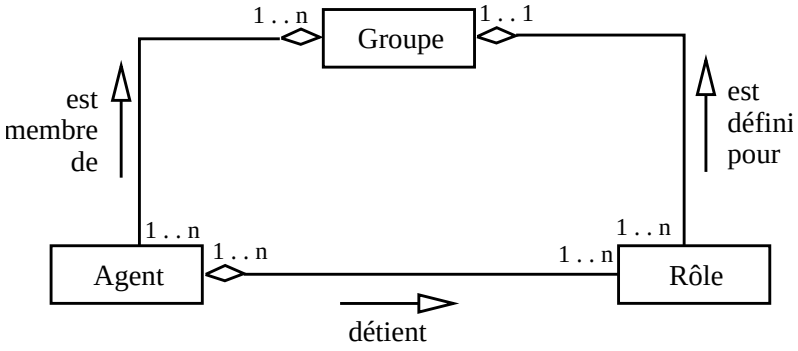


FIG. A.2 – *Le modèle organisationel*

A.2.1 L'agent

Quasiment aucune contrainte n'est imposée sur l'architecture interne ou sur le modèle de comportement de l'agent. L'agent est simplement décrit comme une entité autonome communiquante qui joue des rôles au sein de différents groupes. La très faible sémantique associée dans ce modèle est tout à fait volontaire. Le but est de laisser toute liberté au concepteur pour l'architecture interne appropriée à son domaine applicatif. Cette liberté se retrouve dans la plate-forme MadKit.

A.2.2 Le groupe

Le groupe est une notion primitive de regroupement d'agents. Chaque agent peut être membre d'un ou plusieurs groupes. D'une façon un peu simpliste, un groupe peut être vu comme un moyen d'identifier par regroupement un ensemble d'agents. Plus classiquement, associé au rôle, il définira la structure organisationnelle d'un SMA usuel. Les différents groupes peuvent se recouper librement.

A.2.3 Le rôle

Le rôle est une repésentation abstraite d'une fonction, d'un service ou d'une identification d'un agent au sein d'un groupe particulier. Chaque agent peut avoir plusieurs rôles. Un même rôle peut être tenu par plusieurs agents et les rôles sont locaux aux groupes. L'hétérogénéité des situations d'interaction est rendue possible par le fait qu'un agent peut avoir plusieurs rôles distincts au sein de plusieurs groupes, les communications étant cloturées par les groupes.

A.3 Un environnement graphique : la G-Box

La G-Box (voir figure A.3) constitue un environnement graphique de développement et de test pour les systèmes multi-agents conçus au moyen de MadKit. Elle permet de contrôler le cycle de vie des agents, de charger dynamiquement des SMA, et de manipuler graphiquement les accointances ou bien les tables de groupes et de rôles.

Les principaux composants de la G-Box sont :

- La « ToolBox » : elle montre la liste des agents instanciables depuis l'environnement graphique de la G-Box. Cette liste peut-être enrichie en ajoutant de nouveau « packs d'agents » au moyen de commandes (« add agent ... » ou « load jar ... ») .
- La zone de définition de propriétés : elle permet d'avoir accès aux propriétés éditables des agents.

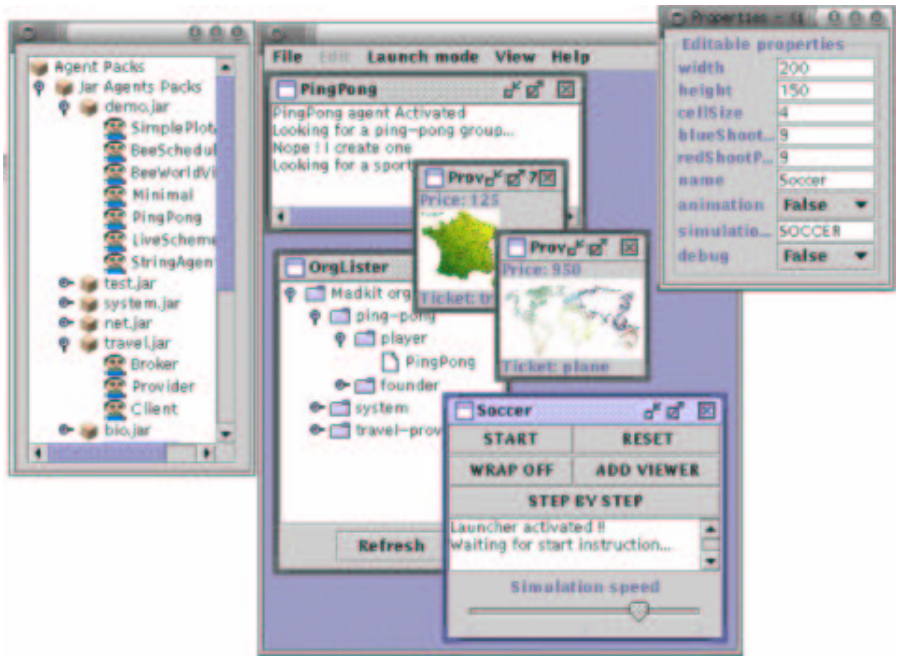


FIG. A.3 – *La G-Box*

– Le « bureau des agents » : il s'agit de l'emplacement où sont localisées les parties graphiques des agents. Pour instancier un agent, il suffit de double-clicker dessus dans la « Toolbox » et il apparaît alors sur le bureau.

A.4 L'architecture générique d'agent dans MadKit

La classe de base d'un agent MadKit (`AbstractAgent`) définit quelques fonctionnalités de base pouvant être nécessaires dans les modèles classiques.

A.4.1 Fonctionnaliés

Les fonctions associées à chaque agent sont :

– **Cycle de vie** : l'agent dispose de quatre cycles (création, activation, exécution, et destruction), et a la possibilité de démarrer d'autres agents sur le noyau local (et les désactiver par la suite). Par contre, aucun mécanisme concrêt n'est défini à ce niveau.

– **Communication** : la communication est implémentée sous forme déchange de messages asynchrone, soit d'un agent à agent identifié par leur `AgentAddress` ou leur groupe et rôle, soit sous la forme d'une diffusion à tous les teneurs d'un rôle dans un groupe donné.

- **Organisation** : tout agent dispose de primitives permettant d'observer son organisation locale (connaître les groupes et rôles courants) et d'agir dessus (prise de rôle, entrée et retrait de groupes).
- **Outils** : la classe de base des agents permet également de manipuler une éventuelle interface graphique associée à l'agent, les flots d'entrée/sortie, etc.

A.4.2 Messages

Les messages sont définis par héritage à partir d'une classe de base (`Message`) qui ne définit que la notion d'émetteur et de destinataire. Certains messages de base sont néanmoins fournis (encapsulation d'une chaîne ou d'un objet arbitraire, messages KQML et FIPA-ACL, messages XML) mais restent extensibles pour s'adapter à tout protocole d'interaction.

A.5 Conclusion

Dans cette annexe, nous avons donné une brève description de la plate-forme MadKit en nous basant sur nos expériences d'utilisation, le site web `www.madkit.org` et les articles de Gutknecht [Gutknecht and Ferber, 1997], [Gutknecht and Ferber, 1999] et [Gutknecht et al., 2000].

Notre choix de la plate-forme MadKit avait été guidé par les propriétés suivantes :

- Son langage de développement (Java) : ce langage dispose de tous les outils nécessaires au développement de SMA.
- Sa flexibilité : il nous a été facile d'enrichir cette plate-forme dans le cadre de nos travaux.
- Sa légèreté : l'absence de toute fonction superflue qui en facilite son utilisation pour développer des systèmes multi-agents.

Bibliographie

[AFIA, 1994] AFIA (1994). REAKT : Article de présentation du projet. *Journal de l'AFIA*, (16).

[Alur and Henziger, 1990] Alur, R. and Henziger, T. (1990). Real-time logics : complexity and expressiveness. In *Proceedings of the* 5^{th} *Symposium on Logics Computer Science*, Philadelphia.

[ARPA, 1993] ARPA (1993). Specification of the KQML agent-communication. Technical report, Knowledge Sharing Initiative, External Interfaces Working Group working paper.

[Axelrod, 1997] Axelrod, R. (1997). *The Complexity of Cooperation : Agent-Based Models of Competition and Collaboration.* Princeton University Press.

[Azvine et al., 2000] Azvine, B., Azarmi, N., and Nauck, D. (2000). *Intelligent Systems and Soft Computing : Prospects, Tools and Applications*, volume 1804 of *LNCS*. Springer-Verlag.

[Benchimol et al., 1990] Benchimol, G., Lévine, P., and Pomerol, J.-C. (1990). *Systèmes experts dans l'entreprise.* Editions Hermès.

[Berry and Cosserat, 1984] Berry, G. and Cosserat, L. (1984). The ESTEREL synchronous programming language and its mathematical semantics. volume 197 of *LNCS*. Springer-Verlag.

[Bertelle et al., 2000] Bertelle, C., Olivier, D., Tranouez, P., and Jay, V. (2000). Agent-based simulation of water flow for environment modelling in estuaries. In *Agent-Based Simulation*, Passau, Germany.

[Berthomieu and Diaz, 1991] Berthomieu, B. and Diaz, M. (1991). Modeling and verification of time dependant systems using time petri nets. *IEEE T.S.E.*, 17(3) :259–273.

[Booth, 1997] Booth, G. (1997). Gecko : A continuous 2D world for ecological modeling. *Artificial Life*, 3(3) :147–163.

[Boukachour et al., 1998] Boukachour, H., Allorge, S., Lesage, F., and Cardon, A. (1998). Information and communication systems : evolving toward interoperability. In *Proceedings of ISD'98.*

[Boukachour et al., 2000] Boukachour, H., Duvallet, C., and Cardon, A. (2000). Multiagent system to prevent technological risks. In *Proceedings of ACIDCA'2000.*

[Brunessaux et al., 1992] Brunessaux, S., Charpillet, F., Haton, J.-P., and Mentec, J.-C. L. (1992). Une architecture à bases de connaissances multiples et orientée temps réel. *Génie Logiciel et Systems Experts*, 28.

[Bucci et al., 1995] Bucci, G., Campanai, M., and Nesi, P. (1995). Tools for specifying real-time systems. *Journal of Real-Time Systems*, 8 :117–172.

[Burkhart, 1994] Burkhart, R. (1994). The SWARM multi-agent simulation system. In *Object Oriented Programming Systems, Languages and Applications (OOPSLA)'94.*

[Burns and Wellings, 1994] Burns, A. and Wellings, A. (1994). HRT-HOOD : A structured design method for hard real-time systems. *Journal of Real-Time Systems*, 6 :73–114.

[Cardon, 1996] Cardon, A. (1996). La complexité des systèmes d'expression du sens. Technical report, IBP.

[Cardon, 1997] Cardon, A. (1997). A multi-agent model for co-operative communications in crisis management system : The act of communication. In *The* 7^{th} *European-Japanese Conference in Information Modelling and Knowledge Bases*, pages 111–123, Toulouse.

[Cardon, 1998a] Cardon, A. (1998a). *Les systèmes adaptatifs à architecture d'agents dynamiques : une approche de la conscience artificielle.* LIP6.

[Cardon, 1998b] Cardon, A. (1998b). *Modélisation des Systèmes Adaptatifs par Agents : vers une Analyse-conception Orientée Agent.* LIP6.

[Cardon, 2000] Cardon, A. (2000). *La conscience artificielle.* Editions Eyrolles.

[Cardon and Durand, 1997] Cardon, A. and Durand, S. (1997). A model of crisis management system including mental representations. In *Proceedings of the AAAI Spring Symposium.*

[Cardon and Lesage, 1997] Cardon, A. and Lesage, F. (1997). An interpretation process on communication between actors in a distributed system for crisis management. In *Symposium of Informatic Economics*, Kichinev.

[Cardon and Lesage, 1998] Cardon, A. and Lesage, F. (1998). Toward adaptative information systems : considering concern and intentionality. In *Proceedings of KAW'98.*

[Castillo et al., 1998] Castillo, A., Kawaguchi, M., Paciorek, N., and Wong, D. (1998). Concordiatm as enabling technology for cooperative information gathering. In *Japanese Society for Artificial Intelligence Conference*, Tokyo.

[Chaib-draa, 1999] Chaib-draa, B. (1999). *Agents et systèmes multi-agent.* Université de Laval, Québec. Notes de cours.

[Cohen and Levesque, 1995] Cohen, P. R. and Levesque, H. J. (1995). Communicative actions for artificial agents. In *ICMAS'95*, pages 65–72. AAAI Press.

[Cohen, 1995] Cohen, V. (1995). *La recherche opérationnelle*, volume 941 of *Que sais-je ?* Presses Universitaires de France.

[Collinot et al., 1996] Collinot, A., Ploix, L., and Drogoul, A. (1996). Application de la méthode cassiopee à l'organisation d'une équipe de robots. In *JFIADSMA'96*, pages 137–152. Editions hermès.

[Collis and Lee, 1998] Collis, J. and Lee, L. (1998). Building electronic marketplaces with ZEUS agent tool-kit. In Pablo Noriega, C. S., editor, *The* 1^{st} *International Workshop on Agent Mediated Electronic Trading(AMET'98)*, LNCS, pages 1–24, Minneapolis, MN, USA. Springer-Verlag.

[Courbon et al., 1994] Courbon, J.-C., Dubois, D., Roy, B., and Pomerol, J.-C. (1994). Autour de l'aide à la décision et l'intelligence artificielle. Rapport de recherche, IBP-Laforia.

[Courtiat et al., 1993] Courtiat, J., Camargo, M. D., and Saîdouni, D. (1993). Rt-lotos : Lotos temporisé pour la spécification de systèmes temps réel. In Hermès, E., editor, *Actes CFIP'93*, Montréal.

[Daniels, 1999] Daniels, M. (1999). Integrating simulation technologies with swarm. In *Agent Simulation : Applications, Models, and Tools*, University of Chicago.

[Daniels, 2000] Daniels, M. (2000). An open framework for agent-based modeling. In *Applications of Multi-Agent Systems in Defense Analysis*, Los Alamos National Labs.

[Dean and Boddy, 1988] Dean, T. and Boddy, M. (1988). Solving time-dependant planning problems. In *Proceedings of the* 7^{th} *National Conference of Artificial Intelligence*, pages 419–454, Minneapolis, Minnesota.

[Deangelis and Gross, 1992] Deangelis, D. and Gross, L. (1992). *Individual-Based Models et Approaches in Ecology : Populations, Communities et Ecosystems.* Chapman et Hall.

[Decker et al., 1992a] Decker, K., Garvey, A., Humphrey, M., and Lesser, V. (1992a). Effects effects on parallelism on blackboard systems scheduling. In *Proceedings of the* 12^{th} *IJCAI*, pages 15–21, Sidney, Australia. Morgan Kauffman.

[Decker et al., 1992b] Decker, K., Garvey, A., and Lesser, V. (1992b). A blackboard system for real-time control of approximate processing. In *Proceeding of the Hawai International Conference on systems science, BlackBoard Sytems*, pages 1–10.

[DeLespinasse et al., 1995] DeLespinasse, A. J. A., Tauber, J., Gifford, D., and Kaashoek, M. (1995). Rover : A toolkit for mobile information access. In *Proceedings of the* 15^{th} *ACM Symposium on Operating Systems Principles*, pages 156–171, Copper Mountain, Co.

[Desard et al., 1991] Desard, F., Brunessaux, S., Bertuzzi, S., Joubert, T., Mentecand, J.-C. L., Lalanda, P., Charpillet, F., and Haton, J.-P. (1991). Mécanismes temps réel

dans un système multibases de connaissances. *Revue Scientifique et Technique de la Défense*, pages 61–68.

[Dodhiawala et al., 1989] Dodhiawala, R., Sridharan, N., Raulefs, P., and Pickering, C. (1989). Real-time ai systems : A definition and an architecture. In *Proceeding of the* 11^{th} *IJCAI*, pages 256–264, Detroit, MI, USA. Morgan Kauffman.

[Downing and Zvirinsky, 1999] Downing, K. and Zvirinsky, P. (1999). The simulated evolution of biochemical guilds : Reconciling Gaia theory and natural selection. *Artificial Life*, 5(4).

[Drogoul, 1993] Drogoul, A. (1993). *De la simulation multi-agents à la résoltion collective de problèmes : une étude de l'émmergence de structure d'organisation dans les systèmes multi-agents.* PhD thesis, Université de Paris VI.

[Durand, 1999] Durand, S. (1999). *Représentation des points de vues multiples dans une situation d'urgence : une modélisation par organisations d'agents.* PhD thesis, Université du Havre.

[Duvallet et al., 1999] Duvallet, C., Mammeri, Z., and Sadeg, B. (1999). Les SGBD temps réel. *Technique et Science Informatique*, 18(5) :479–516.

[Emerson and Halpern, 1986] Emerson, E. and Halpern, J. (1986). Sometimes and not never revisited : on branching versus linear time temporal logic. *Journal of ACM*, 33(1).

[Eriksson et al., 1998] Eriksson, J., Finne, N., and Janson, S. (1998). SICS marketplace - an agent based market infrastructure. In Pablo Noriega, C. S., editor, *The* 1^{st} *International Workshop on Agent Mediated Electronic Trading(AMET'98)*, LNCS, pages 41–53, Minneapolis, MN, USA. Springer-Verlag.

[Fagin et al., 1995] Fagin, R., Halpern, I., Moses, Y., and Vardi, M. (1995). *Reasonning about knowledge.* MIT Press : cambridge.

[Farrell and Arthur, 1998] Farrell, W. and Arthur, B. (1998). *How Hits Happen–Forecasting Predictability in a Chaotic Marketplace.*

[Ferber, 1995] Ferber, J. (1995). *Les systèmes multi-agents.* InterEditions, Paris.

[Finin et al., 1994] Finin, T., Fritzson, R., McKay, D., and McEntire, R. (1994). KQML, as an agent communication language. In *CIKM'94*. ACM Press.

[Firby, 1987] Firby, R. (1987). An investigation into reactive planning in complex domains. In *Proceedings of AAAI*, page 202, Seattle, Washington.

[Fox, 1981] Fox, M. (1981). Factory modelling, simulation, and scheduling in the intelligent management system. In *Proceedings of the* 7^{th} *International Joint Conference on Artificial Intelligence.*

[Fukuda, 1997] Fukuda, M. (1997). *MESSENGERS : A Distributed Computing System Based on Autonomous Objects.* PhD thesis, University of Tsukuba, Japan.

[Fukuda et al., 1999] Fukuda, M., Bic, L., Dillencourt, M., and Cahill, J. (1999). Messages versus messengers in distributed programming. *Journal of Parallel and Distributed Computing*, 57 :188–211.

[Geib et al., 1997] Geib, J.-M., Gransart, C., and Merle, P. (1997). *Corba : des concepts à la pratique*. InterEditions.

[Ghezzi et al., 1990] Ghezzi, C., Mandriolli, D., and Marzenti, A. (1990). Trio, a logical language for executable specifications of real-time systems. *Journal of System Software*, 12 :107–123.

[Ghidini and Serafini, 1998] Ghidini, C. and Serafini, L. (1998). Information integration for electronic commerce. In Pablo Noriega, C. S., editor, 1^{st} *International Workshop on Agent Mediated Electronic Trading(AMET'98)*, LNCS, Minneapolis, MN, USA. Springer-Verlag.

[Gomaa, 1986] Gomaa, H. (1986). Software development for real-time systems. *Communication of ACM*, 29(7) :657–668.

[Guessoum, 1996] Guessoum, Z. (1996). *Un Environnement Opérationnel de Conception et de Réalisation de Systèmes Multi-Agents*. PhD thesis, Laforia, Paris 6.

[Guessoum and Briot, 1997] Guessoum, Z. and Briot, J.-P. (1997). From concurrent objects to autonomous agents. In *The* 8^{th} *European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, Ronneby.

[Gutknecht and Ferber, 1997] Gutknecht, O. and Ferber, J. (1997). Madkit : Organizing heterogeneity with groups in a platform for multiple multi-agent systems. Technical report, LIRMM, UMR 9928, Université de Montpellier II.

[Gutknecht and Ferber, 1999] Gutknecht, O. and Ferber, J. (1999). Vers une méthodologie organisationnelle de conception de systèmes multi-agents. Technical report, LIRMM, UMR 9928, Université de Montpellier II.

[Gutknecht et al., 2000] Gutknecht, O., Ferber, J., and Michel, F. (2000). Madkit : une architecture de plate-forme multi-agent générique. Technical report, LIRMM, UMR 9928, Université de Montpellier II.

[Guttman and Maes, 1998] Guttman, R. and Maes, P. (1998). Agent-mediated integrative negotiation for retail electronic commerce. In Pablo Noriega, C. S., editor, *The* 1^{st} *International Workshop on Agent Mediated Electronic Trading(AMET'98)*, LNCS, pages 70–90, Minneapolis, MN, USA. Springer-Verlag.

[Halpern, 1995] Halpern, J. (1995). *Reasoning about knowledge : a survey*, pages 1–34. University Press, Oxford England.

[Harel et al., 1987] Harel, D., Pneuli, A., Schmidt, J., and Shermann, R. (1987). On the formal semantics of statecharts. In *Proceedings of the* 2^{nd} *IEEE Symposium on Logic Computer Science*, pages 54–64, New York.

[Haton et al., 1991] Haton, J.-P., Bouzid, N., Charpillet, F., Haton, M.-C., Laasri, B., Laasri, H., Marquis, P., Mondot, T., and Napoli, A. (1991). *Le raisonnement en intelligence artificielle*. InterEditions, Paris.

[Hayes-Roth, 1985] Hayes-Roth, B. (1985). A blackboard architecture for control. *Artificial Intelligence*, pages 251–321.

[Hayes-Roth et al., 1989a] Hayes-Roth, B., Hewett, R., Washington, R., Hewett, M., and A.Seiver (1989a). *Distributin intelligence within a single individual*, volume 2 of *Distributed Artificial Intelligence*. Morgan Kauffman.

[Hayes-Roth et al., 1989b] Hayes-Roth, B., Washington, R., Hewett, R., Hewett, M., and Seiver, A. (1989b). Intelligent real-time monitoring and control. In *Proceedings of the* 11^{th} *IJCAI*, pages 243–249, Detroit, Michigan.

[Horvitz, 1987] Horvitz, E. (1987). Reasonning about beliefs and actions under computational resource constraints. In *Proceedings of the Workshop on Uncertainty in Artificial Intelligence*, Seattle, Washington.

[Huget and Pinson, 1999] Huget, M.-P. and Pinson, S. (1999). Une typologie des modèles d'agents. Les cahiers du lamsade, Université Paris Dauphine.

[Ingrand and Coutance, 1993] Ingrand, F. and Coutance, V. (1993). Prs : Real time reasoning using procedural reasoning. Technical report, LAAS/CNRS, Toulouse.

[Jennings et al., 1998] Jennings, N., Sycara, K., and Wooldridge, M. (1998). A roadmap of agent research and development. *Autonomous Agent and Multi-Agent Systems*, 1(5) :7–38.

[Johanian and Mok, 1986] Johanian, F. and Mok, A. (1986). Safety analysis of timing properties in real-time systems. *IEEE T.S.E.*, 12(3) :890–904.

[Kersten and Mallory, 1990] Kersten, G. and Mallory, G. R. (1990). Supporting problem representation in decisions with strategic interactions. *European Journal of Operational Research*, 46 :200–215.

[Kohler and Carr, 1996] Kohler, T. and Carr, E. (1996). Swarm-based modeling of prehistoric settlement systems in southwestern north america. In *Proceedings of Colloquium II, UISPP,* 13^{th} *CongressSeptember*, volume 5 of *Sydney University Archaeological Methods*, I. Johnson and M. North. School of Archaeology, University of Sydney, Australia.

[Koymans, 1989] Koymans, R. (1989). *Specifying passing message and time critical systems with temporal logic*. PhD thesis, Eindhoven University, The Netherlands.

[Laasri and Laasri, 1988] Laasri, H. and Laasri, B. (1988). Atome : a blackboard architecture with temporal and hypothetical reasonning. In *Proceedings of th* 8^{th} *ECAI*, pages 1–5, Munich.

[Lai, 1997] Lai, H. (1997). *UML, La Notation Unifiée de la Modélisation Objet*. InterEditions.

[Lalanda et al., 1992] Lalanda, P., Charpillet, F., and Haton, J. (1992). Une architecture temps réel à base de tableau noir. In *Proceedings of the* 12^{th} *Avignon Conference on Expert Systems*, volume 1, pages 671–682.

[Lange and Oshima, 1998] Lange, D. and Oshima, M. (1998). *Programming and Deploying Mobile Agents with Java Aglets*. Addison-Wesley.

[Lementec and Brunessaux, 1992] Lementec, J. and Brunessaux, S. (1992). Atome-tr, a real-time control for BlackBoard scheduling. In *Proceedings ECAI*, pages 255–256.

[Lesage, 2000] Lesage, F. (2000). *Interprétation adaptative du discours dans une situation multiparticipants : modélisation par agents*. PhD thesis, Université du Havre.

[Lesser and Corkill, 1983] Lesser, V. and Corkill, D. (1983). The dvmt : a tool for investigating distributed problem solving networks. *Artificiel Intelligence*, 4(3) :15–33.

[Lesser et al., 1988] Lesser, V., Pavlin, J., and Durfee, E. (1988). Approximating processing in real-time problem solving. *Artificiel Intelligence*, 9(1) :49–61.

[Lévine and Pomerol, 1989] Lévine, P. and Pomerol, J.-C. (1989). *Systèmes interactifs d'aide à la décision et systèmes experts*. Editions Hermes.

[Luna and Stefannson, 2000] Luna, F. and Stefannson, B. (2000). *Economic Simulations in Swarm : Agent-Based Modelling and Object Oriented Programming*, volume 14 of *Advances in Computational Economics*. Kluwer Academic Publisher.

[MADKIT, 2001] MADKIT (2001). *http ://www.madkit.org*.

[Mammeri, 1998] Mammeri, Z. (1998). Expression et dérivation des contraintes temporelles dans les applicationtemps réel. *APII-JESA*, 32(5-6) :609–644.

[Molloy, 1985] Molloy, K. (1985). Discrete time stochastic Petri Nets. *IEEE T.S.E.*, 11(4) :417–423.

[Morzenti and Pietro, 1994] Morzenti, A. and Pietro, P. (1994). Object-oriented logical specification of time critical systems. *ACM T.O.S.E.M*, 3(1) :56–98.

[Mouaddib, 1993] Mouaddib, A. (1993). *Contribution au raissonnement progressif temps réel dans un univers multi-agents*. PhD thesis, Université de Nancy I.

[Mouaddib, 1997] Mouaddib, A. (1997). Progressive negotiation for time-constrained autonomous agents. In *Proceedings of the* 1^{st} *International Conference on Autonomous Agents*, pages 8–15, Marina del Rey, CA, USA. ACM Press, New York.

[Mouaddib et al., 1992] Mouaddib, A., Charpillet, F., and Haton, J. (1992). Raisonnement temps-réel dans un système multi-agents. In *Actes PRC-GDR Intelligence Artificielle*, pages 597–607.

[Mouaddib et al., 1993] Mouaddib, A., Charpillet, F., and Haton, J. (1993). Real-time engine of messages for multi-agents architecture. In *Actes 13èmes Journées sur les Systèmes Experts et leurs Applications, Avignon*, pages 589–599.

[Mouaddib and Zilberstein, 1997] Mouaddib, A. and Zilberstein, S. (1997). Handling duration uncertainty in meta-level control of progressive processing. In *Proceedings of the* 5^{th} *International Joint Conference on Artificial Intelligence, IJCAI 97*, pages 1201–1207, Nagoya, Japan. Morgan Kaufmann.

[Mouaddib and Zilberstein, 1998] Mouaddib, A. and Zilberstein, S. (1998). Optimal scheduling of dynamic progressive processing. In *Proceedings of the* 13^{th} *European Conference on Artificial Intelligence*, pages 499–503, Brighton, UK. John Wiley and Sons, Chichester.

[Muller, 1997] Muller, P. (1997). *Modélisation objet avec UML.* Editions Eyrolles.

[Newell and Simon, 1972] Newell, A. and Simon, H. (1972). *Human Problem Solving.* Prentice Hall, Englewood Cliffs.

[Occello and Demazeau, 1994] Occello, M. and Demazeau, Y. (1994). Building real time agent using parallel blackboards and its use for mobile robotics. In *IEEE International Conference on Systems, Man and Cybernetics*, San Antonio.

[Olivier et al., 2000] Olivier, D., Jay, V., and Bertelle, C. (2000). Distributed multi-agent system used for dynamic aquatic simulation. In *Proceedings of the ESS'2000 Congress*, Hambourg, Germany.

[OMG, 1995] OMG (1995). The common object request broker : Architecture and specification. In *OMG TC Document 95-7-X.*

[OMG, 1997] OMG (Octobre 1997). Mobile agent system interoperability facilities. In *OMG TC Document 97-10-05.*

[Ostroff, 1989] Ostroff, J. (1989). *Temporal logic for real-time systems*, volume 1 of *Advanced Sofware Development Series.* Research Studies Press, Taunton, Somerset, U.K.

[Ostroff and Wohnam, 1987] Ostroff, J. and Wohnam, W. (1987). Modeling and verifying real-time embedded computer systems. In *Proceedings of IEEE RTS Symposium*, pages 124–132.

[Pomerol, 1992] Pomerol, J.-C. (1992). Aide à la décision et IA. *L'Intelligence Artificielle : une discipline et un carrefour inter-disciplinaire.*

[Pomerol, 1996] Pomerol, J.-C. (1996). Artificial intelligence and human decision making. Rapport de recherche, IBP-Laforia.

[Pomerol, 1997] Pomerol, J.-C. (1997). Artificial intelligence and human decision making. *European Research of Operational Research*, 99 :3–25.

[Prietula et al., 1998] Prietula, M., Carley, K., and Gasser, L. (1998). *Simulating Organizations : Computational Models of Institutions et Groups.* The MIT Press Book.

[Ramamritham, 1996] Ramamritham, K. (1996). Where do time constraints come from and where do they go? *Journal of Database Management*, 7(2) :4–10.

[Razouk and Gorlick, 1989] Razouk, R. and Gorlick, M. (1989). A real-time interval logic for reasonning about execution of real-time program. In *ACM/SIGSOFT'89(TAV3)*.

[Salvant, 1998] Salvant, T. (1998). CAAM, un modèle d'agent anytime coopératif pour l'aide à la décision en temps critique. Thèse de doctorat, ENST.

[Salvant and Brunessaux, 1997] Salvant, T. and Brunessaux, S. (1997). Multi-agent based time-critical decision support for C31 systems. In *Practical Applications of Intelligent Agents and Multi-Agents, PAAM*, London, 1997.

[Schoppers, 1987] Schoppers, M. (1987). Universal plans for reactive robots in unpredictable environments. In *Proceedings of the* 9^{th} *IJCAI conference*, pages 1039–1046, Milan, Italy. Morgan Kauffman.

[Shoham, 1993] Shoham, Y. (1993). Agent oriented programming. *Journal of Artificial Intelligence*, 60 :51–92.

[Sprague, 1987] Sprague, R. (1987). DSS in context. *Decision Support System*, 3 :197–202.

[Stankovic and Ramamritham, 1988] Stankovic, J. and Ramamritham, K. (1988). *Hard real-time systems : a tutorial*. IEEE Computer Society Press.

[UIT, 1996] UIT (1996). SDL : Langage de description et de spécification. Union Internationale des Télécommunications, Recommendation UTI-TZ100.

[Vongkasem and Chaib-draa, 1999] Vongkasem, L. and Chaib-draa, B. (1999). ACL as a joint project between participants : A preliminary report. In *The* 1^{st} *workshop on Agent Communication Langage (ACL'99)*, Stockholm, Sweden.

[Voyager, 1997] Voyager (1997). *Voyager Core Technology User Guide, Version 2.0 Beta 1.*

[Ward, 1986] Ward, P. (1986). The transformation scheme : an extension of the dataflow diagram to represent control and timing. *IEEE T.S.E.*, 12(2) :198–210.

[Wicke et al., 1998] Wicke, C., Bic, L., Dillencourt, M., and Fukuda, M. (1998). Automatic state capture of self-migrating computations in messengers. In *Second International Workshop on Mobile Agent*. Springer Verlag.

[Wooldridge and Jennings, 1994] Wooldridge, M. and Jennings, N. (1994). Agent theories, architectures and language : A survey. In Wooldridge, M. and Jennings, N., editors, *Intelligent Agents, ECAI 1994*, volume LNAI 890, pages 1–32. Springer Verlag.

[Wooldridge and Jennings, 1995] Wooldridge, M. and Jennings, N. R. (1995). Intelligent agents : Theory and practice. *The Knowledge Engineering Review*, 10(2) :115–152.

[Zilberstein, 1996] Zilberstein, S. (1996). Resource-bounded sensing and planning in autonomous systems. *Autonomous Robots*, 3 :31–48.

[Zilberstein and Russell, 1993] Zilberstein, S. and Russell, S. (1993). Anytime sensing, planning and action : A practical model for robot control. In *The* 13^{th} *International Joint Conference on Artificial Intelligence*, Chambéry.

[Zilberstein and Russell, 1996] Zilberstein, S. and Russell, S. (1996). Optimal composition of real-time systems. *Artificial Intelligence*, 82(1-2) :181–213.

Publications

1. C. Duvallet, and B. Sadeg. Concevoir un système d'aide à la décision basé sur un système multi-agent anytime. In *Proceedings of JFIADSMA'2001*, Montréal, Canada, novembre 2001.
2. C. Duvallet, B. Sadeg, and A. Cardon. An anytime multiagent system to manage electronic commerce transactions. In S. Verlag, editor, *Proceedings of OOIS'2000*, décembre 2000.
3. C. Duvallet, H. Boukachour, and A. Cardon. Intelligent and self-adaptative interface. In S. Verlag, editor, *Proceedings of IEA/AIE'2000*, volume LNCS 1821, New Orleans, USA, juin 2000.
4. C. Duvallet and H. Boukachour. Modélisation par agents d'une interface intelligente. In *Rencontre jeunes chercheurs en Interaction Homme-Machine, Communication publiée dans les actes*, Ile de Berder, mai 2000.
5. C. Duvallet, B. Sadeg, and A. Cardon. How to build real-time multi-agent systems using Anytime technics. In *Proceedings of CATA'2000*, pages 126–129, New Orleans, Louisiane, mars 2000.
6. H. Boukachour, C. Duvallet, and A. Cardon. Multiagent systems to prevent technological risks. In *Proceedings of ACIDCA'2000*, Monastir, Tunisie, 2000.
7. C. Duvallet, B. Sadeg, and A. Cardon. An Anytime Multi-Agent System to manage an Electronic Marketplace. In *Proceedings of AIEC'99, Sydney, Australia*, 1999.
8. C. Duvallet, B. Sadeg, and A. Cardon. Real-time in Multi-Agents Systems. In *Proceeding of CAINE'99, Atlanta*, pages 212–215, 1999.
9. C. Duvallet, Z. Mammeri, and B. Sadeg. Analyse des protocoles de contrôle de concurrence et des propriétés ACID dans les SGBD temps réel. In Teknea, editor, *Real-time systems and embedded systems*, pages 123–139, février 1999.
10. C. Duvallet, Z. Mammeri, and B. Sadeg. Les SGBD temps réel. *Technique et science informatique*, 18(5), pages 479–516, 1999.

Résumé

Les systèmes d'aide à la décision (SAD) doivent permettre aux utilisateurs (décideurs) de prendre les meilleures décisions dans les meilleurs délais. Dans cette thèse, nous nous sommes intéressés aux systèmes qui reposent sur une architecture multi-agents. En effet, les systèmes multi-agents (SMA) permettent de construire des systèmes informatiques ayant recours à l'interrogation multi-critères, souvent utilisée dans les SAD. De façon plus générale, les SMA permettent de concevoir des systèmes qui sont de nature complexe. Cependant, ils n'intègrent pas la notion de contraintes temporelles qui sont souvent très fortes dans les SAD. De plus, dans ces systèmes, des résultats même partiels ou incomplets obtenus dans les temps sont souvent préférés car plus utiles pour la prise de décision que des résultats complets et précis obtenus en retard. Pour cela, les techniques « anytime » (raisonnement progressif) semblent une excellente solution. Dans cette thèse, nous présentons une méthode de conception d'un système multi-agent temps réel basé sur l'exploitation des techniques « anytime ». De plus, nous prenons en compte dans notre modèle l'aspect souvent distribué des SAD.

Mots-clés: systèmes multi-agents, systèmes temps réel, algorithmes anytime, interrogations intelligentes, systèmes d'information, gestion de marché.

Abstract

Decision Support Systems (DSS) should help users to take the best decisions as earlier as possible. In this thesis, we are interested in such DSS which are based on the multiagent paradigm. Indeed, multiagent systems are often the best way used to build computer systems that need multicriteria queries because they permit to design complex computerized systems. Moreover, in DSS, temporal constraints are very strong and in a lot of cases partial or incomplete results obtained early are preferred and more useful than complete and accurate results obtained too late. Anytime techniques seem to be a good solution in order to deal with these situations, that is to take into account real-time aspects in this framework. In this document, we present a method and a model to design a real-time multiagent system is based on anytime techniques. Furthermore, we have taken into account the inherent distributed aspect of current DSS in our model.

Keywords: multiagent systems, real-time systems, anytime algorithms, intelligent interrogation, information systems, marketplace management.