MASTER EN SCIENCES INFORMATIQUES

Conception d'une méthode de modélisation pour systèmes multi-agents: application à un cas particulier

Cayphas, Jean-Louis; Piette, Didier

Award date:
1992

Awarding institution:
Universite de Namur

FACULTÉS UNIVERSITAIRES NOTRE-DAME DE LA PAIX

NAMUR

Institut d'Informatique

21, rue Grandgagnage

5000 Namur

Conception d'une méthode de modélisation pour systèmes multi-agents : application à un cas particulier.

(I)

Jean-Louis Cayphas
Didier Piette

Mémoire de fin d'étude en vue d'obtenir le diplôme de
Licencié et Maître en Informatique

Promoteur : Monsieur R. Lesuisse

Troisième Licence et Maîtrise en Informatique
Année académique 1991-1992

Résumé

Ce travail présente une méthode de modélisation pour la résolution de problèmes distribués multi-agents. Nous étudions la notion d'agent et les concepts de contrôle, d'organisation et de la communications entre ces agents. Cela permet de mieux comprendre et d'aborder les systèmes multi-agents. Nous étudions ensuite la méthode EPAS utile à la modélisation de systèmes distribués. Suite à ces deux premières phases, nous élaborons une démarche de conception permettant d'identifier les agents et leurs tâches que nous modélisons. Nous débouchons sur des spécifications directement supportée par un outil d'aide à la programmation. Ce travail s'accompagne d'une étude de cas appartenant au domaine des jeux.

Abstract

This work present a modelisation method for the solving of distributed multi-agents problems. We at first talk about the notion of agent and we present the control, the organisation and the communication between agents. Thes preliminary steps allow us to have a clear understanding of what about a multi-agents system is. We present the EPAS method which allow the modelisation of distributed systems. Afterwards we propose a conception method based on the agent. Thus we identify the agents and their tasks which we modelise. We arrive on specifications directly supported by a helpful programmation tool. This memoir involves a case study in the scope of games.

Remerciements

Avant d'aborder la présentation de ce travail, il nous semble opportun de placer ici quelques remerciements.

Nous adressons en tout premier lieu nos plus vifs remerciements à Mr Lesuisse, notre promoteur, qui nous a procuré un stage inoubliable et qui nous a judicieusement guidé par ses remarques et encouragements tout au long de ce travail.

Nous tenons également à remercier le professeur B. Moulin de l'université Laval de Québec dont la compétence, la motivation et la disponibilité nous ont été d'un grand secours. Nous remercions également B. Pelletier et L. Cloutier qui nous ont fait part de leurs connaissances.

Finalement, nous sommes très reconnaissant à toutes les personnes ainsi qu'à nos parents pour nous avoir conseillé, documenté et soutenu tout au long de la rédaction de cet ouvrage.

Table des matières

INTRODUCTION

Il y a à peine vingt ans, les systèmes distribués faisaient leur apparition. Dix ans plus tard, alors que leur reconnaissance était bien établie et les principes qui les animaient, étaient solidement ancrés dans les esprits, les systèmes dits intelligents émergeaient progressivement de l'ombre pour donner le jour à des systèmes distribués multi-agents. C'est précisément dans cette catégorie que se situe le présent travail.

En toute logique, nous nous sommes penchés sur les systèmes multi-agents et en avons fait un tour d'étude général afin de mieux cerner les mécanismes les animant : le contrôle, l'organisation et la communication des agents. Le but des deux premiers chapitres est de présenter de façon succincte ces mécanismes.

Lorsque sont sortis ces notions de systèmes intelligents et de systèmes multi-agents, les outils d'aide à la conception étaient essentiellement issus des méthodes systémiques. Nous en avons découvert une parmi d'autres, la méthode EPAS, que nous nous efforçons de présenter le plus clairement possible au chapitre 3. Par la suite, nous l'avons aménagée pour en faire une méthode d'aide à la conception d'un système multi-agent. Nous avons de ce fait constitué une méthode hybride par extraction de mécanismes appartenant aux méthodes objets et à la méthode EPAS. Notre méthode est présentée au quatrième volet de ce travail. Nous nous sommes fixés un double objectif. Tout d'abord, obtenir un approche de conception simple, claire et efficace en sorte de déboucher sur des spécifications aisées à comprendre par un lecteur peu averti. Mais aussi, la méthode doit permettre le passage à un outil informatique existant : le système générique de groupes transitoire, présenté au chapitre 5.

Le jeu FRIK présenté complètement en annexe a servi de support à la vérification des diverse étapes de la méthode de conception. Malgré que son application soit spécifique, nous pensons néanmoins que la méthode est applicable à tous types de systèmes multi-agents de par le caractère général que possède ses différents concepts, extraits de méthodes ayant déjà fait leur preuve.

CHAPITRE 1 : Problématique de la résolution distribuée de problème

Section 1 - Introduction

Nous allons présenter dans ce chapitre l'environnement dans lequel notre travail s'est déroulé. Son objectif consistait à développer une méthode s'adaptant à la résolution distribuée de problème, qu'il s'agissent de distribuer la connaissance, les tâches ou le contrôle de l'exécution. Il existe évidement plusieurs types de méthodes, et nous allons les détailler dans ce premier chapitre, de même que les deux concepts indissociables de la résolution distribuée, à savoir le contrôle de l'exécution et la communication entre agents.

Nous allons également situer la résolution distribuée de problème dans un cadre plus vaste qu'est l'intelligence artificielle distribuée. Celle-ci applique en fait les techniques de traitement des connaissances issues de l'intelligence artificielle à de multiples résolveurs de problèmes. Dans ce contexte, chaque résolveur possède ses propres connaissances et son propre mécanisme de contrôle. L'intelligence artificielle distribuée est donc caractérisée par le fait que le contrôle et les connaissances sont tous les deux distribués, et en ce sens, elle diffère de l'informatique distribuée classique dont l'objectif est la coordination d'un réseau de machines afin d'accomplir un ensemble de tâches disparates et indépendantes.

Pour étudier le contrôle et les connaissances distribués, deux approches de recherche ont émergé ces dernières années : l'approche connexionniste [Fogelman-Soulie 1985] qui consiste à effectuer une décomposition fine du problème à résoudre, et l'approche dite de "l'expertise distribuée" où la décomposition se fait en sous-problèmes. L'objectif de la résolution distribuée de problème concerne la seconde approche et consiste à concevoir un groupe de systèmes intelligents coopérants, capables de résoudre l'ensemble des sous-problèmes obtenus après décomposition d'un problème donné. Pour cela, chaque système doit coopérer avec les autres pour un des sous-problèmes en rapport avec ses connaissances et ses compétences. Finalement, les solutions locales obtenues par chaque système sont synthétisées en vues d'obtenir une solution globale.

L'inclination de rechercher des solutions distribuées est souvent suggérée par les problèmes eux-mêmes qui se situent déjà dans un environnement multi-agents. On peut par exemple concevoir un contrôle pour des agents multiples, produire un plan dans un environnement multi-agents, ou évaluer une situation distribuée par un réseau de senseurs, ces derniers étant considérés comme des agents récepteurs analysant les situations. Toutefois, dans la plupart des applications réelles comme les réseaux de senseurs ou le contrôle d'agents multiples, si la structure du problème posé peut

initialement suggérer une décomposition, celle-ci aboutit rarement à des sous-problèmes triviaux et indépendants. En outre, les sous-problèmes sont souvent insolubles isolément, et même lorsqu'ils le sont, il est difficile de synthétiser les réponses en une solution globale. Ces difficultés ont alors conduit les chercheurs à développer des méthodes de coopération entre les résolveurs distribués de problèmes.

La résolution distribuée de problème est donc caractérisée par la mise en commun des connaissances et des compétences de chaque participant à la résolution. Elle est en outre dépendante du degré de coopération entre les participants, de leur organisation ainsi que des communications échangées entre eux. Ce chapitre positionne la problématique de la résolution distribuée de problème dans l'intelligence artificielle et passe en revue les méthodologies de contrôle et de communication utilisées par les systèmes intelligents impliqués dans la résolution de problèmes. Il est composé de trois parties :

- La première partie présente une taxonomie de l'intelligence artificielle distribuée et situe la résolution distribuée de problème, notamment par rapport aux études menées sur le parallélisme en intelligence artificielle. De manière analogue, une taxonomie relative à la résolution distribuée de problème est proposée afin de cerner les problèmes auxquels doit faire face ce type de résolution, en particulier les stratégies de contrôle des systèmes intelligents coopérants, et la politique de distribution de l'information entre ces systèmes.

- Le contrôle dans les systèmes de résolution distribuée de problème peut varier suivant trois dimensions : le degré de coopération entre participants à la résolution, l'organisation du groupe, et enfin l'évolution de cette organisation en cours de résolution. L'analyse des principaux modes de contrôle fait l'objet de la deuxième partie

- Enfin, la nature et la qualité des communications entre les systèmes intelligents coopérant à la résolution d'un problème sont abordées dans la troisième partie.

Section 2 - Taxonomie, caractéristiques et intérêts de la résolution distribuée de problème

Le but de cette section est de situer la résolution distribuée de problème par rapport à l'intelligence artificielle distribuée, et de présenter ses principaux éléments constitutifs, afin d'en dégager les caractéristiques et les intérêts attendus. Auparavant, quelques termes porteurs spécifiques sont définis.

2.1 - Systèmes intelligents, agent et environnement multi-agents

Nombreux sont les chercheurs qui placent sous les vocables *systèmes intelligents* et *agents* des significations parfois fort éloignées les unes des autres. Le sens donné ici à un "système intelligent" est celui de tout système capable d'exploiter ses connaissances, de les enrichir, et si possible de les acquérir. Il doit, en outre, être capable d'interagir avec toute autre système intelligent.

Le terme "agent" fait, lui aussi, l'objet d'acceptions parfois fort éloignées les unes des autres. Ferber et Ghallab [1988] en donnent la définition suivante :

> "On appelle agent une entité physique ou abstraite capable d'agir sur elle-même et sur son environnement. Cette entité dispose d'une représentation partielle de cet environnement et peut communiquer avec d'autres agents. Son comportement est la conséquence de sa perception du monde, de ses connaissances et des interactions avec les autres agents".

Il convient de remarquer que la définition du terme "agent" est beaucoup plus précise que celle de système intelligent. Toutefois, certains auteurs préfèrent utiliser "système intelligent" pour marquer, sans doute, l'évolution des systèmes qui sont devenus "intelligents", donc capables d'acquérir et de traiter les connaissances. En revanche, les chercheurs qui travaillent sur les environnements multi-agents, préfèrent utiliser le terme "agent", pour marquer généralement cet environnement particulier. En ce qui nous concerne, les deux termes auront dorénavant le même sens et il seront utilisés indifféremment.

En fait, la définition du terme "agent" suggère deux classes d'entités :

1. La première classe concerne l'agent en tant qu'entité physique autonome, percevant et agissant sur l'environnement extérieur. C'est le cas par exemple d'un robot dans un environnement où évoluent d'autres robots et des opérateurs humains. Dans ce cas, l'agent possède une autonomie d'action, donc de décision, et ses interactions avec les autres agents se font au moyen de systèmes de perception et de communication.

2. Dans la seconde classe, l'agent est une entité informatique. Il s'agit alors d'un

module logiciel et/ou matériel plus ou moins élaboré, interagissant avec d'autres modules selon une architecture particulière. Les systèmes multi-experts où chaque système est constitué de sa propre base de connaissance et de son propre mécanisme de contrôle illustrent cette seconde classe d'agents.

Il y a bien entendu une très forte corrélation entre les problématiques de ces deux classes au point de vue de l'organisation des agents, l'évolution de cette organisation, ainsi que les méthodologies de communication. Toutefois, les motivations et les contraintes sont distinctes. Dans le cas où l'agent est une entité informatique, les raisons qui conduisent à privilégier l'approche fondée sur la résolution distribuée sont dictées par une nécessité de réduire la complexité du système informatique en mettant en place une modularité permettant d'améliorer les performances et la fiabilité.

En revanche, dans le cas où l'agent est une entité autonome, on met en avant les notions d'actions sur le monde extérieur, et les interactions entre agents. Ceci privilégie alors les problématiques de planification, de contrôle d'exécution et également les performances et la fiabilité.

Nous allons approfondir davantage cette seconde classe d'agents et mettre en évidence une théorie formelle qui a pour but de fournir des spécifications à la création d'agents artificiels autonomes capables de coopérer. Bien entendu, pour mener à bien cette étude théorique, et pour faciliter la lecture de ce premier chapitre, il convient tout d'abord de situer la résolution distribuée de problème par rapport à l'intelligence artificielle distribuée et de faire ensuite l'analyse de sa problématique.

2.2 - Intelligence artificielle distribuée et résolution distribuée de problème

Deux taxonomies sont décrites ci-dessous. La première est relative à l'intelligence artificielle distribuée, la seconde à la résolution distribuée de problème. Celles-ci permettent de situer la résolution distribuée, non seulement par rapport à l'intelligence artificielle distribuée, mais aussi par rapport à l'intelligence artificielle d'une façon générale.

2.2.1 - Taxonomie de l'intelligence artificielle distribuée

L'intelligence artificielle distribuée se réfère à la décomposition du problème à étudier, qui peut être soit très fine, c'est à dire de la taille d'une instruction-machine, auquel cas on s'intéresse au parallélisme, soit de la taille des bases de connaissances habituelles, auquel cas on s'intéresse à la distribution de l'expertise. Si les différents fragments de cette expertise sont globalement gérés par un seul mécanisme de contrôle, ils constituent alors des sources de connaissances. En revanche, lorsque chaque fragment possède son propre mécanisme de contrôle, il forme un résolveur

distribué de problème.

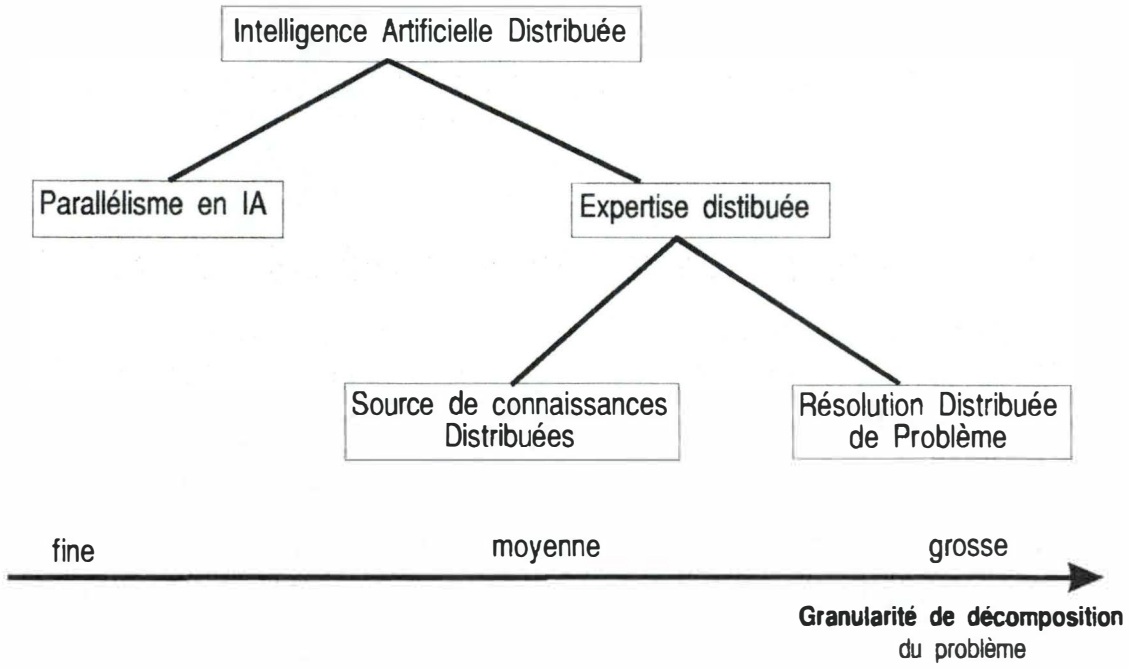

Figure 1.1 Taxonomie de l'intelligence artificielle distribuée (d'après Decker [1987])

Pour illustrer ces distinctions, Decker [1987] a proposé la taxonomie présentée à la figure 1.1 qui situe le parallélisme et l'expertise distribuée en fonction de la granularité selon laquelle le problème a été décomposé. L'expertise distribuée implique une décomposition du problème en plusieurs sous-tâches où l'expertise nécessaire est généralement de l'ordre de grandeur d'une base de connaissances. Ceci caractérise en fait une granularité moyenne ou grosse, par opposition aux décompositions à granularité beaucoup plus fine, c'est à dire du niveau de l'instruction et qui relèvent des études sur le parallélisme.

Bien entendu, décomposer un problème suivant une grosse ou une moyenne granularité est une tâche beaucoup plus cognitive que la décomposition en granularité fine dont l'objectif est d'améliorer l'efficacité de la résolution du problème.

2.2.2 - Taxonomie de la résolution distribuée de problème

La résolution distribuée de problème s'attache à définir, d'une part l'agent au sein d'un groupe, et d'autre part le groupe en tant qu'entité (société d'agents). La mise en oeuvre de l'agent implique de le doter de capacités "sociales" telles que :

- *le raisonnement sur autrui*, c'est à dire la capacité de modéliser les connaissances d'autres agents, leurs compétences, leurs objectifs et leurs intentions, de façon à pouvoir prédire leur comportement;

- *la cohabitation et la coopération avec autrui*, de façon à partager des ressources (cohabiter), ou à partager des tâches et des buts (coopérer);

- *la capacité d'évaluation d'une situation complexe distribuée* où interviennent plusieurs agents, afin d'avoir une "vue globale" du groupe et de mettre à jour son modèle du monde environnant et les modèles d'autrui en les complétant si possible au moyen des *communications*.

La mise en oeuvre du groupe d'agents impose, quant à elle, de définir :

- *une organisation* efficace du groupe en recherchant des méthodes et des moyens de cohabitation, de coopération, et de communication;

- l'*évolution* de cette organisation, qui doit veiller à maintenir la cohérence globale du groupe.

Decker [1987] a suggéré une taxonomie de la résolution distribuée selon les deux principales difficultés soulevées par ce type de résolution : la répartition du contrôle et le mode de communication entre les agents. Comme le montre la figure 1.2, la définition des modes de communication entre agents soulève des difficultés concernant les moyens de communications, le contenu des messages échangés, le protocole d'échange, et enfin, la nécessité de réduire les informations échangées. La définition du contrôle concerne, quant à elle, le degré de coopération entre les agents, leur organisation et l'évolution de cette organisation pendant la résolution. Toutes ces notions sont détaillées dans ce qui suit.

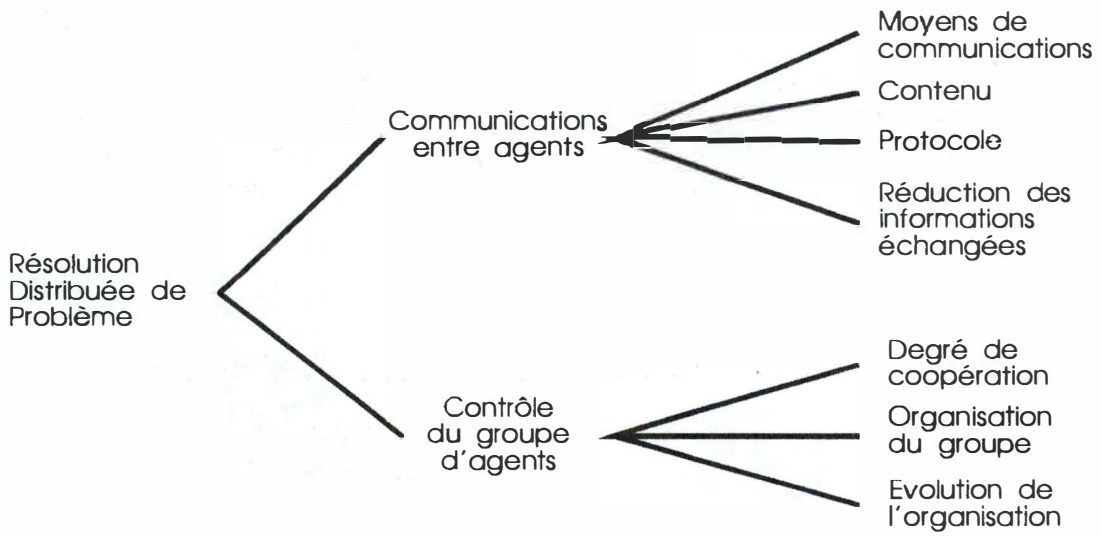

Figure 1.2 Taxonomie de la résolution distribuée de problème (Decker [1987])

Après avoir présenté la résolution distribuée de problème, il convient maintenant de préciser ses caractéristiques ainsi que ses intérêts et ses domaines d'applications.

2.3 - Caractéristiques, intérêts et domaines d'applications de la résolution distribuée de problème

La résolution distribuée de problème relève plus particulièrement des techniques des traitements de connaissances distribuées, donc de l'intelligence artificielle, et en ce sens, elle diffère aussi bien de l'informatique distribuée classique, que des systèmes hiérarchisés classiques. Ce paragraphe a pour objectif de le montrer en indiquant les intérêts et les domaines d'applications de la résolution distribuée.

2.3.1 - Caractéristiques et intérêts de la résolution distribuée

L'intelligence artificielle distribuée a emprunté à l'informatique classique distribuée une certaine terminologie telle que la coopération, la négociation, le protocole de communication, etc.

Toutefois, les réseaux de résolveurs distribués de problèmes diffèrent des systèmes informatiques classiques de traitement de données par le style de distribution et par le type de problèmes abordés (Smith, Davis [1981]). Ces différences sont plus apparentes si on examine plus particulièrement les interactions entre les agents dans chaque type de réseau. Ainsi, dans un réseau de données classiques qui comporte généralement plusieurs tâches s'exécutant en parallèle, l'accès partagé aux ressources informationnelles est le but principal des interactions entre les tâches. L'objectif dans un tel réseau est de donner l'*illusion* que chaque tâche s'exécute seule sur un système d'exploitation qui gère les interactions de partage de ressources et les conflits entre les tâches dans le réseau.

Au contraire, dans les réseaux de résolution distribuée de problème, les modules (i.e. entités informatiques) sont explicitement *conscientes* de la distribution des composants du réseau, et peuvent même prendre des décisions sur la base de cette information. Cette différence est due en partie aux caractéristiques des applications abordées par les méthodologies de la résolution distribuée, et qui sont décomposées en tâches dont l'exécution par chaque agent résolveur requiert l'assistance d'autres agents.

Dans le même cadre, il convient aussi de distinguer les systèmes intelligents conçus en vue d'une résolution distribuée de problème, des systèmes hiérarchisés classiques. D'une façon générale, l'objectif de ces derniers s'intègre dans la recherche de nouvelles méthodes de contrôle-commande pour des systèmes complexes. Cette recherche impose :

1. de nouvelles méthodes d'analyse des systèmes complexes, étant donné le nombre important de variables à traiter;

2. mais aussi et surtout une étape intermédiaire avant la commande, qui peut consister :

   - soit à réduire la dimension du problème par une procédure d'agrégation de variables; pour ensuite appliquer, sur un problème de dimension acceptable, des méthodes "classiques" et bien connues actuellement (programmation mathématique, principe de maximum, programmation classique, etc.).

   - soit à décomposer le système de façon à obtenir des sous-systèmes pour mettre en oeuvre des méthodes de décomposition-coordination et à élaborer ainsi que des structures de commande à plusieurs niveaux (commande hiérarchisée).

3. la synthèse d'algorithmes de commande utilisant ces principes de décomposition-coordination et la synthèse des structures de commande multi-niveaux qui permettent la mise en oeuvre effective de ces algorithmes.

En revanche, les systèmes intelligents qu'ils soient des systèmes complètement automatisés (robots) ou qu'ils soient des systèmes d'aide à la décision du type système expert, sont conçus de façon à assurer la continuité des systèmes de commande automatique dans lesquels le degré d'intelligence va croître de plus en plus. Ces systèmes mettent en avant explicitement les problèmes de modélisation et de raisonnement sur autrui et peuvent s'organiser de façon à s'adapter à toute nouvelle situation.

En fait, la résolution distribuée de problème combine les recherches menées dans le cadre de l'intelligence artificielle et du traitement de connaissances distribuées (Chandrasekaran [1981]; Davis [1982]). Ainsi, dans tout réseau de résolveurs distribués qui peut être considérer comme un réseau d'agents interagissant de façon coopérative à la résolution d'un même problème, chaque agent est lui-même un système de résolution complexe capable non seulement de modifier son comportement suivant les circonstances, mais aussi de planifier ses stratégies de coopération et de communication avec les autres agents.

De ce fait, l'intérêt et l'engouement des chercheurs pour l'intelligence artificielle distribuée, et plus particulièrement pour la résolution distribuée de problème, s'explique par plusieurs raisons. La plupart de ces raisons ont déjà été évoquées dans l'introduction comme la coopération homme-machine, l'apport aux sciences cognitives, ou la possibilité d'apporter des solutions à des problèmes complexes. Elles sont, par ailleurs, complétées par les objectifs fonctionnels et l'intérêt culturel suivants :

- La technologie informatique a progressé de façon telle que les réseaux de résolveurs distribués sont devenus une réalité économique. Ce bond technologique permet actuellement d'avoir des dizaines, et à terme des centaines, de résolveurs de problèmes. L'utilisation efficace de ces résolveurs est un des buts de la recherche menée en intelligence artificielle distribuée.

- Pour les applications naturellement distribuées, ainsi que pour celles dont la

fonctionnalité ou la complexité suggèrent une décomposition en sous-systèmes, une architecture distribuée basée sur un réseau de résolveurs offrent donc beaucoup d'avantages. Il en est ainsi des applications suivantes, naturellement distribuées, citées à titre d'exemples : le contrôle de trafic aérien, la maintenance d'un réseau de distribution d'énergie nationale, la reconnaissance de signaux par plusieurs senseurs, le contrôle du trafic routier, la coopération entre plusieurs robots, etc.. Parmi les applications qui suggèrent une décomposition fonctionnelle, la reconnaissance de la parole et la fabrication des circuits intégrés à très grande échelle, sont sans aucun doute des tâches qui nécessitent, plus que d'autres, d'être étudiées selon une approche fondée sur la résolution distribuée, car elles mettent en jeu différents "experts" qui doivent coopérer.

- A ces objectifs fonctionnels s'ajoute un intérêt culturel visant à comprendre le processus d'interaction, et plus particulièrement la coopération entre agents. Il semble qu'à ce jour, les chercheurs comprennent mieux le processus de compétition entre agents dans les systèmes biologiques, mécaniques ou sociaux, que le processus de coopération. Il est alors possible que le développement des recherches de la résolution distribuée de problème puisse apporter une architecture de base à des modèles dans des domaines tels que la sociologie, la gestion ou la théorie de l'organisation, et de la même façon, que les systèmes d'intelligence artificielle soient utilisés pour tenter de modéliser les démarches de résolution de problème en automatisation humaine.

Il convient de signaler aussi que certains domaines d'applications se prêtent mieux que d'autres à la coopération entre entités intelligentes pour aboutir à un objectif global. Quelques uns de ces domaines sont maintenant abordés.

2.3.2 - Exemples de domaines d'applications de la résolution distribuée de problème

La plupart des travaux actuels dans le domaine de la résolution distribuée de problème portent sur les réseaux de senseurs, tels que le contrôle de trafic aérien, ou les systèmes robotiques distribués (Davis [1980]; Fehling, Erman [1983]; Smith [1985]). Toutes ces applications nécessitent une interprétation distribuée, c'est à dire une évaluation de situation et une planification distribuée au moyen de différents capteurs "intelligents" appelés senseurs. La planification ne se réfère pas uniquement aux actions à effectuer, mais aussi à l'utilisation des ressources matérielles et cognitives pour accomplir de manière efficace les tâches d'interprétation et de planification des actions.

En plus de la similitude des tâches génériques à résoudre, ces domaines d'application sont caractérisés par une distribution naturelle des senseurs et des effecteurs dans l'espace. Ces derniers peuvent être considérés, dans ce cas ci, comme les agents qui agissent. Par ailleurs, les sous-problèmes d'interprétation des données sensorielles et de planification des actions sont interdépendantes dans l'espace et dans

le temps. Par exemple, dans le contrôle de trafic aérien, un plan pour guider un avion doit être coordonné avec les plans des avions voisins afin d'éviter les collisions.

Cette interdépendance est encore plus visible dans les réseaux de senseurs, en raisons des recouvrements possibles des zones captées. La meilleure façon d'exploiter ces redondances, pour éliminer l'incertain et l'imprécis, est de faire coopérer un groupe de senseurs voisins pour interpréter les différentes zones captées. Si tel est le cas, la résolution par le groupe est vue comme la résolution d'un seul problème, et non comme celle d'un ensemble de sous-problèmes indépendants.

2.3.3 - Conclusion

Il ressort de cette présentation que la résolution distribuée de problème concerne les domaines où l'expertise est dispersée sur plusieurs systèmes possédant chacun leur propre mécanisme de contrôle, leurs propres connaissances et leurs propres compétences. En outre, chaque système évolue dans un univers qui nécessite des interactions avec les autres agents pour un partage de tâches, des informations ou des ressources. Cette interaction s'étend de la coopération totale jusqu'au conflit (on parle alors de degré de coopération), et ne peut être efficace que si le groupe s'accorde sur une politique organisationnelle et une politique de distribution de l'information. La politique organisationnelle doit faire intervenir l'organisation adoptée par le groupe et l'évolution de cette organisation en cours de résolution. La politique organisationnelle, l'évolution de l'organisation et le degré de coopération constituent le *contrôle*. La politique de distribution de l'information, quant à elle, doit faire intervenir les méthodes et les moyens de *communications*, le protocole utilisé et le contenu échangé. Celui-ci doit être aussi condensé que possible pour des raisons liées à l'incertitude de l'information, à la limitation de la bande passante des canaux de transmission et à leur non-fiabilité. Ainsi, dans la résolution distribuée de problème, les *communications* font intervenir les moyens et les méthodes d'échanges d'informations entre agents. Le contrôle et les communications sont abordés successivement dans la suite de ce chapitre.

Section 3 - Le contrôle dans la résolution distribuée de problème

Le contrôle dans les systèmes de résolution distribuée de problème peut varier suivant trois dimensions : le degré de coopération entre les agents, l'organisation du groupe d'agents et l'évolution de cette organisation en cours de résolution. Le but de cette partie du chapitre est d'aborder ces trois dimensions, en mettant l'accent sur les objectifs de chacune.

3.1 - Coopération entre agents

3.1.1 - La coopération et ses objectifs

Aujourd'hui, l'arrivée en masse des calculateurs dans les environnements humains a donné lieu à d'intéressants problèmes, qui renforcent l'intérêt de la coopération homme-machine et machine-machine. La figure 1.4 illustre ces deux types de coopération où la structure recherchée se fait généralement suivant trois niveaux (Sheridan [1985]) : les calculateurs locaux, un étage de coordination des calculateurs locaux et un superviseur humain.

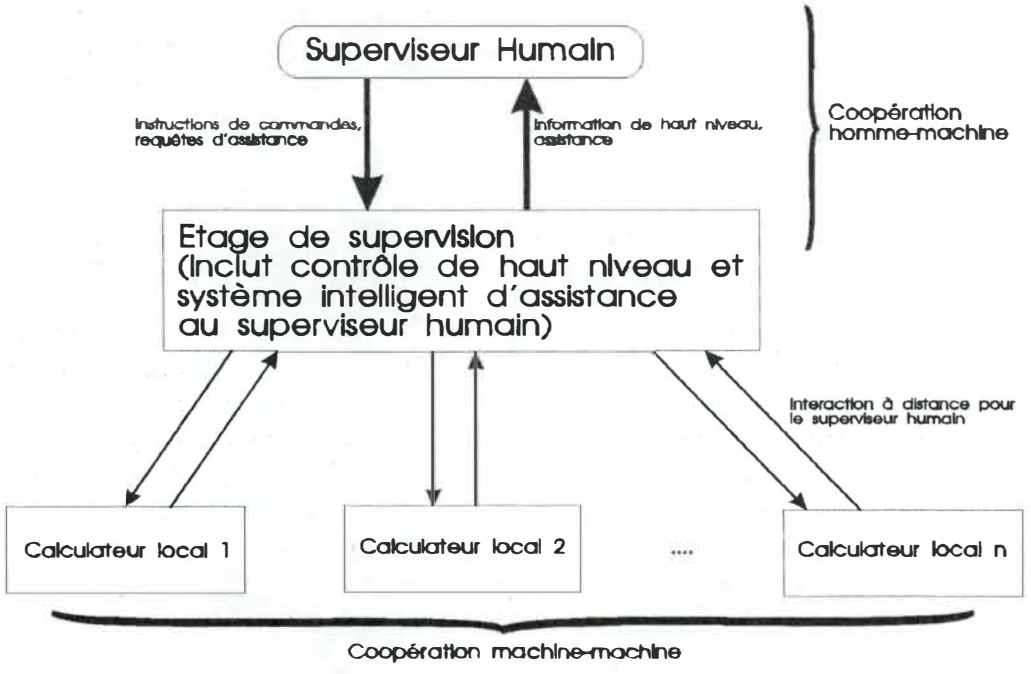

Figure 1.3 Coopération homme-machine et machine-machine

Cependant, si la coopération entre humains est facilitée par une bonne compréhension mutuelle où chacun peut prédire les actions des autres, tel n'est pas le cas des interactions homme-machine. En effet, la compréhension entre les humains est due aux connaissances partagées, c'est à dire à la culture générale que partagent les être humains. En revanche, les interactions homme-machine sont souvent sujettes à des frustrations et à de mauvaises compréhensions, car ni l'homme, ni la machine n'ont une vue adéquate de leurs domaines de compétence respectifs. Les mêmes problèmes se posent dans les interactions machine-machine car la capacité de compréhension de la machine est assez primitive, voire quasiment inexistante en regard à celle des humains.

Bien entendu, si des agents humains ou artificiels coopèrent, c'est dans le but d'atteindre un ou plusieurs objectifs de la coopération. Parmi ceux-ci, il convient de citer celui de l'amélioration de la performance globale, puisque les agents travaillent en parallèle tout en cherchant à minimiser, si possible, leur temps d'attente vis-à-vis des résultats partiels, pour mieux coordonner les activités du groupe.

La coopération permet aussi de développer une grande variété de solutions et de points de vue lorsque le groupe s'active sur une même tâche. La probabilité de trouver une solution s'en trouve aussi augmentée, car lorsque certains agents sont incapables de trouver des solutions à leurs tâches, il suffit alors d'assigner celles-ci à d'autres agents plus compétents. Si, de plus, chaque agent est autorisé à vérifier les résultats de l'autre, le degré de confiance de chaque solution se trouve alors amélioré.

L'objectif de la coopération vise aussi à réduire l'effort inutile, si les agents savent reconnaître et éviter les activités redondantes. Dans le même temps, les agents peuvent aussi améliorer l'utilisation de leurs ressources, s'ils équilibrent leur charge de travail par un partage dynamique des tâches.

Tout en cherchant à atteindre un ou plusieurs de ces objectifs, les agents doivent s'attacher à réduire la quantité d'informations échangées en étant plus sélectifs vis-à-vis des messages échangés, car les moyens de communication sont limités physiquement. Pour cela, les agents doivent orienter les échanges d'informations sur les messages à caractère prédictif de façon à permettre à chacun de déterminer son futur comportement. Ceci a pour conséquence d'améliorer la cohérence du groupe.

Par ailleurs, il est clair que les objectifs de la coopération peuvent être conflictuels et que les agents ne peuvent pas les réaliser tous. Ainsi, la coopération suit chaque solution particulière de résolution de problème. Par exemple, l'élaboration d'une solution, en un minimum de temps, exige d'une part la réduction de l'échange d'informations notamment pour les vérifications, et d'autre part, l'abandon d'un développement d'une grande variété de solutions.

En fait, ces objectifs ne peuvent être atteints que si les agents sont d'accord pour une coopération totale sans restriction. Or, dans les environnements multi-agents, donc réels, la coopération n'est pas toujours totale et peut varier de l'entente totale entre agents jusqu'à l'ignorance complète. Ceci caractérise ce qu'on appelle le degré de coopération entre agents. Ce point est maintenant abordé.

3.1.2 - Degré de coopération

Le degré de coopération caractérise en fait la coopération d'un point de vue quantitatif. En effet, la coopération entre agents dans un environnement réel s'étend de la coopération totale jusqu'à l'antagonisme complet.

Ainsi, d'un côté se trouvent les *systèmes à coopération totale* qui sont capables de résoudre des sous-problèmes interdépendants, avec toutefois une surcharge de communications. Ces systèmes sont même capables de changer leurs objectifs pour satisfaire les besoins d'autres agents, si la situation l'exige. D'un autre côté, se trouvent les *systèmes antagonistes* qui peuvent ne pas coopérer du tout et même avoir des objectifs contradictoires aboutissant alors à un blocage. La charge de communication requise par ces agents est généralement faible.

Les systèmes traditionnels, quant à eux, sont des systèmes intermédiaires se situant entre les deux types précédents. Il est à noter que la plupart des systèmes réels possèdent un faible degré de coopération.

Le recours à des systèmes où la coopération est totale se fait généralement lorsque le contrôle multi-agents est incertain, c'est à dire quand la distribution de l'information diffère des décisions qui sont actuellement prises. La présence d'autres incertitudes relatives par exemple à l'incomplétude ou à l'imprécision de la représentation de l'information, ou au mélange d'informations issues de sources multiples, rend encore plus crucial le besoin d'une coopération totale. En outre, l'entente totale entre agents est particulièrement utile quand les agents ne savent pas clairement ce qu'ils doivent faire ni quelle information ils doivent échanger (Lesser and al. [1981]).

La coopération totale est aussi recommandée dans les problèmes où chaque agent a une connaissance spécifique. En effet, les systèmes d'intelligence artificielle sont souvent construits à partir d'experts *coopératifs* possédant chacun une connaissance globale de son domaine particulier.

D'une façon générale cependant, l'hypothèse d'une coopération totale peut s'avérer inappropriée, dans la mesure où les systèmes réels peuvent avoir à un moment ou à un autre des objectifs conflictuels. Seuls Genesereth, Ginsberg, Rosenschein [1986] et Rosenschein, Genesereth [1984, 1985] ont, à notre connaissance, étudié le degré de coopération. En revanche, les autres recherches menées sur la coopération supposent que les agents sont *bienveillants*, et que par conséquent, la coopération entre eux est totale.

En résumé, le degré de coopération est une composante importante du contrôle, car le type de problème et l'échange d'information en dépendent. En effet, les systèmes optant pour une coopération totale sont capables de résoudre des sous-problèmes interdépendants, au détriment toutefois d'une surcharge de communication, alors que les systèmes antagonistes peuvent ne pas coopérer du tout, et en ce sens, les problèmes qu'ils sont à même de résoudre doivent être complètement indépendants.

Une autre composante importante, du contrôle multi-agents, est l'organisation du groupe d'agents. Cette composante est maintenant abordée.

3.2 - Organisation du groupe d'agents

La manière dont les agents sont disposés à coopérer, pour résoudre certains problèmes en commun, est très importante. Elle a une incidence directe sur la coordination des activités, sur la répartition des tâches, et sur la cohérence globale du groupe. Il en résulte alors que les agents doivent développer un type d'organisation pour au moins une tâche donnée. C'est ce que s'attache à montrer la première partie de ce paragraphe, avant de présenter les différents types d'organisations dans la seconde partie.

3.2.1 - Nécessité d'une organisation

Un des problèmes majeurs de la résolution distribuée de problème concerne l'obtention et la maintenance d'une *cohérence globale* au niveau du groupe d'agents coopérants. Un comportement cohérent signifie que les solutions, élaborées par les agents distribués, doivent non seulement être acceptables localement et réaliser ainsi les tâches assignées, mais aussi s'intégrer correctement aux solutions des autres agents impliqués dans des tâches dépendantes. Ce comportement cohérent passe d'abord par une *assignation optimale des tâches*. Autrement dit, il convient d'assigner la tâche qu'il faut à l'agent qui convient. Or, une telle distribution n'est pas évidente, puisque à chaque tâche peuvent participer plusieurs agents ayant les qualifications requises. Ainsi, le groupe doit développer des stratégies coopératives pour que cette assignation se fasse suivant les compétences, les connaissances et les capacités de chacun. De plus, il doit veiller à ce que cette assignation soit complète et consistante. En précisant toutefois que l'assignation est dite complète si tous les *rôles*, nécessaires à la résolution du problème global, sont attribués aux agents de façon à assurer une couverture totale du problème; alors que si aucune tâche n'est affectée aux agents inutiles, l'assignation est dite *consistante*.

Bien entendu, la limitation des connaissances des agents aggrave la difficulté de l'assignation optimale et de la couverture totale du problème. En effet, aucun agent n'a généralement une connaissance globale de tous les rôles ou de toutes les tâches à effectuer. En outre, la difficulté d'obtention d'une assignation optimale est également aggravée par le fait que chaque agent ne sait pas pour quelles tâches il est approprié.

Enfin, la maintenance d'une cohérence globale nécessite aussi de résoudre des *problèmes de coordination de tâches* lorsque celles-ci ne sont pas indépendantes. Ceci résulte, une fois de plus, des connaissances limitées de chaque agent. En effet, les agents ont généralement des connaissances locales concernant leur propre environnement ainsi que leurs tâches et leurs intentions, et en fonction de cela, ils peuvent difficilement prédire les interactions possibles entre leurs rôles et ceux des autres agents.

Au vu de tous ces problèmes, les agents coopérant à la résolution d'un même problème doivent immanquablement développer un type d'organisation. Celle-ci a pour but de transformer une collection d'agents prêts à coopérer en un réseau structuré fixé au moins pour une tâche donnée. L'organisation spécifie en fait la manière dont une tâche doit être décomposée en sous-tâches pour que celles-ci soient ensuite assignées

aux agents du groupe, de façon à leur permettre de résoudre un problème en commun. De plus, elle fixe aussi, si elle est hiérarchisée, la manière dont les communications peuvent se faire, car les liaisons hiérarchiques indiquent aussi quelles sont les paires d'agents autorisés à s'échanger des informations.

Bien entendu, une fois la tâche décomposée, l'assignation des sous-tâches spécifiées, et la manière dont les communications pourront se faire, suggérée, les problèmes d'incohérence et de mauvaise coordination, auront très peu de chance de se produire.

Une autre raison de recourir à des structures organisationnelles est donnée par le principe de *rationalité limitée* de Simon [1957]. En effet, Fox [1981] explique comment ce principe, valable dans les sociétés humaines, peut s'appliquer aux systèmes distribués. D'après la théorie de Simon, la capacité de penser d'un cerveau humain est limitée non seulement par la quantité de données manipulées, mais aussi par le niveau de détail du contrôle qu'une personne peut efficacement effectuer. De la même manière, un processeur possède également une capacité de traitement limitée ne serait-ce que par le nombre d'instructions qu'il peut traiter par seconde. L'analogie entre processeur et capacité humaine semble donc établie. Ceci implique donc le besoin de recourir à des organisations complexes pour résoudre des problèmes complexes.

Ainsi, la résolution distribuée de problème passe forcément par le développement d'une structure "organisationnelle". Il convient donc de préciser quelles sont les différents types d'organisations que les agents sont susceptibles de développer.

3.2.2 - Les différents types d'organisations

Dans le cadre de la résolution distribuée de problème, et parallèlement aux organisations humaines, la gamme des organisations possibles s'étend des groupes totalement libres, appelés *équipes ou comités*, à des groupes dont les relations sont du type *maître/esclaves*. Les comités n'ont pas de hiérarchie de contrôle et sont dirigés par les données. Un problème est alors résolu, par ce type d'organisation, quand n'importe quel membre de l'équipe arrive à une solution, celle-ci n'étant pas nécessairement optimale. En revanche, les organisations du type maître/esclaves sont très hiérarchisées et tout agent est contraint d'agir immédiatement à la demande de ses supérieurs.

Entre ces deux organisations en comités et en groupes de type maître/esclaves, il existe plusieurs niveaux de hiérarchie possibles, dans lesquelles des agents peuvent être chargés d'élaborer des solutions intermédiaires, ou de donner des directives aux autres agents hiérarchiquement inférieurs. Dans une organisation où existent ces niveaux hiérarchiques intermédiaires, un agent n'est pas contraint d'agir immédiatement à la demande de ses supérieurs et le contrôle passe d'un niveau hiérarchique à un autre par la négociation. Les agents de bas niveau sont dirigés par les données, tandis que ceux du niveau supérieur sont dirigés par les buts, et tous les agents

intermédiaires peuvent être dirigés par les uns ou par les autres. Toutefois, tous les systèmes ne permettent pas la négociation, et les agents doivent dans ce cas se conformer aux ordres donnés par leurs supérieurs.

Il convient de noter que les organisations les plus utilisées en intelligence artificielle distribuée sont généralement celles qui permettent l'interprétation distribuée. Dans ce contexte, on trouve deux types d'organisations :

- La première organisation est communément désignée en intelligence artificielle sous le nom *d'experts coopératifs*. En général, les organisations de ce types sont composées de spécialistes qui résolvent les problèmes par partage de perspectives et donc la structure hiérarchique est faible ou inexistante. Le travail le plus connu de ce type en intelligence artificielle distribuée est le système de compréhension de la parole Hearsay II (Erman, Hayes-Roth, Lesser, Reddy [1980]) qui sera détaillé à la section 4.1 du chapitre 2.

- La deuxième organisation appelée *théorie Y* (Lawrence, Lorsch [1967]) ou *cône perceptuel* est, contrairement à la première, très hiérarchisée. Les organisations de ce type sont assemblées suivant une hiérarchie stricte de niveaux d'abstraction. A chaque niveau, les éléments individuels reçoivent des rapports émis par les niveaux inférieurs, les intègrent selon leurs capacités et leur positions hiérarchiques, et transmettent leurs résultats aux niveaux supérieurs.

Ces deux types d'organisations visent des problèmes de nature très différente. Ainsi, la première organisation est plutôt orientée vers des problèmes d'évaluation distribuée de situation dans lesquels il n'y a aucune hiérarchie de niveaux, alors que la seconde est plus apte aux problèmes décomposables en différents niveaux hiérarchiques tout comme l'informatique classique.

Une autre organisation, située entre les deux précédentes est proposée par Kornfeld et Hewitt [1981]. Cette organisation est issue de l'observation des communautés scientifiques et du constat que celles-ci font des systèmes hautement parallèles. Ses auteurs l'ont appelée *métaphore de la communauté scientifique.* Dans ce type d'organisation, les agents sont organisés en des niveaux faiblement couplés. Un premier niveau d'agents propose des solutions possibles au problème à résoudre. Un deuxième niveau d'agents est chargé de collecter et de mettre en évidence les solutions proposées, tandis que des agents spécifiques recherchent des arguments pour désapprouver ces solutions. Enfin, des agents évaluateurs examinent les solutions proposées, et demandent au système de confirmer les propositions qui semblent les plus favorables.

Il importe de souligner, pour finir, que le choix d'un type d'organisation à lui seul est insuffisant, car il n'est pas sûr que l'organisation choisie au départ évoluera dans le sens voulu, en cours de résolutions. Il est donc important de préciser la dynamique que doit avoir l'organisation choisie.

Section 4 - Communications entre agents

L'échange d'informations entre les agents participant à la coopération est totale, non seulement pour résoudre leurs tâches locales, mais pour améliorer la cohérence et la coordination au niveau du groupe.

Cependant, la limite en capacité des moyens d'échange d'informations ainsi que les risques d'imprécision et d'incertitude dus à la transmission de l'information imposent de minimiser l'information échangée. Pour tenter de résoudre ce dilemme, Shimanoff [1984] a identifié sept règles destinées à guider la conception des moyens de communications. Ces règles sont évoquées sous la forme suivante :

- par quel moyen le message est-il transmis ?
- quel est le contenu du message ?
- qui est l'expéditeur du message ?
- à qui le message est-il envoyé ou quel est le destinataire ?
- à quel instant le message est-il émis ?
- quelle est la durée et la fréquence d'émission du message ?
- selon quelle procédure d'émission le message a-t-il été envoyé ?

Ces règles peuvent être explicites, implicites, ou élaborées, et servent à établir la cohésion du groupe dans l'accomplissement de ses objectifs.

4.1 - Paradigme de communication

Les moyens de communications entre agents relèvent de deux techniques, l'une fait transiter les informations par une mémoire globale partagée et appelée méthode du blackboard, l'autre consiste à établir des liaisons directes entre les agents pour échanger des messages. Chacune des deux méthodes présente des avantages et des inconvénients qui sont évoqués ci-après. Il faut noter que plusieurs systèmes utilisent les deux méthodes de façon simultanée : une mémoire commune pour le partage des informations entre agents travaillant localement sur un sous-problème, et un échange par messages pour les communications entre les groupes d'agents.

4.1.1 - Communication par partage d'informations : le modèle de blackboard

En intelligence artificielle, le modèle le plus utilisé pour l'échange d'informations au moyen d'une mémoire globale partagée, est le tableau noir ou blackboard (Hayes-Roth [1985]; Chaib-Draa, Millot [1987]). Celui-ci, constitué de plusieurs niveaux d'abstraction du problème à résoudre, est en fait une mémoire dans laquelle chaque agent peut

rechercher une information, inscrire des résultats partiels ou des hypothèses, voire même des messages. Par ce moyen, les agents se partagent les interprétations partielles dans la mesure où une nouvelle hypothèse ou un résultat partiel porté sur le blackboard par l'un des agents peut s'avérer intéressant pour un autre. Si c'est le cas, l'agent intéressé est évoqué par un mécanisme d'agenda.

Nous reviendrons plus en détail sur le blackboard dans la section 4.3 du chapitre 2, lors de l'étude du modèle Hearsay II dont il est issu.

4.1.2 - Communication par envoi de messages

A l'inverse des techniques de partage d'informations basées sur le blackboard, les systèmes fondés sur la communication par messages relèvent d'une distribution totale à la fois des connaissances, des résultats partiels et des méthodes utilisées pour aboutir à un résultat (Yang, Huhns, Stephens [1985]). Les travaux de Hewit et de son équipe sur les langages d'acteurs en sont une excellente représentation (Hewit [1976]; Hewit et Kornfeld [1980]; Clinger [1981]). Ces systèmes sont caractérisés par :

- *Le traitement local* : un agent ne peut manipuler que sa base de connaissances locale; il peut cependant envoyer des messages aux autres agents qu'il connaît, appelés ses *accointances*, et créer de nouveaux agents, au sens où un agent est un programme ou une procédure.

- *L'envoi de message avec continuation* : quand un agent envoie un message, il précise à quel agent la réponse de ce message doit être envoyée. Il peut s'agir tout simplement de l'agent émetteur du message, ou d'un autre agent utilisé pour la circonstance. Si l'agent n'existe pas, il est crée spécialement à cet effet.

Aujourd'hui, l'approche la plus prometteuse, pour l'échange par messages, est indubitablement celle qui fait intervenir les états "mentaux" des agents et plus particulièrement leurs croyances et leurs intentions (Théorie des actes du discours). Ce type de message est en fait dérivé des actes du langage naturel et se prête à la modélisation de l'interaction rationnelle entre agents.

Dans le même contexte, le système Ether conçu par Ferber [1987] utilise un langage de représentation de connaissances, appelé Paladin, composé de trois couches : une couches structurelle qui s'articule autour de la notion d'entité épistémique, une couche conceptuelle fondée sur une théorie intentionnelle et une couche actancielle. L'auteur met en avant l'importance de la communication entre entités considérées comme des agents autonomes, capables de communiquer à l'aide d'un langage structuré et "doué" de motivation.

4.1.3 - Comparaison des deux moyens de communication

Alors que les systèmes d'échange par blackboard et par envoi de messages standards prolifèrent, il existe encore peu de systèmes fondés totalement sur les

acteurs. Les systèmes basés sur l'échange des informations et faisant référence aux états mentaux, font quant à eux l'objet d'intenses activités de recherche actuellement.

De la comparaison entre les deux modes d'échange d'informations, il résulte que l'échange par envoi de messages est plus facile à programmer et plus efficace dans certains cas que l'échange par mémoire commune. Dans le même cadre, Hewit et Lieberman [1983,1984] soutiennent l'idée que les systèmes à blackboard sont irréalisables pour les systèmes d'intelligence artificielle distribuée. La raison qu'ils invoquent est que; s'il existe un seul blackboard, il devient un goulot d'étranglement dès que l'échange devient important, et s'il en existe plusieurs, on en vient à des communication par envoi de messages entre blackboards.

En conséquence, il est préférable, lorsque le problème à résoudre le permet, d'utiliser l'échange par envoi de messages plutôt que l'échange par mémoire commune.

Section 5 - Conclusion

Cette section a souligné l'intérêt de mettre en oeuvre une résolution distribuée des problèmes par un groupe d'agents coopérants et a présenté les problèmes soulevés par cette approche.

Plus particulièrement, la problématique des univers multi-agents a été décrite au vu des deux principaux problèmes rencontrés dans la résolution distribuée de problème : le contrôle et les communications.

L'efficacité du contrôle du groupe d'agents coopérants nécessite la prise en compte de son organisation et la définition d'une ligne de conduite pour maintenir sa cohésion et sa cohérence tout au long de la résolution distribuée. A cet effet, un contrôle dynamique peut s'avérer difficile à mettre en oeuvre et à utiliser, mais il permet au groupe de se réorganiser lui-même en cas d'impasse.

Le choix du moyen de communication, soit par l'emploi d'une mémoire commune, soit par l'envoi de messages, est dépendant du problème. L'envoi de messages est une meilleure solution pour les systèmes largement dispersés physiquement. En revanche, le partage d'information par mémoire commune ou blackboard est conseillé lorsque les échanges d'informations sont importants. Bien entendu, il ne faut pas que ces échanges soient très importants auquel cas il y a risque de blocage du blackboard.

La réduction des informations échangées entre agents est indubitablement le point crucial de la résolution distribuée de problème. Cette réduction qui impose à chacun de raisonner à partir d'informations incomplètes, est un défi que les chercheurs ont relevé ces dernières années par une approche de raisonnement sur les connaissances et les croyances. Cette approche, basée sur des logiques non classiques, temporelles, etc. est très prometteuse non seulement pour l'intelligence artificielle distribuée, mais aussi pour la modélisation du raisonnement.

Au vu de sa problématique, la résolution distribuée de problème laisse augurer d'intenses activités de recherches. Aussi, et afin de fixer les idées, il nous a semblé utile de présenter les différentes approches de résolution distribuée, aussi bien celles qui font actuellement l'objet de recherches, que celles qui ont fait leurs preuves dans certains domaines particuliers. Ces différentes approches sont détaillées à le chapitre suivante.

Dans une deuxième section, nous présenterons quelques approches de solutions à la résolution distribuée de problèmes en y apportant une critique la plus objective possibles. Il est évident que chaque méthode s'adapte plus particulièrement à un type de cas, et que par conséquent, nous accompagnerons les approches de solutions avec un exemple reprit dans le domaine de prédilection de la méthode.

CHAPITRE 2 : Approche de solutions à la résolution distribuée de problème

Section 1 - Introduction

Après avoir situé la problématique des univers multi-agents dans le précédent chapitre, nous abordons dans celui-ci les différentes approches existantes de résolution distribuée de problème. Ces approches sont non seulement présentées et discutées par rapport aux objectifs de la résolution distribuée, mais sont aussi situées par rapport aux problématiques du contrôle et des communications entre agents, soulevées dans le chapitre précédent. Bien entendu, chacune des approches apporte des réponses partielles à cette problématique, et à notre connaissance, il n'y en a pas qui réponde à toutes les questions soulevées. C'est pourquoi ce chapitre s'attache à décrire les différentes méthodes de résolution distribuée existants, et les contextes auxquels ils se réfèrent. Il est décomposé en cinq parties :

- La première partie recense les différentes approches de solutions à la résolution distribuée que l'on trouve dans les deux formes de coopérations les plus utilisées : le partage des tâches et le partage des connaissances. A cet effet, les contextes, auxquels se réfère chaque forme de coopération, sont décrits de façon à dégager les paradigmes de résolution. L'utilité des deux formes de coopération est ensuite examinée et leur complémentarité est discutée.

- La seconde partie détaille l'approche *empirique* qui a été développée dans le domaine du contrôle aérien. Elle doit son nom à la manière empirique selon laquelle ont été proposées les différentes stratégies coopératives qu'un groupe d'avions doit suivre pour résoudre un conflit aérien. Ces stratégies sont fondées sur le choix d'une structure organisationnelle et le choix d'une politique de distribution de l'information.

- L'approche basée sur un protocole de *négociation*, issue des sociétés humaines, fait l'objet de la troisième partie. Un tel protocole permet surtout un équilibre de la charge de travail entre les agents, et évite ainsi que certains soient inoccupés alors que d'autres sont surchargés. Le contenu et la chronologie des messages sont détaillés, et ce protocole est illustré par un exemple dans le domaine du contrôle d'un atelier de fabrication.

- Une approche très connue en intelligence artificielle distribuée consiste à construire la solution d'un problème par *raffinements successifs*. Cette approche est illustrée par le modèle Hearsay II dont l'architecture a été conçue au départ pour la compréhension de la parole et utilisée par la suite pour la résolution

distribuée de problème, en particulier dans le domaine de l'évaluation distribuée de situation. Le détail de l'architecture de Hearsay II et son application à la détection distribuée, à travers un exemple concret, font l'objet de la quatrième partie.

- Finalement, la dernière partie présente une approche originale et récente visant à modéliser le *raisonnement sur les connaissances*, et les actions dans un univers multi-agents. Le but de cette approche est de développer un formalisme pour prendre en compte les connaissances et les croyances de chaque agent, en vue de concevoir des modèles d'agents capables d'agir, d'interagir, et de communiquer entre eux de manière rationnelle.

Section 2 - Approches de solutions à la résolution distribuée de problème

Malgré l'aspect de nouveauté qui s'en dégage, et l'engouement qu'il suscite de la part des chercheurs, le thème de l'intelligence artificielle distribuée n'en est pas à ses débuts, loin s'en faut. Dès les années septante, plusieurs travaux, essentiellement américains, ont été directement influencés par le technique de résolution distribuée. Depuis ces années, la résolution distribuée de problème s'est développée suivant plusieurs directions et a donné naissance à différentes méthodes de résolution. Celles-ci peuvent être dégagées en étudiant les deux formes de coopération les plus répandues dans la résolution distribuée de problème : le partage de tâches (la coopération entre les agents) et la partage de connaissances (le contrôle des différents agents).

2.1 - Partage de tâches

Le partage de tâches encore appelé partage des rôles, est une forme de coopération dans laquelle les agents s'assistent mutuellement en partageant les rôles ou les tâches nécessaires à la résolution d'un même problème, figure 2.1. Bien entendu, ce type de coopération est dirigé vers la satisfaction d'un but global, auquel les agents contribuent en satisfaisant leurs sous-buts.

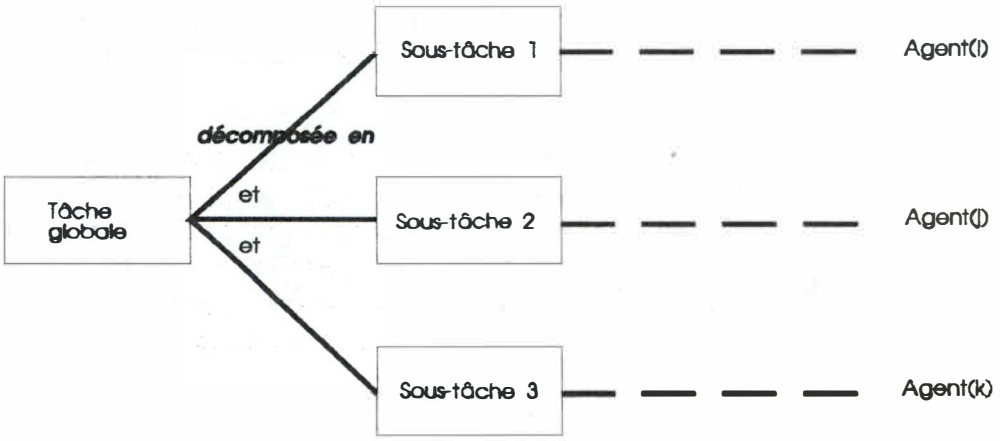

Figure 2.1 Partage de tâches

Le problème central du partage de tâches est de savoir comment les sous-tâches, ou les rôles, doivent être distribués aux différents agents participant à la résolution. En d'autres termes, quels doivent être les critères d'attribution des sous-tâches aux agents, et comment le groupe d'agents, considéré comme une entité globale, peut-il déterminer

l'agent le plus approprié pour chacune des sous-tâches ?

Ce problème d'assignation est crucial dans la résolution distribuée de problème, car il constitue un des moyens pour assurer d'une part, la cohérence et la coordination des agents, et d'autre part pour réduire l'échange d'informations entre les agents. En fait, ce problème d'assignation se pose dans deux situations différentes :

- La première situation est caractérisée par le fait que l'agent le plus approprié pour une tâche donnée est connu à priori, et c'est le cas lorsque le groupe arrive à maintenir ses connaissances sur les capacités et les connaissances de chaque agent. Toutefois, dans bien des cas, cela ne suffit pas, et pour une tâche donnée, plusieurs agents peuvent posséder les compétences requises, et se trouver par conséquent en compétition. Aussi, une approche de solution consiste à déterminer empiriquement différentes structures organisationnelles et à les tester afin d'évaluer leur incidence sur la performance du groupe. Cette *approche empirique* a été testée dans le contrôle aérien et fait actuellement l'objet de tentatives d'applications pour la coopération entre robots doués de capacités de raisonnement.

- Dans la deuxième situation, aucun agent approprié pour une tâche donnée n'est connu à priori. Dans ce cas, l'approche basée sur la *négociation*, déjà évoquée dans le chapitre 1, peut être utilisée pour donner au groupe une structure organisationnelle et résoudre le problème d'assignation. Dans cette approche, le problème global à résoudre est décomposé, par l'un des agents, en différents sous-problèmes et soumis à d'autres agents selon un protocole de négociation prédéfini. Suite à cette négociation, un contrat d'agrément est établi entre l'agent demandeur d'aide, et l'agent capable de l'aider. Bien entendu, ce dernier est choisi pour ses capacités et ses compétences, et il n'est évidemment pas désigné à priori.

En résumé, deux approches peuvent être utilisées pour la répartition des tâches : l'approche empirique et l'approche basée sur le mécanisme de la négociation. La première est à utiliser lorsque sont connus à priori les capacités et les compétences des agents ainsi que les rôles qu'ils sont susceptibles de tenir dans la résolution distribuée. La seconde, en revanche, est à utiliser dans les environnements qui requièrent des agents hautement spécialisés et dont le groupe ne connaît pas à priori le domaine de compétence.

2.2 - Partage de connaissances

Dans cette forme de coopération par partage de connaissances schématisées par la figure 2.2, les agents s'assistent mutuellement par des échanges de connaissances, de façon à permettre à chacun d'eux d'affiner sa propre connaissance du monde et de construire progressivement sa propre solution. Cependant, en général, l'interaction entre les agents est dirigée vers l'échange d'informations et non vers la satisfaction d'un but global, comme c'est le cas dans le partage de tâches.

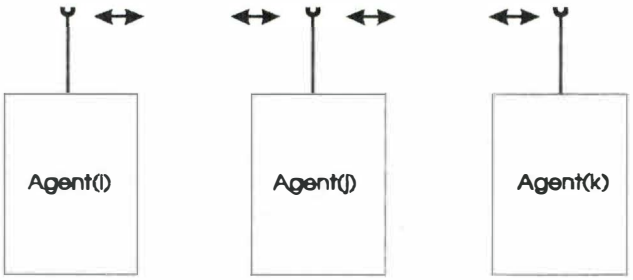

Figure 2.2 Partage des connaissances

Bien entendu, le *contenu* des messages échangés est le point central de cette forme de coopération, et il dépend du type de problème à résoudre.

- Dans certains domaines d'applications et en particulier pour l'analyse de situation, la conception et la planification des actions en général, la solution à un problème ne peut s'obtenir que par des agrégations successives de résultats partiels basés sur différentes perspectives du problème global. Dans ce cadre, les chercheurs en intelligence artificielle ont développé l'approche dite par *raffinements successifs* qui permet aux agents de s'échanger des tentatives de solutions partielles.

- Toutefois, dans la plupart des applications, les agents doivent s'adapter au problème qu'ils ont à résoudre, et ne s'échangent pas toujours les mêmes connaissances suivant la même méthode comme le fait l'approche par *raffinements successifs*. En conséquence, dans ces cas, l'agent doit être conçu comme une entité physique autonome, percevant et agissant sur l'environnement extérieur. Son interaction avec les autres agents doit être fondée sur la planification des actions, sur le contrôle d'exécution et surtout sur le raisonnement sur autrui, une fois celui-ci modélisé. C'est dans ce but que l'approche appelée *raisonnement sur les connaissances* s'attache à concevoir des agents autonomes capables d'interagir de manière rationnelle pour la résolution en commun d'un même problème. Dans cette approche, l'échange d'information entre les agents portent sur des connaissances qui permettent à chaque agent de mettre à jour son modèle d'autrui et de l'environnement. Ceci a pour conséquence d'affiner son raisonnement à propos de ses futures actions, en tenant compte du comportement des autres agents.

En résumé, l'approche de résolution distribuée par raffinements successifs s'applique à des domaines d'évaluation de situations "routinières", où des agents non-autonomes ont besoin de s'échanger des résultats partiels. L'approche basée sur le raisonnement, quant à elle, s'applique à des agents autonomes capables de s'adapter à toute nouvelle situation. Bien entendu, à la différence des autres approches, l'adaptabilité de cette dernière lui permet d'utiliser de façon complémentaire les deux formes de coopération, le partage de tâches et le partage de connaissances.

Ainsi, de tous les paradigmes de résolution distribuée développés à ce jour, quatre approches peuvent être dégagées : l'approche empirique, l'approche utilisant un protocole de négociation, l'approche de génération de solution par raffinements successifs, et enfin, l'approche basée sur le raisonnement à partir des connaissances et des actions. Dans la suite de ce chapitre, chacune d'elles est détaillée de façon critique en mettant l'accent sur les réponses apportées aux problématiques et aux objectifs de la résolution distribuée de problème. De plus, l'exemple d'une application concrète est détaillée pour trois d'entre elles, de façon à montrer l'applicabilité du paradigme utilisé. L'approche basée sur le raisonnement est, quant à elle, encore au stade d'étude théorique, et fera l'objet d'une étude spécifique.

Section 3 - Approche empirique

3.1 - Contexte de mise en oeuvre de l'approche empirique

Cette approche a été conçue pour le contexte particulier du contrôle de trafic aérien, étant donné que la détection et la résolution d'un conflit aérien est un problème de planification distribuée multi-agents visant à déterminer quel sera le rôle de chacun des agents, c'est à dire des avions. Plus précisément, les avions concernés par un éventuel conflit, doivent déterminer lesquels d'entre eux sont appropriés pour exécuter les sous-tâches induites par ce conflit, c'est à dire la détection du conflit, l'élaboration d'un nouveau plan de vol permettant d'éviter le conflit, la transmission de ce nouveau plan de vol aux avions voisins afin de les avertir, et enfin l'exécution de ce nouveau plan. En d'autres termes, le problème est de sélectionner selon des critères qui restent à déterminer, l'agent le mieux adapté à toutes ces sous-tâches ou à chacune d'elles.

C'est essentiellement à la Rand Corporation que les travaux relatifs à la simulation du contrôle aérien, par une approche utilisant les techniques d'intelligences artificielle distribuée, ont été effectués. Les chercheurs de cette institution, et plus particulièrement Cammarata, McArthur, Steeb [1983] ont essayé d'approcher le domaine du contrôle aérien de façon originale et empirique. A cet effet, ils ont développé des concepts permettant de décrire différentes stratégies coopératives qu'ils ont testées, en les expérimentant sur différents groupes d'agents et en observant leurs effets sur la performance du groupe.

Chacune de ces stratégies coopératives porte sur le choix d'une structure organisationnelle et le choix d'une politique de distribution de l'information. Les auteurs de la méthode s'en sont fixés une proche de la réalité et ont testé quatre structures organisationnelles. Parmi celles-ci, trois centralisent toutes les sous-tâches du contrôle aérien au niveau d'un même agent et en ce sens, chacune des trois organisations permet au groupe de sélectionner l'agent qui doit exécuter toutes les sous-tâches du conflit. Ainsi, la première organisation sélectionne cet agent suivant des règles préétablies qui sont essentiellement basées sur des informations relatives aux autres avions telles que la vitesse, la position, et la destination. La seconde organisation, quant à elle, sélectionne l'agent le moins contraint (vis à vis du carburant restant, de l'horaire ou des conflits potentiels) pour exécuter toutes les sous-tâches du conflit. La troisième, en revanche, sélectionne pour toutes les sous-tâches, l'agent le mieux informé du groupe. La quatrième structure enfin, est basée sur le partage des tâches et sélectionne l'agent le mieux informé du groupe pour élaborer le nouveau plan de vol, et l'agent le moins contraint du groupe pour l'exécuter.

A part la première organisation qui est basée sur des règles préétablies, les trois autres structures sont contrôlées dynamiquement. En effet, pour ces trois structures, la hiérarchie est mise en place suivant les contraintes et les connaissances de chaque participant.

Les auteurs de ces stratégies coopératives ont simulé la tâche de trafic aérien dans une portion d'espace rectangulaire dans laquelle chaque avion peut pénétrer à

n'importe quel moment, soit par l'une des deux entrées fixées, soit à partir de deux aéroports. L'objectif de chaque avion est de traverser l'espace aérien jusqu'à sa destination finale (sortie fixée ou aéroport) sans violer les règles de séparations entre avions. Autrement dit, le problème global du contrôle aérien est un problème de planification distribuée, qui est résolu lorsque tous les avions atteignent leur destination sans violer les normes de séparation minimale. Ces normes, imposées par le code de l'aviation civile, sont de mille pieds pour la séparation verticale et de trois mille pieds pour la séparation horizontale.

Il est important maintenant de détailler davantage les quatre différentes structures organisationnelles, pour savoir comment elles répondent aux deux problèmes de la résolution distribuée : le contrôle du groupe et les communications entre agents. Les effets de ces différentes stratégies coopératives sur la performance du groupe sont ensuite comparées entre elles.

3.2 - Structures organisationnelles et distribution de l'information dans le contrôle de trafic aérien

Dans le contrôle de trafic aérien, la détection et la résolution de conflits sont les tâches que doit résoudre le groupe d'avions concernés par le conflit aérien. Ces tâches peuvent être elles-mêmes décomposées en plusieurs sous-tâches que les agents doivent exécuter suivant leurs qualifications. Ces sous-tâches reprennent : la collecte d'informations, l'élaboration d'un nouveau plan de vol pour éviter le conflit, et l'exécution de ce nouveau plan.

Pour chacune de ces sous-tâches, les agents sont plus ou moins appropriés suivant l'état de leurs connaissances actuelles, leur contraintes et leurs disponibilités. En effet, la sous-tâche d'*élaboration d'un nouveau plan* nécessite de connaître les intentions des autres agents impliqués, alors que la sous-tâche d'*exécution de ce nouveau plan* exige moins de contraintes et plus de disponibilité de la part de l'agent exécutant. Aussi, le problème qui se pose au groupe d'avions concernés par le conflit est celui de l'assignation de chacune des sous-tâches composant la tâche globale du conflit aérien.

C'est dans le but de montrer l'impact de l'assignation optimale des rôles, sur la performance globale du groupe, que Cammarata, McArthur, et Steeb [1983] ont développé leurs stratégies coopératives. Ils se sont aussi attachés à montrer l'impact d'une assignation complète et consistante. Rappelons qu'une assignation est dite complète si *tous* les rôles, nécessaires à une tâche donnée, sont attribués aux agents, alors que l'assignation est dite consistante si aucune tâche n'est allouée aux agents inutiles.

En fait, ces stratégies coopératives comprennent la politique choisie en matière de communication entre agents, les structures organisationnelles et le degré de coopération. C'est ainsi que dans le contexte du contrôle aérien, les avions sont supposés être totalement coopérant, et la politique de distribution de l'information est

supposée transmise explicitement d'un avion à un autre, sans nécessiter de demande préalable, et sans qu'il y ait ni accusé de réception, ni répétition. Ces choix paraissent convenables pour une simulation puisqu'on suppose que les transmissions se font sans erreurs. Les structures organisationnelles, quant à elle, consistent à sélectionner selon un critère donné le ou les agents appropriés pour toutes les tâches induites par le conflit ou pour chacune d'elles. Ces structures organisationnelles sont abordées ci-après.

3.2.1 - Sélection par Convention Partagée

Dans cette organisation, les avions utilisent uniquement la vitesse, la position et la destination de chacun d'eux pour décider qui doit élaborer un nouveau plan de vol et l'exécuter pour éviter le conflit. En fait, les avions utilisent implicitement, pour cette décision, un ensemble de règles préétablies et acceptées par tous dans le but de minimiser les communications. Dans ce cas, on dit que les avions partagent la même convention pour cette décision. La figure 2.3 montre une séquence prototype de tâches et d'échange d'informations entre deux avions Av1 et Av2 utilisant cette convention. Dans ce cas de figure, Av2 suspecte en premier un conflit avec Av1 et sait, par exemple, au vu des informations dont il dispose sur les autres avions, que c'est à lui de changer son plan de vol. Selon la convention préétablie Av2 envoie donc son plan de vol à Av1 qui compare alors ce plan de vol avec le sien pour savoir s'il y a conflit. Dans l'affirmative et selon la convention préétablie, Av1 élabore un nouveau plan et le transmet aux avions voisins pour les tenir au courant, avant de l'exécuter.

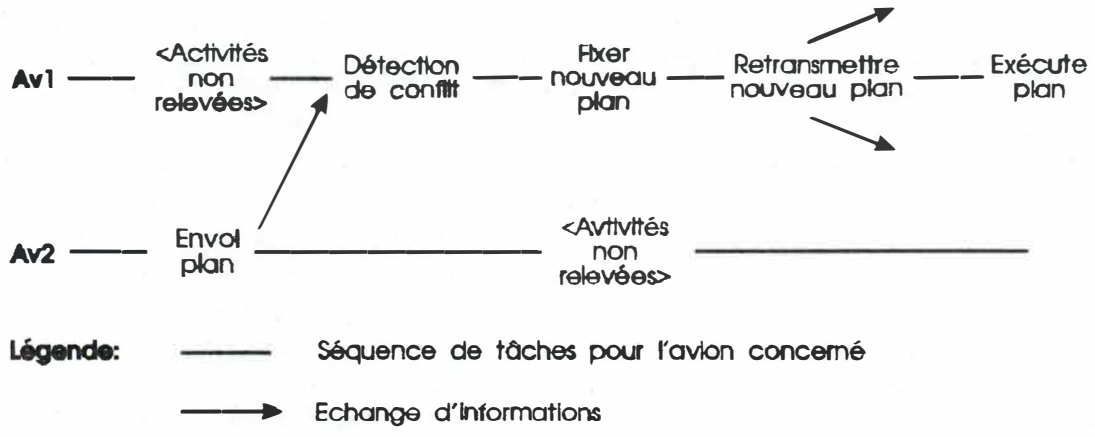

Figure 2.3 Séquence de tâches pour la sélection par convention partagée

Cette organisation présente l'avantage d'avoir une charge de communication réduite, cependant, l'avion sélectionné pour modifier son plan de vol n'est pas forcément le plus approprié pour ce rôle, dans la mesure où il peut engendrer de

nouveaux problèmes de séparation. En fait, cette stratégie a servi de référence aux auteurs pour les trois prochains modes de sélection considérés comme plus "intelligents".

3.2.2 - Sélection de l'agent le moins contraint

Dans cette organisation, chaque avion impliqué dans un conflit potentiel transmet son facteur de contraintes aux autres avions. Ce facteur est une agrégation de plusieurs termes tels que le nombre d'avions voisins, le fuel restant, la distance restant à parcourir jusqu'à la destination finale, etc.. Suite à l'échange de ce facteur, il revient à l'agent le moins contraint du groupe d'élaborer un nouveau plan, et de l'exécuter pour résoudre le conflit. Ceci s'explique par le fait que l'agent le moins contraint est capable d'agir sans engendrer de nouveaux conflits.

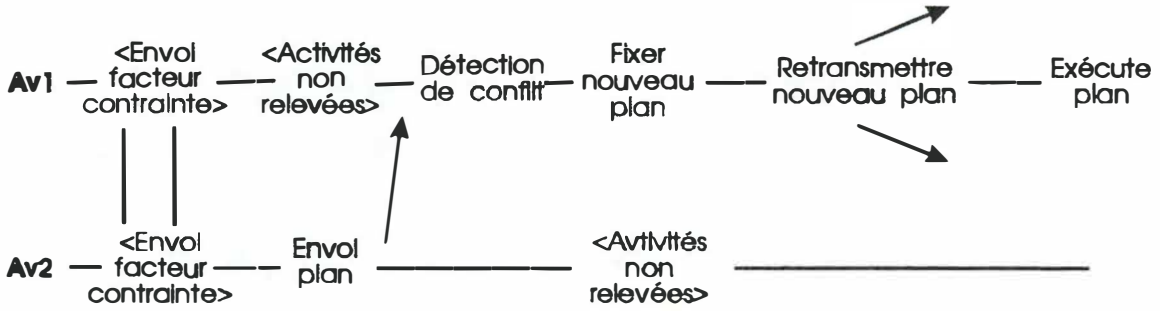

Figure 2.4 Séquence de tâches pour la sélection de l'avion le moins contraint

La figure 2.4 donne une idée des séquences de sous-tâches et de communications entre deux avions Av1 et Av2 utilisant cette stratégie. Dans ce scénario, l'avion Av1 est supposé avoir le facteur de contraintes le plus faible, et par conséquent, il lui revient, suite à l'échange de facteurs entre les deux avions, d'élaborer un nouveau plan et de le transmettre aux avions voisins avant de l'exécuter.

Cette stratégie sélectionne donc l'agent le moins contraint, c'est à dire l'agent possédant le plus de degré de liberté, pour manoeuvrer et ainsi résoudre le conflit tout en évitant d'engendrer de nouveaux problèmes de séparation entre avions. Il est à noter que bien que cette stratégie soit plus difficile à mettre en oeuvre que la précédente, elle permet de choisir d'une façon plus appropriée l'avion qui doit changer son plan de vol actuel. En revanche, elle impose une plus grande charge de communication due à l'échange du facteur de contraintes.

3.2.3 - Sélection de l'agent le mieux informé du groupe

Cette méthode sélectionne l'agent qui possède le plus de connaissances sur les

intentions des autres pour élaborer un nouveau plan de vol. En effet, un tel agent est capable de tenir compte des interactions possibles entre ses intentions et celles des autres agents afin de déterminer le nouveau plan de vol pour résoudre le conflit. En conséquence, les agents dont les intentions sont connues par les autres doivent s'abstenir d'élaborer de nouveaux plans et de les suivre, car ils risquent de rendre la tâche de résolution de conflit plus difficile.

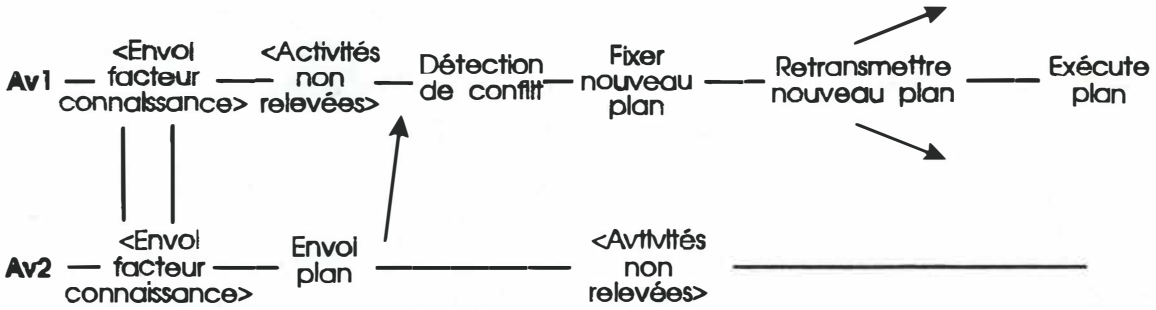

Figure 2.5 Séquence de tâches pour la sélection de l'avion qui a le plus de connaissances

La figure 2.5 montre une séquence de tâches et d'échange d'informations entre deux avions Av1 et Av2. Ces deux avions s'échangent un *facteur de connaissances*, lorsqu'ils se sentent concernés par un conflit potentiel, pour pouvoir déterminer lequel d'entre eux est le mieux informé. Ce facteur reflète ce que chacun sait sur les autres agents, en particulier leur position, leurs performances et leurs intentions.

Dans l'exemple donné en figure 2.5, l'avion Av1 est supposé connaître les intentions de Av2 et, par conséquent, il lui revient d'élaborer un nouveau plan de vol et de l'exécuter pour résoudre le conflit. Inversement, l'avion Av2 doit s'abstenir de prendre toute initiative et doit conserver son plan de vol courant, car ses intentions sont connues de Av1.

Cette stratégie sélectionne donc l'agent le mieux informé du groupe, pour élaborer un nouveau plan de vol. En conséquence, un plan de vol ne générant pas de nouveaux conflits doit en résulter.

Il convient de noter que cette organisation, comme les deux précédentes, présente l'inconvénient de centraliser toutes les sous-tâches au niveau d'un seul agent. Une méthode pour pallier cet inconvénient consiste à distribuer les différentes sous-tâches aux agents suivant leurs compétences et leurs capacités.

3.2.4 - Structure organisationnelle basée sur le partage des tâches

L'inconvénient des trois précédents modes de sélection réside dans l'assignation de toutes les tâches au même agent. En effet, c'est le même agent qui doit élaborer le nouveau plan, le transmettre à ses voisins, et l'exécuter. Or, bien que cet agent puisse

s'avérer globalement le meilleur pour l'ensemble des tâches, il est rarement le plus efficace pour chacune d'elles. Une façon d'y remédier est de partager les tâches entre les différents agents.

Le but de cette nouvelle organisation est de remédier à l'assignation de tous les rôles à un même agent comme le faisaient les précédents modes de sélection. C'est dans ce but que les sous-tâches que constituent le contrôle aérien sont partagées s'abstenir d'élaborer de nouveaux plans et de les suivre, car ils risquent de rendre la tâche de résolution de conflit plus difficile.

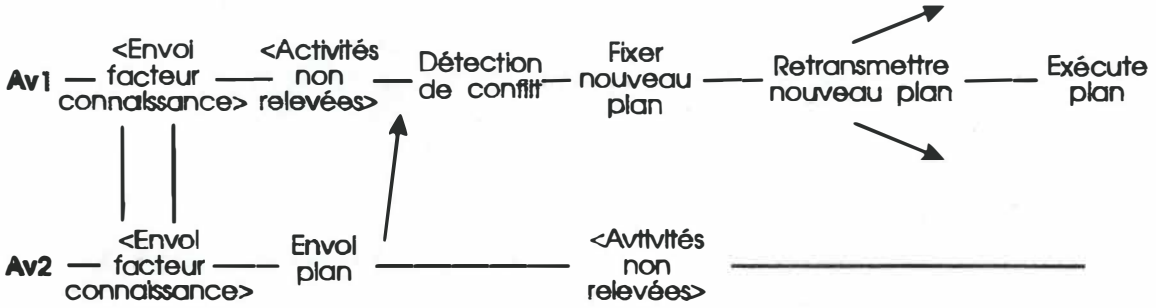

Figure 2.5 Séquence de tâches pour la sélection de l'avion qui a le plus de connaissances

La figure 2.5 montre une séquence de tâches et d'échange d'informations entre deux avions Av1 et Av2. Ces deux avions s'échangent un *facteur de connaissances*, lorsqu'ils se sentent concernés par un conflit potentiel, pour pouvoir déterminer lequel d'entre eux est le mieux informé. Ce facteur reflète ce que chacun sait sur les autres agents, en particulier leur position, leurs performances et leurs intentions.

Dans l'exemple donné en figure 2.5, l'avion Av1 est supposé connaître les intentions de Av2 et, par conséquent, il lui revient d'élaborer un nouveau plan de vol et de l'exécuter pour résoudre le conflit. Inversement, l'avion Av2 doit s'abstenir de prendre toute initiative et doit conserver son plan de vol courant, car ses intentions sont connues de Av1.

Cette stratégie sélectionne donc l'agent le mieux informé du groupe, pour élaborer un nouveau plan de vol. En conséquence, un plan de vol ne générant pas de nouveaux conflits doit en résulter.

Il convient de noter que cette organisation, comme les deux précédentes, présente l'inconvénient de centraliser toutes les sous-tâches au niveau d'un seul agent. Une méthode pour pallier cet inconvénient consiste à distribuer les différentes sous-tâches aux agents suivant leurs compétences et leurs capacités.

3.2.4 - Structure organisationnelle basée sur le partage des tâches

L'inconvénient des trois précédents modes de sélection réside dans l'assignation de toutes les tâches au même agent. En effet, c'est le même agent qui doit élaborer le nouveau plan, le transmettre à ses voisins, et l'exécuter. Or, bien que cet agent puisse s'avérer globalement le meilleur pour l'ensemble des tâches, il est rarement le plus efficace pour chacune d'elles. Une façon d'y remédier est de partager les tâches entre les différents agents.

Le but de cette nouvelle organisation est de remédier à l'assignation de tous les rôles à un même agent comme le faisaient les précédents modes de sélection. C'est dans ce but que les sous-tâches que constituent le contrôle aérien sont partagées suivant les connaissances, les compétences et les capacités de chaque participant au conflit. Il revient donc à l'avion le moins contraint du groupe d'exécuter le nouveau plan car c'est lui qui a le plus de degrés de liberté pour le faire dans les meilleures conditions et sans générer de nouveaux conflits. Par contre, c'est à l'avion le mieux informé du groupe que revient le rôle d'élaborer le nouveau plan, car cet avion connaît les performances, les positions et intentions des autres agents, et peut par conséquent élaborer et évaluer les conséquences d'un nouveau plan de vol.

Bien entendu, contrairement aux précédentes stratégies de coopération, où un seul tour de négociation suffisait pour déterminer l'agent qui planifie et exécute le nouveau plan de vol, la présente stratégie nécessite, quant à elle, deux tours de négociation successifs : le premier définit l'agent qui fixe le nouveau plan de vol, et le second détermine l'agent qui exécute celui-ci.

La figure 2.6 montre comment une telle stratégie de coopération peut se dérouler entre deux avions Av1 et Av2. Tout d'abord, ils s'échangent successivement les facteurs de contrainte et de connaissance. Ensuite, l'avion Av1 supposé posséder le facteur de connaissance le plus élevé, fixe le nouveau plan. Enfin, l'avion Av2 supposé être moins contraint, exécute ce nouveau plan. Ainsi, dans cette distribution des rôles, Av1 est le planificateur et Av2 est l'acteur.

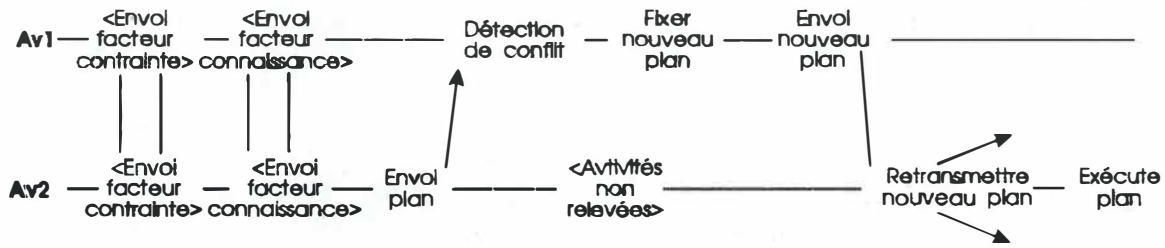

Figure 2.6 Séquence de tâches pour la sélection par partage des rôles

3.3 - Comparaison des quatre structures organisationnelles

Il est clair que la méthode de sélection par partage des tâches conduit à une meilleure distribution des rôles et à une optimisation locale de chacune des sous-tâches, alors que les trois structures basées sur la centralisation des rôles, conduisent à une optimisation globale. Les auteurs n'ayant pas testé l'organisation basée sur le partage des tâches, il est difficile de conclure à propos des deux méthodes d'optimisation. Toutefois, il convient de remarquer qu'au stade actuel des recherches, l'organisation basée sur le partage des tâches pose d'énormes problèmes de coordination, de cohérence et de surcharge de communications entre les agents. De plus, cette organisation a l'inconvénient de dépendre des canaux de transmission et de leur fiabilité, car elle nécessite deux tours de négociation et induit donc une charge de communication plus importante que les trois autres structures.

Seules les trois premières structures furent implémentées par leurs auteurs, et la sélection fondée sur *l'agent moins contraint* s'est avérée la meilleure. En effet, les résultats comparatifs ont montré que, malgré l'utilisation d'un nombre élevé de messages, pour déterminer l'agent planificateur et maintenir la consistance après la planification, la sélection de l'agent le moins contraint est celle qui convient le mieux, non seulement à toute situation complexe, mais aussi lorsque les tâches sont très difficiles. La sélection de *l'agent le mieux informé du groupe*, quant à elle, convient mieux aux situations moins complexes, mais très difficiles. Pour les auteurs, la complexité est évaluée par le nombre de solutions possibles au problème du conflit aérien.

En résumé, cette approche empirique convient pour la distribution des rôles entre agents, dont les capacités et les compétences sont, ou peuvent être, préalablement déterminées. C'est donc seulement après avoir testé les performances du groupe que les structures organisationnelles peuvent être dégagées.

En revanche, si les compétences et les capacités des agents ne sont pas connues à priori, le partage des rôles peut se faire par une approche basée sur le mécanisme de la négociation entre un demandeur de services et ses éventuels contractants. Ce mécanisme de négociation est détaillé ci-après.

Section 4 - Approche basée sur la négociation

Partant d'une métaphore des sociétés humaines, Smith [1980] a élaboré un protocole de négociation entre systèmes intelligents pour un partage équitable des tâches entre eux. Ainsi, lorsqu'un système ne parvient pas à exécuter seul sa tâche, il demande l'aide des autres agents. Il s'en suit une négociation entre le demandeur et les agents sollicités, selon un protocole prédéfini issu de la notion de contrat semblable à celle existant dans les sociétés humaines.

Cette approche répond en partie à la problématique de la résolution distribuée de problème soulevée dans le chapitre 1. Plus particulièrement, ce sont les communications qui sont ici abordées, puisque l'échange par message obéit à un protocole dont le contenu est à la base de cette manière d'aborder la résolution distribuée de problème. En effet, dans ce type d'approche, un agent manager envoie un message d'annonce de tâche à des agents susceptibles de l'aider (coopération totale). Ces agents font des offres suivant leur disponibilité et leurs compétences. Le manager choisit alors parmi ces offres celles qui lui conviennent, et envoie un message d'attribution aux contractants qui émettent à leur tour un accusé de réception. Suite à cet échange, une organisation définie spécialement pour la tâche donnée prend place, et son contrôle d'évolution se fait dynamiquement par l'échange de rapports entre manager et contractants.

Le contenu et la chronologie des messages échangés dans le cadre de cette négociation, ainsi que l'application de celle-ci à un ensemble d'usines de fabrication, sont maintenant détaillées.

4.1 - La négociation comme métaphore à la résolution distribuée de problème

Le réseau à contrats (contract net) de Smith et Davis [1981] utilise la métaphore de négociation entre demandeur et contractants. Dans ce réseau, des agents autonomes, appelés noeuds, s'échangent des messages pour pouvoir se partager les tâches. Aussi est-il important de caractériser ces noeuds pour différencier le mandeur des contractants, et de spécifier la chronologie et le contenu des messages échangés.

Pour cela, trois classes de noeuds sont identifiées, quelle que soit la transaction :

- le noeud *manager* qui identifie la tâche à effectuer, et demande de l'aide aux autres noeuds, car il lui est impossible de la résoudre tout seul, figure 2.7.
- les noeuds qui offrent leurs services pour exécuter la tâche, figure 2.8. Ils sont appelés par les auteurs *enrichisseurs* (bidders),
- le noeud *contractant*, dont l'offre a été acceptée par le manager, figure 2.9

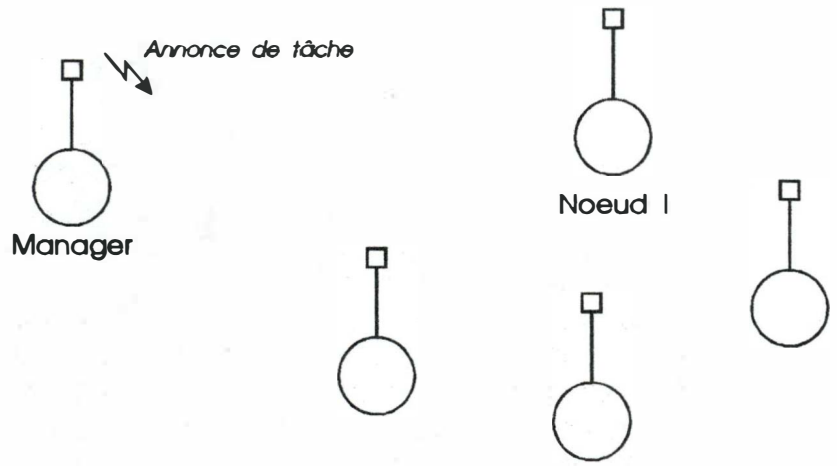

Figure 2.7 Annonce de tâche par le manager

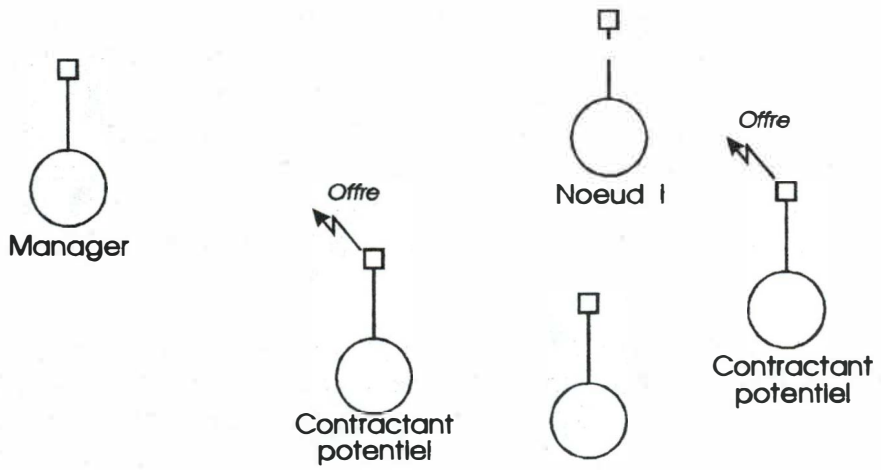

Figure 2.8 Propositions d'offres par les contractants potentiels

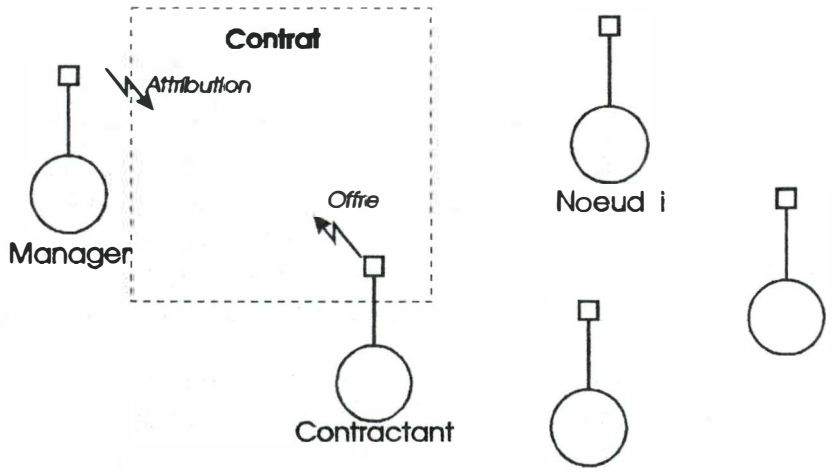

Figure 2.9 Lien contractuel entre manager et contractant

Les échanges de messages s'effectuent, entre ces trois sortes de noeuds, selon la chronologie suivante :

- le manager envoie une *annonce de tâche* décrivant la tâche à effectuer et les critères que doivent satisfaire les noeuds sollicités pour pouvoir y répondre en faisant une offre de service,
- les noeuds qui offrent leurs services, lui proposent des *offres* pour annoncer leur disponibilité ainsi que leur aptitude à effectuer la tâche,
- le message *d'attribution* du manager au noeud choisi, établit celui-ci comme le contractant pour la tâche,
- le contractant émet un *accusé de réception* suite à l'attribution, pour accepter ou rejeter celle-ci,
- le contractant lorsqu'il a accepté la tâche, l'exécute et envoie des *rapports* au manager pour annoncer l'état ou la fin de la tâche,

En outre :

- le manager peut émettre un *message d'interruption* au contractant pour mettre fin au contrat de façon prématurée,
- et les noeuds peuvent émettre une *annonce de noeud disponible* quand c'est le cas.

Un tel protocole est aujourd'hui appliqué dans bien des domaines concrets comme le montre l'exemple d'application abordé ci-après.

4.2 - Exemple d'application du protocole de négociation dans des usines de fabrication

YAMS (Yet Another Manufacturing System), conçu par Parunak [1985] est un système de contrôle dans un environnement de fabrication d'automobiles qui utilise les techniques de l'intelligence artificielle distribuée. Le choix de ces techniques est lié, bien entendu, aux différentes contraintes qui existent dans ce type d'environnement. En effet, comme les différentes opérations effectuées dans chaque usine doivent être contrôlées en temps réel et que, dans ce cas, chaque unité et, dans la plupart des cas, chaque machine nécessite sa propre unité centrale, le système YAMS est conçu de façon à coordonner les différentes unités centrales entre elles afin de gérer les actions de chaque unité de production. De plus, YAMS doit être capable de contrôler un ensemble d'usines largement dispersées; il est de ce fait un *système distribué*.

Par ailleurs, le problème de contrôle dans une usine peut être considéré comme une recherche à travers un espace dont les dimensions incluent les équipements disponibles sur le site, les produits à fabriquer et les ressources disponibles telles que le temps, les stocks, l'espace de stockage, etc.. Dans les usines traditionnelles, le logiciel intègre à la fois la connaissance de ces paramètres et les différentes décisions les concernant, réduisant ainsi considérablement l'espace de recherche. En général cependant, ce logiciel est façonné pour une usine et une ligne de produits, ce qui a pour conséquence de rendre coûteuse toute opération de modification. Par contre, dans YAMS, le logiciel est le même sur chaque site, car les caractéristiques propres à chaque usine sont enregistrées dans des bases de données locales et non dans le logiciel lui-même. En outre, YAMS doit traiter la complexité totale de l'espace-problème, et prendre des décisions en temps réel. Les techniques traditionnelles d'optimisation s'étant révélées trop lentes, YAMS *utilise les techniques d'intelligence artificielle* et modélise une entreprise comme hiérarchie de cellules, ou de groupes de machines.

Actuellement, YAMS est appliqué à une firme automobile schématisée en figure 2.10 où la hiérarchie comprend au niveau supérieur trois usines, deux usines de fabrication, de carrosserie et de moteurs, et une usine d'assemblage. Chaque usine est ensuite décomposée en unités fonctionnelles. L'usine moteurs par exemple est décomposée suivant les systèmes flexibles d'assemblage (SFA) qui la composent. Enfin, chaque noeud (usine, SFA ou machine quelconque) comme une unité munie de sa librairie décrivant ce qu'il sait faire. Cette unité est en outre capable de négocier avec d'autres noeuds pour un éventuel partage des tâches. A titre d'exemple, supposons que "l'usine moteurs" est contractante pour exécuter la tâche *faire moteur*. Consultant sa base de données locale, elle constate que la première étape pour faire un moteur est la tâche *faire bloc moteur*. Ne sachant exécuter elle-même cette tâche, elle va alors diffuser une annonce de tâche. En réponse à cette annonce, les systèmes flexibles "blocs moteurs" et le systèmes flexible "de secours" qui savent conduire de telles tâches lui proposent des offres. L'offre du système flexible "blocs moteurs" est considérée comme recevable car ce système est déjà configuré pour les blocs moteurs, et en conséquence, le noeuds "usine moteurs" lui attribue la tâche.

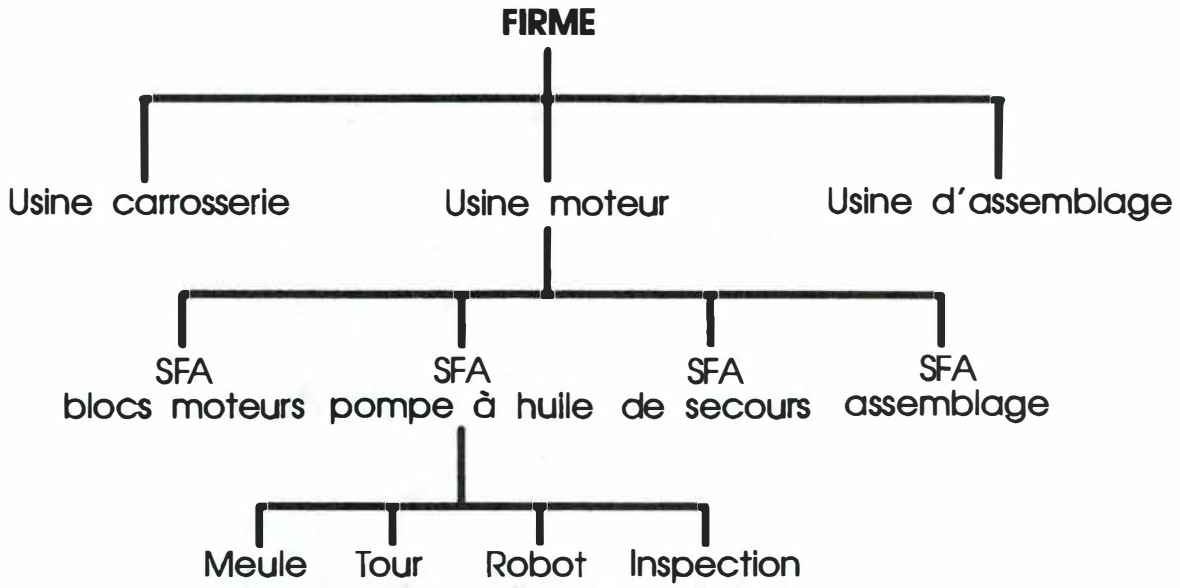

Figure 2.10 Schéma d'organisation d'une firme automobile comme instance possible de YAMS

S'il s'avère que le système flexible "blocs moteurs" devienne indisponible, par exemple à la suite d'un dysfonctionnement, la tâche sera alors attribuée au système flexible "de secours", permettant ainsi de réparer le système flexible en panne et cela sans modification du logiciel.

Deux points importants sont à relever pour cette approche. Le premier concerne la facilité de mise en oeuvre du protocole de négociation, car les classes de noeuds, la chronologie et le contenu que doivent avoir les messages échangés, sont prédéfinis avec précision. Le deuxième point concerne la facilité d'intégration d'un tel protocole dans des applications concrètes où les tâches sont dispersées géographiquement comme le montre l'application donnée en exemple.

Cependant, toutes les tâches auxquelles doit faire face le concepteur d'un système de résolution distribuée, ne sont pas toujours dispersées géographiquement, et il peut arriver que la décomposition du problème global soit plutôt fonctionnelle. Dans ce cas, la solution est généralement obtenue par raffinements successifs, au sens où chaque agent contribue à la solution en infirmant, en confirmant, en critiquant ou en remplaçant les hypothèses des autres agents. Cette approche de solution qui se construit graduellement par l'apport de solutions partielles, fait l'objet de la section suivante.

Section 5 - Approche de résolution par raffinements successifs

Dans ce type d'approche orientée vers le partage des connaissances, plusieurs agents appelés experts coopèrent d'une façon totale à la résolution d'un même problème, en générant et en échangeant des tentatives de solutions partielles. Ces solutions sont partielles car elles sont basées sur la vue locale et limitée que chacun des agents possède du problème global. En outre, ces solutions ne sont que des tentatives dans la mesure où elles peuvent être infirmées ou confirmées par d'autres agents. L'avantage d'une telle approche, est que les agents, en échangeant leurs solutions partielles, potentiellement incomplètes, imprécises voire inconsistantes, peuvent éventuellement converger par raffinements successifs vers une solution valable pour le groupe d'agents concernés. Une telle approche, particulièrement efficace dans le domaine d'évaluation distribuée de situation, utilise le blackboard comme moyen de communication et de contrôle pour les différents experts coopérants qui sont organisés une fois pour toutes, dès la conception du système. Enfin, il convient de signaler que, dans cette approche, les différents experts sont d'accord pour coopérer à l'élaboration d'une solution globale et, en ce sens, la coopération entre les experts est totale.

D'une façon plus précise, le concept de blackboard, tableau noir, permet à chaque expert participant à la résolution de contribuer à la solution en infirmant, en acceptant, en critiquant ou en remplaçant les hypothèses déjà inscrites sur le blackboard. La solution globale se construit donc par combinaison d'interprétations partielles dérivées de connaissances et de compétences diverses. Autrement dit, la solution globale est obtenue de manière itérative. Bien entendu, la blackboard en tant que moyen de communication impose un certain protocole et exige un certain contenu des informations échangées. Ce protocole et ce contenu sont détaillés ci-dessous dans le modèle Hearsay II.

Il convient de préciser aussi que le fait d'utiliser le blackboard n'interdit nullement l'utilisation de messages comme moyen de communication et certains concepteurs allient les deux, en développant des systèmes distribués munis chacun de leur blackboard. De cette façon, le blackboard sert à l'échange local entre les experts d'un même système, et l'échange par messages est utilisé entre les différents systèmes. Une application de cette méthode à la détermination des signaux de véhicules est présentée ci-après. Auparavant, nous décrivons le modèle Hearsay II, initialement conçu pour la compréhension de la parole, et dont le blackboard est issu.

5.1 - Modèle du système Hearsay II

Hearsay II est un système conçu au départ pour la compréhension de la parole

delà de l'application, le concept mis en oeuvre s'est révélé très utile pour définir la structure d'un noeud pour la résolution distribuée d'un problème. D'ailleurs, il a été utilisé dans de nombreuses applications où l'interprétation est issue de différentes sources de connaissances. Dans ce qui suit, le modèle et l'architecture d'Hearsay II sont détaillés, et suivis par l'utilisation de ce modèle à une application d'interprétation distribuée à la surveillance de trafic de véhicules.

5.1.1 - Le modèle

Dans Hearsay II, une évaluation de situation se construit par combinaisons d'interprétations partielles dérivées à partir de connaissances diverses. Chaque domaine de connaissances est représenté par un module indépendant appelé *source de connaissances*. Le but de ces modules, autre appellation pour le terme 'agent', est d'interagir de manière coopérative et compétitive pour construire la solution d'un problème donné. Pour l'application d'Hearsay II à la compréhension de la parole, ces sources de connaissances couvrent des connaissances phonétiques, syntaxiques, acoustiques et sémantiques.

L'interaction des sources de connaissances, dans le modèle Hearsay II, est basée sur l'obtention d'une solution de manière itérative sur le blackboard. Cette solution dirigée par les résultats partiels est appelée "génération d'hypothèses et test", car une itération implique la création d'une hypothèse sur la solution, suivie d'un test sur sa plausibilité.

Pour la génération d'hypothèses, les sources de connaissances utilisent des connaissances à priori sur le problème et sur les hypothèses déjà générées et inscrites sur le blackboard. Quand les sources de connaissances créent une hypothèse à partir des hypothèses précédentes, elles étendent en fait la solution partielle existante avec davantage d'informations réduisant ainsi l'incertitude. Le traitement est terminé lorsqu'une solution hypothétique consistante est générée.

Toutefois, une source de connaissances génère souvent des hypothèses incorrectes, non seulement à cause de ses connaissances, généralement limitées, mais aussi à cause des hypothèses antérieures qui peuvent être incomplètes ou entachées d'erreurs. Comme chaque source de connaissances contribue à une partie de la solution globale, celle-ci, une fois obtenue, peut être imprécise et incomplète. Pour éviter ce problème, les sources de connaissances créent *différentes possibilités* d'hypothèses et associent à chacune d'elles une e*stimation de crédibilité*, c'est à dire une probabilité que l'hypothèse soit correcte.

5.2 - Utilisation du Blackboard : le paradigme multi-experts

Bien qu'issu de la reconnaissance de la parole et plus précisément d'Hearsay II, le concept blackboard s'est révélé par la suite très utile pour la mise en place d'un

résolveur de problème faisant intervenir plusieurs experts capables de coopérer afin d'obtenir une solution par raffinements successifs. Plus précisément, ce type de résolveur de problème est conçu de façon à permettre à chaque expert participant de donner son point de vue sur les hypothèses inscrites sur le blackboard. Ainsi, il peut infirmer, confirmer, critiquer ou remplacer les hypothèses inscrites suivant ses connaissances et ses capacités. Cette approche a été utilisée par plusieurs équipes de recherches dans différents domaines.

Tel est le cas par exemple du projet à l'université de Rochester concernant le langage naturel (Davis [1982]). En effet, l'objectif est ici de construire un ensemble d'experts coopérant pour l'analyse des mots. Un tel système vise à éviter les ambiguïtés de sens, à trouver des références, à comprendre des locutions et des métaphores, et à apprendre de nouveaux mots ainsi que leur sens. Le système est construit de façon à permettre un contrôle dynamique qui profite des opportunités nouvelles. Il utilise pour cela une mémoire globale de communication semblable au blackboard.

Un autre domaine d'application qui se prête bien à la coopération entre experts est le diagnostic médical. Dans ce cadre, Gomez et Chandrasekaran [1981] utilisent un ensemble de spécialistes coopérant à l'élaboration d'un diagnostic. Leur système, appelé MDX, est bâti autour d'un blackboard, et communique avec deux autres systèmes : PATREC, une base de données relative aux malades, et RADEX un spécialiste de consultation en radiologie.

Cette méthode a en fait permis à ses auteurs de construire graduellement des systèmes à bases de connaissances complémentaires, qu'ils pouvaient difficilement intégrer en une seule fois et en un seul module vu la quantité de connaissances manipulées.

L'équipe de la Rand Corporation a, quant à elle, développé le système AUTOPILOT destiné à la planification distribuée dans le domaine du contrôle aérien. Dans ce cadre, chaque avion considéré comme agent est modélisé par un système AUTOPILOT qui est capable d'interagir avec les autres systèmes AUTOPILOT pour résoudre avec eux tout éventuel conflit aérien. Plus précisément, l'architecture de chaque avion est basée autour d'un blackboard central, contenant un modèle du monde et différents experts coopérants ; le senseur, le générateur de plan, l'évaluateur de plan, le communicateur et le contrôleur. AUTOPILOT procède alors de la façon suivante : toute information sur l'environnement issue des senseurs est à la fois utilisée par le générateur pour l'élaboration du plan, et par l'évaluateur pour définir le plan de vol optimal. Une fois le plan établi, le contrôleur le met en action. Le communicateur, quant à lui, permet l'échange d'information entre les différents systèmes AUTOPILOT.

Les auteurs ont particulièrement étudié l'organisation d'un groupe d'agents concernés par un éventuel conflit aérien et chargés de le résoudre de matière totalement coopérative. Dans ce cadre, trois structures organisationnelles ont été proposées :

- une organisation sans communication ni coopération;

- une organisation dans laquelle les communications sont limitées et où les avions sont uniquement autorisés à s'échanger leurs plans de vol;
- une organisation à hiérarchie initialisée dynamiquement qui place le dernier avion, pénétrant dans l'espace aérien, au sommet de la hiérarchie.

Seules les deux dernières organisations furent implémentées. Les résultats de la simulation, concernant le temps requis pour la résolution du conflit aérien, ont montré que les deux structures conviennent à un espace aérien faiblement chargé, mais que l'organisation hiérarchique est plus performante dans un espace chargé. Ces expériences ont mené l'équipe de la Rand Corporation à développer un langage spécifique pour décrire les tâches distribuées similaires au contrôle de trafic aérien. Ce langage appelé ROSIE s'est révélé plus commode à utiliser pour les tâches distribuées que les langages classiques.

5.3 - Conclusion

Cette partie du chapitre s'est attachée à montrer l'importance de la coopération entre agents au moyen du blackboard. Il est clair que ce puissant moyen de communication convient bien à des experts qui doivent mettre en jeu leurs connaissances pour construire la solution d'un problème par raffinements successifs. La solution globale, obtenue de cette façon, a en outre l'avantage d'exclure l'incertitude et l'imprécision, à cause de la multiplicité des points de vue des participants. Cependant, le blackboard, qui ne peut convenir qu'à des experts travaillant localement, risque de devenir un goulot d'étranglement pour ces mêmes experts, si l'échange d'informations entre eux devient trop important. Une autre possibilité serait de concevoir des systèmes qui puissent combiner les deux moyens de communication, le blackboard pour l'élaboration par raffinements successifs de la solution locale et l'échange par messages pour le partage des tâches et des informations entre systèmes muni chacun de son blackboard. Un tel modèle semble bien convenir aux applications où intervient l'interprétation distribuée de plusieurs senseurs, comme l'a montré l'exemple précédent de la surveillance de trafic de véhicules.

Une autre approche à la coopération entre agents qui s'échangent des connaissances, consiste à munir chaque agent d'un modèle d'autrui. Cela permet à tout agent capable de raisonner et de lier ses connaissances aux actions qu'il a à entreprendre, de tenir compte de l'environnement et d'exécuter ses actions dans de meilleures conditions. Cette approche est détaillée ci-dessous.

Section 6 - Approche utilisant le raisonnement sur les connaissances et les actions

L'objectif de cette approche est de donner à chaque agent les moyens de raisonner afin d'atteindre ses propres buts, tout en possédant un comportement social cohérent. Cependant, dans un environnement réel comprenant d'autres entités intelligentes, tout agent doit se contenter d'informations incomplètes, imprécises, voire même incertaines pour raisonner afin d'extrapoler et de prédire, aussi bien son futur comportement que celui des autres agents. Ceci est dû, d'une part aux moyens de communications qui sont bruités, imprécis et limités physiquement, et d'autre part aux connaissances restreintes de chaque agent.

Dans ce contexte, on associe à chaque agent un modèle de son environnement sous forme d'une base de connaissances, afin de lui permettre de raisonner. Comme ces connaissances peuvent ne pas être nécessairement objectives, elles sont appelées croyances. Bien entendu, la base de croyances ne suffit pas et il faut lui adjoindre un mécanisme, généralement déductif, qui permet à chaque agent de raisonner en tenant compte d'une part de ce qu'il croit, et d'autre part, de ce que croient les autres agents impliqués dans le même problème.

Comme l'univers dans lequel évolue tout agent est dynamique, il faut aussi poser le problème d'une planification réactive de ses actions (replanification), qui soit en mesure de permettre une adaptation de l'agent à l'environnement. Il est donc préférable d'élaborer des plans en tenant compte des connaissances de l'agent et de coupler directement ces plans aux actions à effectuer. Ce couplage permet en fait à chaque agent de raisonner en *boucle fermée*. Pour cela, il surveille les effets des actions, les événements imprévus, de toute nouvelle information susceptible de conduire à un affinement ou à la poursuite du plan à une replanification ou à un changement d'objectif.

Par ailleurs, la structure organisationnelle et le degré de coopération pour ce type d'approche sont dépendants du problème à résoudre. Autrement dit, les agents sont conçus de façon à ce qu'ils sachent s'auto-organiser face à tout problème. L'échange des connaissances entre agents se fait, quant à lui, par messages dont le protocole et le contenu sont à définir en fonction du problème à résoudre.

En fait, la difficulté de cette approche réside, d'une part dans la représentation en détail des effets des actions d'un agent donné sur l'état cognitif d'autres agents, et d'autre part dans la recherche d'un formalisme adéquat pour le raisonnement à partir des connaissances, une fois celles-ci représentées.

Section 7 - Conclusion

Ce chapitre a présenté les différentes approches de résolution distribuée de problème. L'approche empirique, illustrée par les stratégies de coopération dans le contrôle aérien, est une bonne initiative pour tester des structures organisationnelles et des politiques de distribution de l'information dans un contexte où les compétences de chacun sont soit connues à priori, soit estimables d'une façon ou d'une autre. Le but de cette approche est d'élaborer des structures organisationnelles de façon formelle et de tester leur incidence sur la performance du groupe. Ainsi pourra se dégager le choix d'une stratégie coopérative tenant compte de l'assignation des rôles, de la charge de communication et de la performance du groupe.

Une deuxième approche de résolution distribuée utilise le mécanisme de négociation entre demandeur et contractants dans le but, soit de donner aux agents les plus compétents la tâche qui leur revient au vu de leurs compétences et de leurs capacités, soit de donner aux agents inactifs certaines tâches pour équilibrer la charge de travail. Ce protocole, basé sur l'échange par messages, est généralement utilisé dans les problèmes décomposables en plusieurs tâches et/ou dispersées géographiquement. Toutefois, dans un tel protocole de négociation, si un agent est choisi pour une tâche donnée, il ne doit pas s'avérer par la suite incapable de résoudre une sous-tâche issue de la tâche donnée au départ. C'est donc à l'agent demandeur de prédire qu'une telle incapacité pourrait se produire et ainsi choisir le "bon contractant". L'incapacité des agents à faire de telles prédictions peut mener à des incohérences dans le réseau. En conséquence, dans un réseau à contrats, les agents peuvent former un équipage cohérent et améliorer leur performance globale s'ils font une assignation des sous-tâches aux noeuds qui conviennent et qui ne sont pas connus à priori.

L'approche de résolution par raffinements successifs permet aux experts de résoudre un problème par partage des données et des objectifs au moyen d'une zone commune appelée blackboard. Pour pouvoir coopérer de façon cohérente, les experts doivent prédire les futures solutions partielles qu'ils doivent s'échanger dans le blackboard, et ainsi modifier leurs activités courantes dans le but de former des solutions partielles compatibles. Pour faire ces prédictions, chaque expert doit comprendre ses propres plans d'actions et les plans des autres experts qui interagissent avec lui. Sans cette compréhension, les experts peuvent mettre beaucoup plus de temps à converger vers une solution puisqu'ils peuvent travailler à des buts conflictuels.

Enfin, l'approche utilisant les connaissances, les croyances et les actions, bien que difficile à mettre en oeuvre, est à notre avis une voie très prometteuse pour au moins trois raisons :

- La première raison est qu'un modèle d'autrui, à la condition d'être bien formulé, permet à chaque agent de raisonner de manière prédictive sur ses intentions, ses buts et ses futures interactions, de manière à n'utiliser les communications que lorsque cela s'avère nécessaire. Par ailleurs, lorsque l'échange

d'informations a lieu, il doit être basé sur les croyances et les intentions de façon à lier le raisonnement aux actes communicatifs. Un modèle basé sur ces concepts permet aux agents *d'interagir de manière rationnelle*.

- La deuxième raison est que cette approche permet une formulation plus riche scientifiquement. Ainsi, des problèmes de modélisation de connaissances ou de croyances et des problèmes de raisonnement sur ces propres connaissances sont soulevés . En outre, les problèmes de communication avec l'extérieur ne sont plus posés en termes d'entrées/sorties, mais exigent que la communication soit modélisée et traitée comme un *acte réfléchi.*

- Enfin, la troisième raison est l'ouverture que permet une telle approche vers d'autres horizons tels que les sciences cognitives et la modélisation des concepts humains (pour les études de connaissances, de croyances et d'intentions), ou les sciences sociales et l'économie (pour les métaphores de coopération entre entités intelligentes).

Le but de cet ouvrage étant de trouver une méthode de spécification pour les systèmes multi-agents, nous avons donc dû opérer certains choix .

En effet, le type d'organisation dans laquelle on se trouve peut être soit très hiérarchisé, soit du type "experts coopératifs". Quant à la communication entre agents, nous trouvant dans l'intelligence artificielle, il s'agit évidemment d'une communication par message pour éviter les goulots d'étranglement provoqués par le blackboard.

Les chapitres suivant présentent une méthode qui nous aidera à résoudre le problème de la négociation. La seule petite différence à mentionner serait le fait que lorsqu'un manager fait une offre, généralement, il n'y a qu'un seul contractant possible.

En dehors de tout cela, il est évident que notre méthode est applicable à n'importe quel type d'organisation. Notre méthode va donc être détaillée en long et en large dans les chapitres suivants.

CHAPITRE 3 : Méthode de modélisation EPAS

Section 1 - Introduction

Le chapitre précédent a présenté les différentes approches existantes dans la résolution distribuée de problèmes. Notre problème consite à présent à mettre au point une méthode qui se veut générale dont l'objectif est la modélisation de la négociation entre agents; elle sera présentée au chapitre 4. Elle se compose de deux parties. La première a pour but l'identification des divers agents; la deuxième s'occupe de modéliser la négociation qu'engendreront les agents afin de résoudre un problème. A cette fin, c'est la méthode EPAS (**E**ntrée **P**rocessus **A**ccumulation **S**ortie, (MOULIN B. [1988])) qui nous servira et qui sera adaptée en conséquence. De plus, nous devons obtenir, au terme de notre méthode, des spécifications compréhensibles par un outil d'aide à la programmation, le Système Générique de Groupes Transitoires (SGGT); celui-ci est présenté au dernier chapitre.

La méthode EPAS permet d'isoler des systèmes, de schématiser leur comportement et enfin de planifier leurs communications, notion fondamentale de la négociation entre ces systèmes. Elle peut être utilisée aussi bien pour la modélisation de systèmes d'information que pour la conception de systèmes à base de connaissances.

Elle fait, dans une première étape, une modélisation des comportements de la réalité grâce aux plans d'action: les diagrammes EPAS. La deuxième étape se préoccupe de la manière dont sera structurée la mémoire du système. Ainsi, il faudra identifier les entités chargées d'emmagasiner les information utiles au système et en donner la structure et, ensuite, il sera nécessaire de schématiser l'évolution de cette mémoire. Enfin la dernière étape de la méthode E.P.A.S. servira d'articulation entre les plans élaborés dans la première partie et la mémoire du système; nous déterminerons ainsi les scénarii qui schématiserons les comportements des divers systèmes modélisés.

Pour fixer définitivement les idées, nous terminerons ce chapitre par une planche aggrégeant les divers éléments de la démarche de modélisation des systèmes.

Section 2 - Choix d'un formalisme

Il existe bien des formalismes de modélisation dans la bibliographie informatique qui auraient pu servir de support à notre méthode. Nous pensons entre autres à la méthode IDA (Interactive Design Approach) développée à l'institut d'informatique des Facultés Universitaires Notre Dame de la Paix de Namur (BODART F. et PIGNEUR Y.[1989]). Cette méthode tout comme la méthode EPAS a l'avantage d'avoir fait ses preuves et n'est donc plus à présenter; de plus, elle est extrêmement précise et rigoureuse et dispose de nombreux mécanismes de validation et de complétude des spécifications.

Le choix de la méthode EPAS comme démarche de modélisation n'est pas entièrement arbitraire. La première raison qui nous a poussé à la choisir parmi les autres est essentiellement pratique. En effet, elle a été mise au point par Moulin [1988], notre chef de stage qui nous l'a recommandée. Nous pouvions donc à tout moment l'interroger à propos de l'utilisation de certains de ses concepts, mais aussi lui soumettre nos éventuelles adaptations. Dans un deuxième temps, cette démarche de conceptualisation se veut générale, ce qui est un atout car cela lui confère une facilité d'adaptation. Enfin, nous avons la possibilité de présenter une nouvelle méthode de modélisation de systèmes ayant déjà fait ses preuves et possédant un bel avenir vu sa facilité d'adaptation.

La méthode EPAS est une méthode de modélisation complète. Composée de quatre phases, la phase d'orientation qui présente le problème, la phase d'analyse conceptuelle permettant la modélisation du problème afin d'obtenir un système-cible c'est-à-dire le système visé dans les spécifications, la phase d'analyse fonctionnelle qui prépare le modèle obtenu à la phase précédente afin d'en dériver un système tenant compte des caractéristiques physiques des éléments sur lesquels sera installé le système et, enfin, la phase de construction qui consiste en l'implémentation du système. On peut se rendre compte que la méthode EPAS s'étend de la spécification du problème à la résolution du système-cible sur machine.

De ces quatre phases, c'est sur la phase d'analyse conceptuelle que nous focaliserons notre attention. En effet, il s'agit de la phase la plus importante dans la modélisation d'un problème car c'est par elle que le concepteur devra passer afin d'obtenir une spécification complète et la plus indépendante possible de tous langages de programmations. Nous la découvrirons au travers des sections suivantes.

Section 3 - La phase d'analyse

3.1 - Introduction

Nous nous trouvons à la phase requérant le plus de temps et d'attention car c'est durant cette phase que nous allons modéliser le problème. Cette phase comprend un certain nombre d'étapes dont nous ne retiendrons et n'expliquerons que celles reprises dans notre méthode, présentée au chapitre suivant. Les autres, jugées moins utiles en ce qui nous concerne, servent à des vérifications, corrections et autres sécurités à prendre lors de développements de logiciels de grande importance, pour lesquels des budgets importants sont consacrés, nécessitant une attention accrue car la moindre négligence pourrait être grave de conséquences. Citons à titre d'exemples la coordination d'un réseau de chemin de fer ou d'une piste aérienne ou encore un programme de contrôle de satellite. Comme ce n'est ni notre cas ni celui d'un grand nombre de logiciels informatiques, nous pouvons nous permettre de ne pas les détailler ici.

3.2 - Approche systémique

Avant de commencer le développement de la phase d'analyse conceptuelle, il nous semble opportun de la situer dans le contexte des méthodologies qui s'offrent aux concepteurs de logiciels. En effet, depuis son avènement, le matériel informatique (hardware) n'a pas cessé d'être confronté à des bouleversements. Parallèlement, les méthodes de résolution subirent des révolutions. C'est ainsi, comme l'explique Rolland (ROLLAND C. [1992]), que l'on passa d'une méthodologie cartésienne, découpe la plus fine possible de manière à obtenir des fonctions atomiques, à une méthodologie orientée objet en passant par une approche de résolution des systèmes dans laquelle le moteur est le regroupement de procédures et fonctions en modules indépendants et autonomes : les méthodologies systémiques.

Cest dans cette dernière catégorie que se situe la méthode de modélisation EPAS. Mais que savons nous au juste de l'approche systémique ? Selon Le Moigne (MOIGNE (LE) J.-L. [1990]), il s'agit là d'une théorie étudiant les phénomènes perçus. Bien maigre définition! C'est pourquoi nous ajoutons que, d'une part, cette approche nous permet une visualisation du phénomène dans sa globalité et une division progressive d'un système en sous-systèmes; d'autre part, elle nous propose de regarder les interactions existant entre les différents éléments mis en évidence. Grâce à sa caractéristique de modélisation par affinement successif, la méthode EPAS se révèle être l'outil idéal pour la découverte et la résolution des sous-systèmes et leurs interactions.

Afin d'exprimer chacun des systèmes et sous-systèmes, nous aurons recours aux diagrammes de modélisations des comportements, appelés aussi les plans de systèmes ou diagrammes EPAS. Ce concept est expliqué en détail au point suivant.

3.3 - Les plans

Nous connaissons d'ores et déjà le domaine dans lequel évolue l'approche de conception EPAS. Il s'agit d'une méthode de modélisation du comportement procédant par raffinements successifs, où, à chaque niveau du raffinement, nous nous voyons confronté à plusieurs sous-systèmes qu'il nous faudra détailler s'ils ne se décomposent pas à leur tour en sous-sysèmes.

Dans cette partie, nous exposerons les différents concepts utilisés dans les diagrammes de modélisation des comportements, appelés plus communément les plans de systèmes ou diagrammes EPAS. Dans la suite de l'exposé, nous parlons de comportements. Nous définissons un comportement comme étant la combinaison d'une ou plusieurs tâches.

Nous définissons un plan comme un moyen de représenter un comportement adopté par un système ou un sous-système. Afin de détailler l'enchaînement des différentes phases du comportement, nous introduirons dans les plans les tâches induites par ce comportement et les interactions qu'il engendre avec les autres systèmes. Nous verrons par la suite les différents principes intervenant dans la méthode de modélisation qui régissent, dans l'approche systémique de problèmes, la découpe du comportement ciblé en sous-sytèmes ou micro-comportements.

tout plan pourra être représenté graphiquement. Cette représentation est constituée d'un rectangle appelé diagramme de plan, un trait horizontal le divisant en deux surfaces inégales. La première, la plus petite, sert à contenir l'intitulé du plan. La deuxième, plus grande, offre la possibilité d' exprimer le comportement que le plan doit représenter.

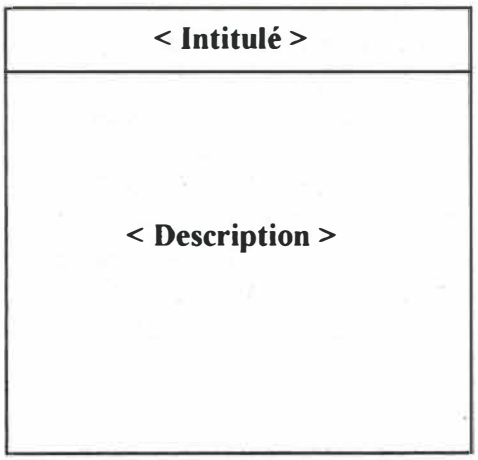

Figure 3.1 Diagramme EPAS - plan de système

Définissons d'abord l'intitulé. Cette partie permet de nommer le plan, et cela rend possible l'identification d'un plan parmi tous les autres. Si nous prenons l'exemple de plan donné à la figure 3.2, on voit que l'intitulé est "P3.2 enregistrer client". On identifie ce plan dans la modélisation du comportement d'enregistrement d'un client dans un hôtel, par le réceptionniste, car l'intitulé porte l'étiquette "*enregistrer client*".

La deuxième partie du diagramme est consacrée à la "*description du comportement*". Selon la méthode EPAS, l'objectif des plans est de détailler un comportement plus général situé dans un plan de niveau supérieur. Afin de décrire les tâches liées à ce comportement, nous utiliserons le concept de *processus*, le moyen d'articuler les différents processus entre eux sera fourni par les concepts d'*accumulations* et de *canaux*; enfin, les *environnements* et les *flux* schématiseront les interactions des systèmes entre eux.

Avant de poursuivre et de définir ces concepts : processus, accumulation, environnement, canal et flux, nous présentons un exemple de plan de système. Ce plan (Figure 3.3) a pour objectif de schématiser le comportement consistant en l'enregistrement d'un client dans un hôtel par un réceptionniste. Pour ce faire, le processus "**enregistrer client**" (P321) est activé lorsque le **client** (E1) se présente au comptoir de réception de l'hôtel pour s'y faire enregistrer. Les informations fournies par le client sont placées dans le **dossier client** (A3). Ce processus met à jour le **feuille de réservation individuelle** (A1) et le **tableau des occupations** (A2) des chambres. En fonction des informations contenues dans le dossier client (A3), le processus "**Mise à jour des réservations**" (P2) met à jour le **tableau des réservations** (A4) s'il y a lieu; par exemple, si le client s'inscrit pour des journées supplémentaires sans qu'il y ait eu de réservation préalable.

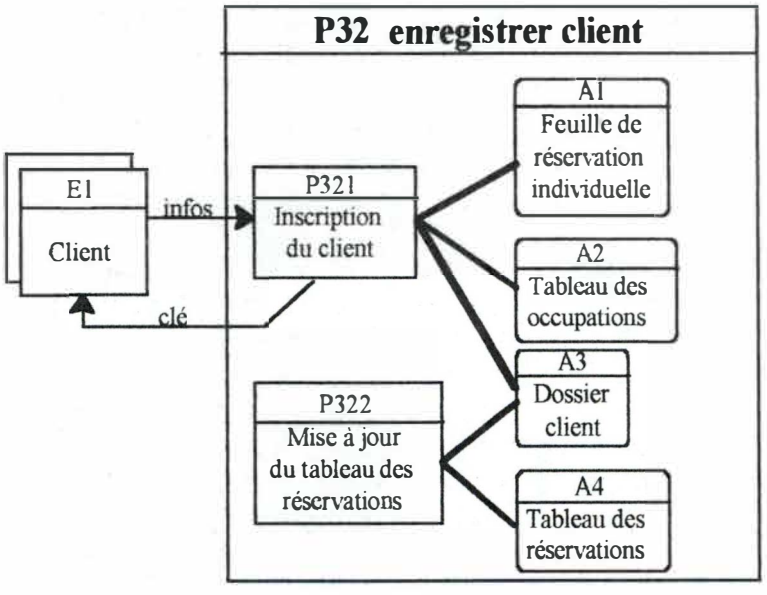

Figure 3.2 Illustration d'un plan du système

3.3.1 - Les processus

Le processus est un élément particulier des diagrammes EPAS car il peut à lui seul décrire l'organisation des divers éléments d'un sous-système mais combiné avec les autres processus d'un même diagramme, il permet de mieux décrire le comportement exprimé par un processus de niveau supérieur, appelé le "processus père".

Le processus est donc destiné à schématiser une des tâches accomplies par un comportement. Ainsi dans l'exemple de l'enregistrement d'un client (Figure 3.2), le comportement "enregistrer client" comprend deux tâches : "**inscription du client**" (P321) et "**Mise à jour du tableau de réservation**" (P322). Si tel est le rôle d'un processus, voyons un peu sa strucure.

La forme d'un processus (Figure 3.3) est identique à celle des plans (Figure 3.1) mais néanmoins plus fine car il doit pouvoir se glisser dans un diagramme de plan. Il possède lui aussi un "intitulé" et une "description du comportement". Cette description consiste à exprimer son rôle, c'est-à-dire un mot ou une petite phrase (sujet-verbe-complément) caractérisant le comportement. Ce n'est pas sans raison qu'il possède une telle ressemblance avec le plan car : "*Le processus correspond à un plan ou une activité du système.*" (MOULIN B. [1988]) Le processus est donc le moyen de représenter un plan ou une partie de plan dans un plan.

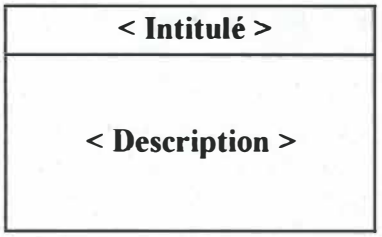

Figure 3.3 Processus

Règle de nomenclature :

Cette règle s'applique à l'intitulé. Celui-ci se trouve décomposé en deux parties : une lettre P et deux indices i et j. Le premier indice "*i*" est identique à l'indice de plan et le deuxième "*j*" identifie le processus dans le plan. Pij pouvant dès lors s'énoncer comme étant le j^{ème} processus du plan i.

L'exemple présenté à la figure 3.2 présente deux processus mais ne montre pas de liens entre-eux. Cela est dû au caractère événementiel dans l'activation des processus provenant d'un changement de valeur d'une accumulation à la quelle est lié ce processus. On parle du caractère "événementiel" car le changement de valeur se fait suite à un événement ponctuel agissant sur l'accumulation.

3.3.2 - L'accumulation

Si les processus nous permettent de décrire le comportement d'un sous-système ou, si l'on préfère, un micro-comportement d'un système, les accumulations, quant à elles, nous offrent le moyen de décrire la mémoire de ce système et de créer les articulations entre les processus des plans. L'accumulation est le moyen par lequel la connaissance des agents est représentée, cela sera détaillé au chapitre suivant. Ainsi dans l'exemple de l'enregistrement d'un client (Figure 3.2), la mémoire du système d'enregistrement se compose de quatre accumulations : une **Feuille de réservation indivuduelle** (A1), le **Tableau des occupations** (A2), le **Dossier client** (A3) et le **Tableau des réservations** (A4).

Il nous faut, à présent, voir la manière de représenter graphiquement cette connaissance selon EPAS. La forme décrite (Figure 3.4) est celle d'un simple rectangle à coins arrondis et divisé en deux surfaces inégales. La partie supérieure identifie l'accumulation, quant à la partie inférieure, elle permet de la nommer avec un nom explicite.

Figure 3.4 Accumulation

Règle de nomenclature :

Une accumulation est identifiée par une lettre "*A*" et deux indices "*i*" et "*k*". *Aik* s'énonce comme étant la k<u>ième</u> accumulation du plan i. Le nom nous donnant un identifiant plus explicatif.

Dans l'exemple de la figure 3.2 se trouvent quatre accumulations : A1, A2, A3 et A4. Nous nous apperçevons immédiatement que les plans seraient illisibles si nous n'inscrivions pas le nom dans les accumulations comme second identifiant.

3.3.3 - L'environnement

Nous savons qu'un système communique avec d'autres systèmes pour mener à bien l'ensemble des tâches qui incombent à son comportement. Nous utiliserons les environnements pour représenter ces autres systèmes. Sa représentation est formée de deux rectangles superposés (Figure 3.5). La partie supérieure, du rectangle en avant

plan, contient l'identifiant de l'environnement; dans la partie inférieure est inscrit le nom du système entrant en interaction avec un des processus du plan.

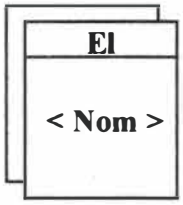

Figure 3.5 Environnement

Règle de nomenclature :

Un environnement est identifié par un numéro "*I*" précédé de la lettre "*E*". Pour une meilleure compréhension, nous lui donnerons un nom. Les environnements sont donc uniformément identifiés au travers de tous les plans. Cette numérotation tire son origine du fait que différents systèmes communiquent entre eux pendant leur négociation afin de résoudre un problème.

Le représentation d'un comportement étant faite, nous disposons de tous les éléments nécessaires afin de représenter un système. Il nous reste à modéliser les interactions de ce système avec ses connaissances et son environnement externe. C'est par l'intermédiaire des canaux et des flux que nous créons les axes interactifs entre ces concepts.

3.3.4 - Le canal

Le canal est représenté par un trait épais reliant un processus à une accumulation (Figure 3.6). Cette association permet de créer un lien entre un processus et une accumulation dans le but de schématiser les transferts d'informations entre ces deux concepts.

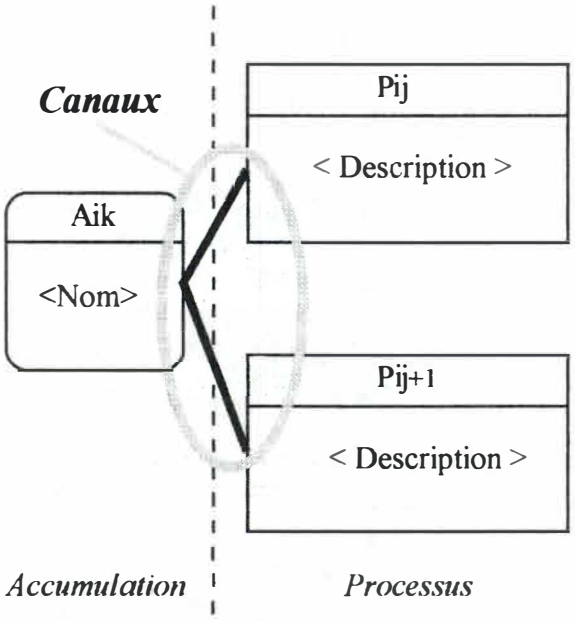

Figure 3.6 Canal

3.3.5 - Le flux

Le flux est représenté par une flèche unidirectionnelle (Figure 3.7) entre un processus et un environnement. Cela signifie qu'il y a "*communication qui se crée entre le processus et l'environnement*" (MOULIN B. [1988]).

Le flux sera utilisé lorsqu'un processus aura besoin de transmettre ou recevoir de l'information de l'environnement extérieur.

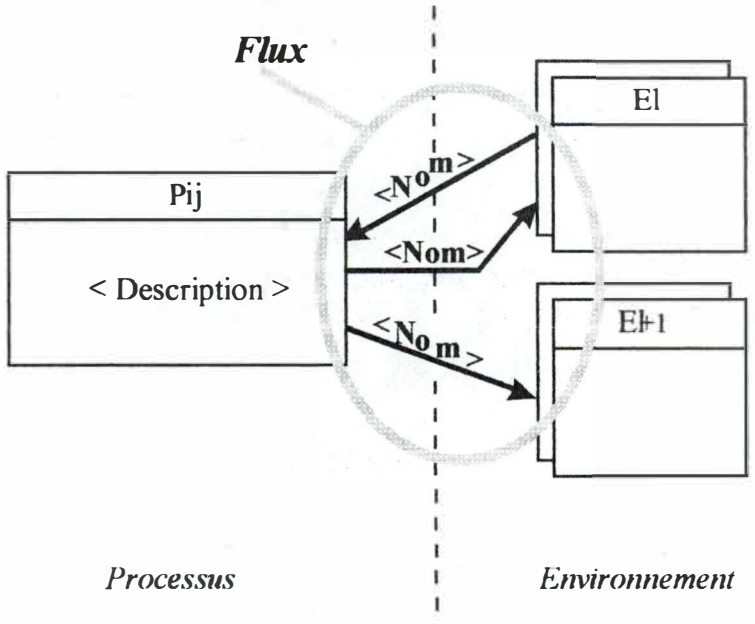

Figure 3.7 Flux

Règle de nomenclature :

Le flux d'un diagramme sera représenté par un arc unidirectionnel liant un processus à l'environnement du système. Cet arc sera dénommé afin de l'identifier comme événement déclencheur d'un processus d'un sous-système et déterminer la nature de ce flux. De plus, l'ordre des flèches n'a pas d'importance car il s'agit de représenter la communication.

Au dernier niveau d'abstraction atteint lors de la modélisation du comportement global du sytème par la méthode des plans, nous aboutissons à un système dans lequel ont été mis en évidence les différents environnements et accumulations intervenant dans la description du comportement ciblé. Tout comme les plans, ces accumulations possèdent leur niveau d'abstraction que nous pouvons qualifier de "flou structural interne", c'est-à-dire qu'elles représentent la mémoire du système sans posséder de réelle structure. C'est ainsi qu'un processus sera en association avec telle accumulation mais pas avec tel élément particulier.

Suite à l'identification des accumulations composant la mémoire du système, il faut les exprimer en terme d'attributs. C'est l'objectif que tentera d'atteidre la section suivante.

3.4 - Structure conceptuelle des données

3.4.1 - Introduction

L'ensemble des accumulations, identifiées au point précédent, constitue la mémoire du système. Mais leur représentation n'est guère détaillée. En effet, face à ces accumulations, une question se pose; quelles informations va-t-on inclure dans la structure de telle ou telle accumulation ? La réponse ne découle pas directement des processus "manipulateurs" bien qu'ils permettent d'en mettre en évidence une partie. Il faut y ajouter la connaissance que nous avons déjà du système.

Au terme de cette étape, on aura spécifié la structure de la mémoire, c'est-à-dire la manière dont seront ordonnées les informations au sein de chacun des systèmes.

3.4.2 - Définition

La structure conceptuelle des données permet d'organiser la mémoire du système, c'est-à-dire qu'elle est un moyen d'exprimer la faculté d'enregistrer et de retrouver des informations. De plus, elle possède une structure associative qui permet de créer des

liens de références entre les éléments d'information dans le but de facilité la mémorisation et la recherche de données.

La structure conceptuelle des données est élaborée en tenant compte des éléments qu'elle doit décrire et se limite à retenir les informations utiles au système; elle tient compte des caractéristiques sémantiques pertinentes. Ainsi dans l'exemple de la figure 3.8, la structure de données représentant une **chambre** (2) ne se souciera pas de mémoriser la couleur de la porte, ni les matériaux composant la décoration de la chambre ou même l'étage à laquelle elle se trouve.

Nous nommerons *entité* un élément de la structure conceptuelle des données. Ainsi dans la figure 3.8, on distingue quatre entités : **occupation** (1), **chambre** (2), **séjour** (3) et **client** (4). Une entité est selon Bodart (BODART F. et PIGNEUR Y.[1989]) : "*une chose concrète ou abstraite appartenant au réel perçu à propos de laquelle on veut enregistrer des informations*."

Comme la structure conceptuelle des données est constituée d'un ensemble d'entités, elle se doit de posséder une structure, c'est-à-dire que les entités seront liées entre elles par des associations ou liens; cette structure montre les relations existant entre les choses, nous le voyons dans l'exemple de la figure 3.8. Ces liens montrent qu'il existe une relation entre l'entité **occupation** et les entités **chambre** et **séjour**, de même qu'il existe un lien entre **séjour** et **client**. Tout d'abord entre occupation et chambre car une occupation porte toujours sur une chambre et ensuite entre occupation et séjour car le client occupera au moins une chambre lors de son séjour.

Enfin bien que la structure des données puisse mémoriser les éléments du réel perçu à partir de certaines valeurs les caractérisant, elle ne donne qu'une vision statique de la mémoire. En effet, chaque entité n'est qu'un reflet statique d'un objet de la réalité faisant partie du système-cible, le système à modéliser.

La définition de la structure conceptuelle de données étant éclairci, il nous faut spécifier sa structure. Le point suivant présentera la manière dont il faut construire une structure de données.

3.4.3 - Structuration des données

Avant de montrer ce qu'est un schéma conceptuel au moyen d'un exemple, nous présentons la structure des divers éléments le constituant. Le rôle du schéma conceptuel des données est de représenter la manière dont est structurée le mémoire du système. Selon cette optique, nous définirons les entités et les liens exprimant les relations qui existent entre les éléments d'information conservés dans ces entités.

Nous commençons par définir formellement la manière dont on représente une entité au sein de l'étape "structure conceptuelle des données" de la démarche d'analyse proposée par la méthode EPAS.

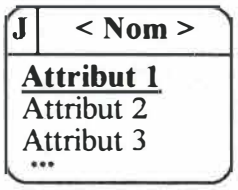

Figure 3.8 Structure d'une entité

Règle de nomenclature :

Une entité de la structure de données (Figure 3.8) possède un numéro identifiant que l'on place dans le coin supérieur gauche. Elle contient aussi un nom, représenté par une chaîne de caractères, placé dans le coin supérieur droit. Enfin, pour décrire l'objet de la réalité, elle est composée d'une série d'attributs dont un pourra être identifiant; si tel est le cas, il sera souligné afin de le différencier des autres.

Le deuxième élément entrant dans le schéma de la structure conceptuelle des données est la flèche (Figure 3.9). Son rôle est la représentation des liens existant entre les entités. L'orientation est importante car elle indique le sens des références. Ainsi dans l'exemple de la figure 3.10, l'entité **occupation** référence les entités **chambre** et **séjour**.

Figure 3.9 Structure d'une référence

Illustrons cette théorie au moyen d'un exemple inspiré par MOULIN [1988] : il s'agit de présenter, de manière simplifiée, la gestion des réservations des chambres d'hôtel. Lorsqu'un client, que le système de réservation conservera dans sa mémoire au moyen de l'entité **client** (4), fait une demande de réservation d'une chambre, entité **chambre** (2), il spécifiera la période de son séjour que le système enregistrera dans sa mémoire ,entité **séjour** (3). Le système réservera une certaine chambre pendant la période souhaitée par le client, cette information sera contenue dans l'entité **occupation** (1) et les références aux autres entités seront crées au moyen de flèches.

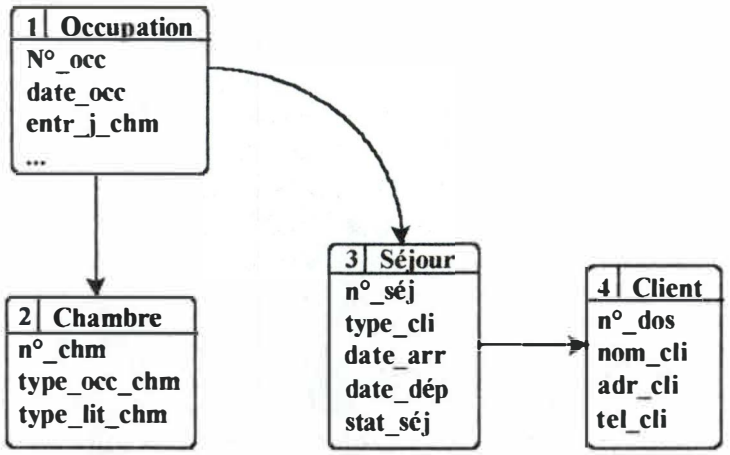

Figure 3.10 Illustration d'un schéma de la structure conceptuelle de données

"La mémoire présentée sous la forme d'une structure conceptuelle de données fournit seulement une vision statique de l'organisation des données. Elle décrit un ensemble d'objets et leurs liens d'association qui représentent en fait des propriétés de classes d'occurrences d'objets." (MOULIN B. [1988])

L'étape suivante constitue donc le complémentaire à cette étape et présente l'évolution de la mémoire dans le but de présenter une vision dynamique de cette mémoire.

3.5 - Cycles de vie des entités

La section précédente a présenté la structure de la mémoire dont le rôle est d'enregistrer et de restituer l'information. Elle permet de représenter dans un langage de modélisation des concepts ou des objets de la réalité. Comme nous l'avons vu, ce mode de représentation ne permet pas d'exprimer l'évolution de l'objet ou du concept car il s'agit d'une photographie de la réalité à un moment donné. Ce point a pour but de présenter cette évolution.

Les objets ou concepts, référencés à ce point, sont ceux montrant l'évolution de la mémoire du système, c'est-à-dire les entités qui possèdent dans leur liste d'attributs un champ "*état*" dont le rôle est de spécifier les états par lesquelles passe cette entité tout au long de l'exécution du comportement du système.

3.5.1 - Qu'est-ce qu'un cycle de vie ?

"Au cours de leur évolution les occurrences d'un objet passent par divers états qu'il peut être nécessaire de connaître à des fins de contrôle ou pour déclencher un

comportement particulier du système en réaction à cette évolution. Le cycle de vie d'un objet décrit les divers états et les règles de transition d'un état à un autre que peuvent subir les occurrences de l'objet." (MOULIN B. [1988]).

Nous pouvons, grâce au cycle de vie des objets, modéliser à un niveau conceptuel les règles d'évolution des occurrences des objets. Nous représentons de ce fait la " dynamique du système", complétant ainsi la structure conceptuelle des données qui, pour sa part, était une représentation conceptuelle statique des connaissances du système.

3.5.2 - Les concepts entrant dans la composition du cycle de vie

A ce stade, nous avons présenté la notion de cycle de vie mais ne sommes pas encore en mesure de représenter graphiquement l'évolution des connaissances du système. Nous avons choisi de les représenter par un graphe orienté, pouvant contenir des cycles, composé de noeuds et d'arcs.

Avant de poursuivre dans la définition des symboles de représentation contenus dans les cycles de vie, reprenons l'exemple donné à l'étape de l'élaboration de la structure conceptuelle des données (Figure 3.10), et présentons le cycle de vie de l'entité **séjour** car elle possède dans la liste de ses attributs un attribut d'état : **état_séj**.

Les différentes valeurs que peut prendre cet état, entre le moment de la création et le moment de la destruction de l'entité **séjour**, tout au long de l'exécution du comportement "enregistrer client" sont : **incomplet** (1), **actif** (1), **inactif** (3), **complété** (4), **archivé** (5).

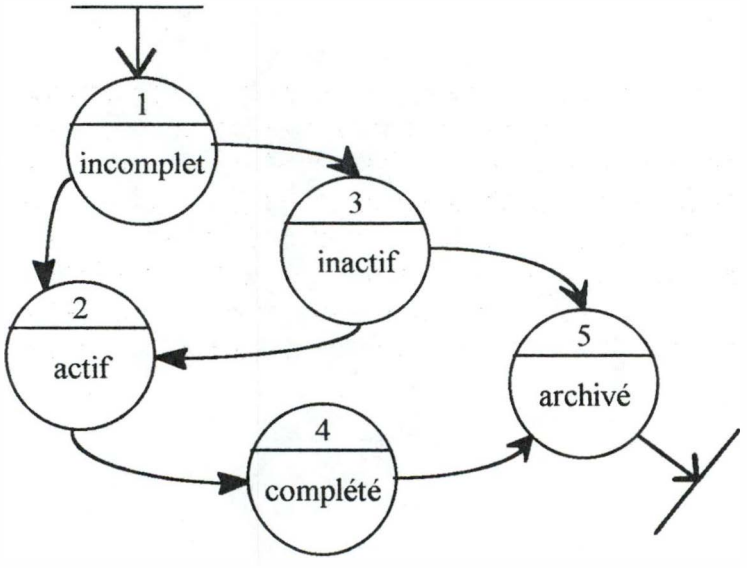

Figure 3.11 Illustration d'un cycle de vie

A Un cercle

Le cercle (**noeud**) est destiné à représenter un état de l'objet. Nous y inscrirons le nom de l'état que ce noeud représente à un moment de l'évolution de l'entité (Figure 3.12 A). De plus, afin d'identifier un état d'un autre, ce cercle est coupé en deux et contient dans sa partie supérieure un numéro identifiant (Figure 3.12 B).

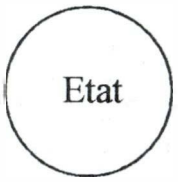

Figure 3.12 A

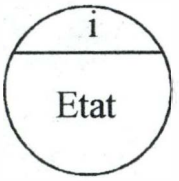

Figure 3.12 B

Dans l'exemple de la figure 3.11, on distingue cinq états : **incomplet**, **actif**, **inactif**, **complété**, **archivé** numérotés, respectivement de 1 à 5.

B Une flèche

Dans un graphe orienté, une flèche porte le nom d'arc orienté. Son rôle symbolique est la représentation d'une transition d'un noeud initial vers un noeud destinataire. Cela permet de montrer les différents changements d'état autorisés dans le système. Nous obtenons grâce à ce mécanisme un graphe de transitions d'états.

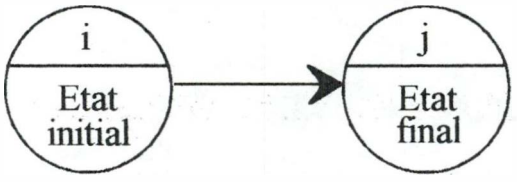

Figure 3.13 Structure d'une transition détat

On peut extraire de l'exemple de la gestion d'hôtel (Figure 3.11) une transition, par exemple, le passage de l'état **complété** (4) à l'état **archivé** (5) de l'entité séjour (Figure 3.10). Cette transistion signifie que lorsque le client sort de l'hôtel et que le dossier de séjour de ce client est complété, le dossier est archivé.

C Un trait

Un trait dans un graphe orienté signifie que l'occurrence se trouve dans un état particulier : le néant. A partir de cela, nous le combinerons avec une flèche pour obtenir les deux derniers concepts.

D Un trait une flèche

Notre intention est de montrer la **création** d'une entité. En effet, si une entité part de l'état néant pour un autre, grâce à une transition, c'est qu'il y a création de cette entité.

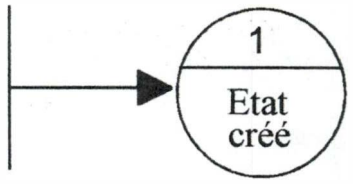

Figure 3.14 Structure de la création d'une entité

Nous sommes en mesure d'expliquer la première transition de l'exemple de la figure (3.11). Lorsque le client fait une demande de réservation d'une chambre, que se soit par lettre, téléphone ou à la réception de l'hôtel, il y a **création** d'une entité **séjour** (3) afin d'enregistrer la demande de réservation. Comme le réceptionniste ne dispose pas de toutes les informations nécessaires pour compléter cette occurence, elle est placée à l'état **incomplète** (1).

E Une flèche et un trait

Partant d'un état quelconque et suite à une transition nous aboutissons au néant, c'est qu'il y a **destruction** d'une entité.

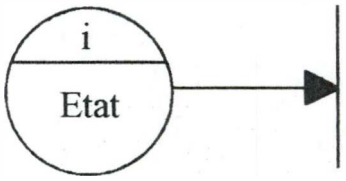

Figure 3.15 Structure de la destruction d'une entité

Reprenons l'exemple de la figure 3.11 et voyons le dernier état : **archivé** (5). On se rend compte qu'après passage par cet état, il y a destruction de l'entité séjour. Cela signifie que lorsque les données relatives au séjour d'un client sont archivées, par exemple sur disque, alors l'entité est détruite de la mémoire du système représentant le comportement "enregistrer client". En effet, lorsque le client à terminé son séjour dans l'hôtel, il n'est plus nécessaire de conserver inutilement les données relatives à ce client dans la mémoire. Si plus tard, il faut le reprendre, cela se fera à partir des archives.

On vient de présenter l'évolution de la mémoire du système grâce aux cycles de vie des objets. Les entités, objets de la mémoire, transitent d'un état (cercle) à un autre, au moyen des arcs orientés du graphe de transitions des états (cycles de vie).

Dans une dernière étape, nous allons présenter la façon dont se conjuguent les plans de systèmes, témoins du comportement des systèmes, et la mémoire qui leur est associée.

3.6. Diagrammes de transitions

Les plans sont faits, la structure conceptuelle statique et dynamique des données est terminée, nous pouvons passer à la dernière phase de conception : la spécification des diagrammes de transitions. Ces derniers représentent, selon EPAS, chacun des micro-comportements du système-cible. Nous définirons donc dans cette dernière étape la notion de scénario puis nous introduirons celle de diagramme de transitions, ainsi que les différents éléments entrant dans sa composition.

3.6.1. La notion de scénario

Nous avons introduit des modèles atemporels lors de la constitution des plans. Mais lorsqu'on veut décrire des systèmes en faisant intervenir la dimension temps, il faut élaborer des **scénarii** que nous représenterons par des **diagrammes de transitions**. De cette manière, nous pourrons déterminer l'ordre d'activation des différentes tâches d'un même plan. Nous pourrons réaliser cet ordre grâce à deux éléments : les états du système et les évènements issus de l'environnement externe mais aussi interne au système (représentés par des messages).

3.6.2. Diagramme de transitions

Tout comme les plans, les diagrammes de transitions sont composés d'un certain nombre de processus. Mais contrairement aux plans, l'ordre d'exécution des processus a de l'importance. Chaque diagramme représente un scénario déterminé pour lesquels il met en oeuvre un ensemble de tâches. Il permet enfin de visualiser les événements externes, issus de l'environnement, dont le but est l'activation d'une tâche, et aussi les événements issus d'une tâche vers l'extérieur.

Pour illustrer le concept, reprenons l'exemple de l'enregistrement d'un client (Figure 3.2). Ce plan est situé au dernier niveau d'abstraction et contient deux processus : "**Inscription client**" (P321) et "**Mise à jour du tableau des réservations**" (P322). On reprend, dans l'exemple le processus "Inscription client" et on en donne le scénario, au moyen du diagramme de transitions (Figure 3.16), tel qu'il se déroulerait dans un système non informatisé. On obtient : lorsque le client se présente à la réception, un

réceptionniste prend une feuille d'inscription et un stylo et note les informations contenues sur la carte d'identité du client.

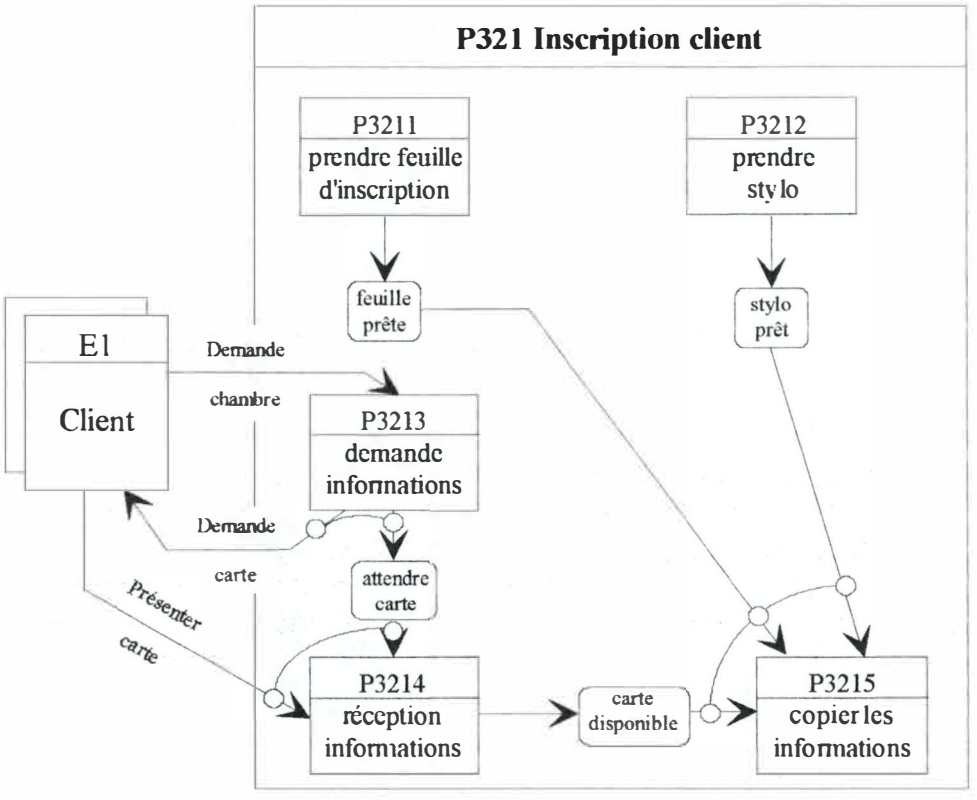

Figure 3.16 Illustration d'un diagramme de transitions

Présentons les concepts contenus dans les diagrammes de transitions ainsi que les éléments d'articulation entre ces concepts.

A Quels sont les concepts intégrés dans les diagrammes de transitions ?

Il existe entre les plans du diagramme EPAS et les diagrammes de transitions une ressemblance certaine. En effet, nous retrouvons dans ces derniers trois concepts : activité, environnement et état du système, qui possèdent une similitude avec les processus, environnements et accumulations. A cela, s'ajoutent deux associations. Les premières donnent la possibilité à une activité d'avoir un ou plusieurs liens avec les états du système : ce sont les événements internes et pour les secondes, elles nous permettent de créer l'axe activité-environnement, il s'agit des événements externes.

Dans les noeuds, on retrouve les concepts d'activité (ce sont les processus des plans définis au point 3.3.1) et d'environnement (déjà définis au point 3.3.3). On voit

apparaître un nouveau concept nommé l'**état du système** (ES) (Figure 3.17).

"*Les états du système sont représentés comme des objets de la structure conceptuelle des données; en fait ils représentent les états de ces objets ou des accumulations locales*" (MOULIN B. [1988]).

Un état du système est formé d'un rectangle à coins arrondis dans lequel on place le nom.

Figure 3.17 Structure d'un ES

Ainsi dans l'exemple présenté à la figure 3.16, on distingue quatre états du systèmes : **feuille prête**, **attendre carte, carte disponible** et **stylo prêt**.

On aura recours aux arcs orientés (flèches) pour représenter les événements. On distinguera les événements internes des événements externes par l'apport d'un nom sur les deuxièmes. On consultera le point 3.3.5 de ce chapitre pour connaître les règles de nomenclature des événements externes.

B Articulations des concepts

La conjugaison des différents concepts, représentés par un diagramme (Figure 3.18), apparaît comme un graphe orienté dont chaque noeud est représenté par un de ces trois concepts et les arcs, reliant les noeuds, ne sont rien d'autres que les événements (internes et externes).

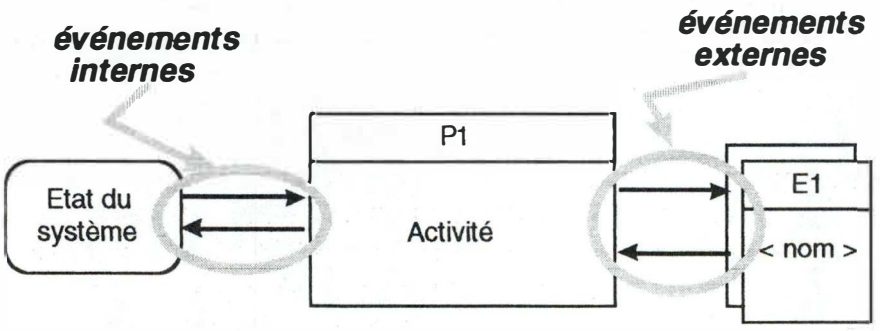

Figure 3.18 Evénements internes et externes

Bien qu'il y ait une similitude entre les événements (définis dans les scénarii) et les canaux et flux (définis dans les plans), la différence n'en est pas négligeable car dans les plans, ces associations n'avaient pour simple but que la création d'un lien accumulation-processus et processus-environnement. Alors que, grâce aux concepts d'événement et état du système, nous ajoutons, dans les diagrammes de transitions, la notion de temps. En effet, d'une part, les activités sont ordonnées dans les diagrammes, c'est-à-dire qu'elle ne s'exécuteront que séquentiellement et, d'autre part, la combinaison des événements issus des états du système et de l'environnement externe au système constitue des points de synchronisation; ce concept n'apparaît pas explicitement dans EPAS mais il convient de le souligner, c'est pourquoi nous en exposons la définition que nous en donne Bodart (BODART F. et PIGNEUR Y.[1989]) : "*Le mécanisme de synchronisation est utilisé pour représenter une situation dans laquelle il faut attendre la survenance d'une combinaison de deux ou plusieurs événements avant de provoquer une réaction particulière dans le système d'information. Il s'agit donc d'un mécanisme de coordination d'événements (...).*"

On peut voir dans le scénario de l'exemple "inscription client" (Figure 3.16) deux points de schynchronisation. Le premier en entrée du processus **"réception carte"** (P3214) et le deuxième en entrée du processus **"copier les informations"** (P3215). Cela signifie, dans le cas du processus **"copier les informations"** (P3215), que l'activation ne se fera qu'après création de l'état du système **"attendre carte"** et réception de l'événement **"présenter carte"**.

Nous avons vu les concepts contenus dans les diagrammes de transitions et la façon dont ils sont représentés. De plus, les conditions en entrée des activités sont formées par combinaison d'événements. Ces mêmes activités produiront en sortie des événements qui pourront être groupés formant les conditions en sortie. La représentation de ces conditions fait l'objet du point suivant.

3.6.3 - Activation et désactivation d'activités

La fin de la partie précédente a introduit la possibilité de combiner les éléments et, ainsi, créer des conditions en entrée (points de synchronisation) et en sortie d'activités. L'objectif de cette partie est de formaliser la représentation de ces conditions.

Tout d'abord citons Moulin (MOULIN B. [1988]) à propos de ces conditions :

"*Les événements en entrée d'une activité correspondent aux conditions d'activation de l'activité. Les événements en sortie d'une activité correspondent aux conditions de changement d'état du système après la mise en oeuvre de l'activité (désactivation). Les conditions portent sur les états et/ou les événements externes (en provenance ou à destination de l'environnement).*"

Comment représenter et exprimer des conditions entre les événements en entrée et entre les événements en sortie d'activités ?

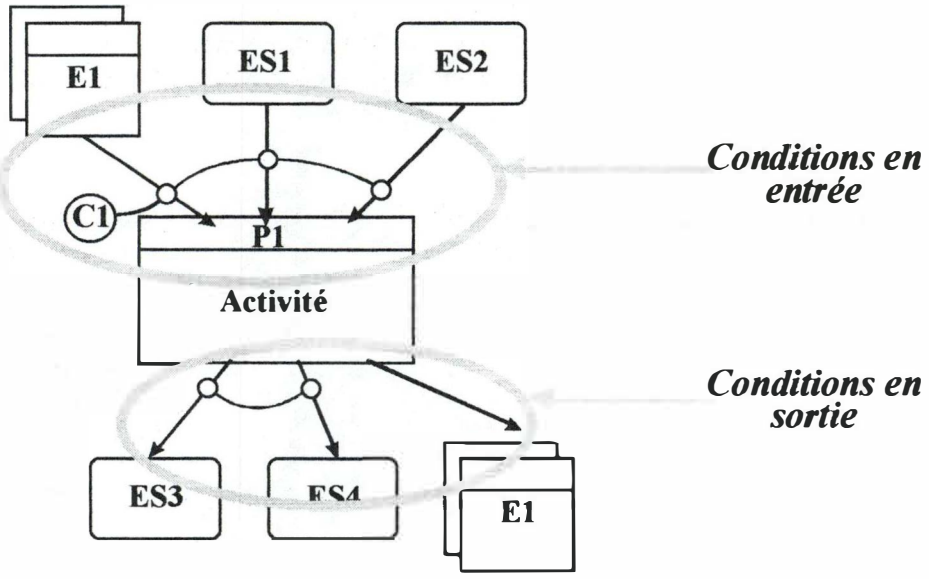

Figure 3.19 Structure des conditions en entrée et sortie

Nous voyons grâce au support graphique (Figure 3.19) la manière dont on doit représenter une condition. Celle-ci se signale par un cercle dans lequel on place le numéro de la condition. Il ne reste plus qu'à l'énoncer :

Condition en entrée :

c1 = ((evt 1 **ET** ES1) **OU** (evt E1 **ET** ES2)).

Condition en sortie :

(ES3 **ET** ES4) **OU** evt 2

Dans l'exemple présenté à la figure 3.16, on voit très nettement apparaître trois conditions, deux en entrée et une en sortie d'activités. On peut voir les événements internes et externes liés entre eux afin de former les conditions.

Les diagrammes de transitions ont permis de grouper les plans, la structure conceptuelle des données statique et dynamique. De plus l'introduction du temps, traduit par un séquencement des activités et des conditions sur les événements, permet de se rendre compte d'oublis commis lors de la construction du système, oublis par rapport aux spécifications du problème posé. Dans ce cas, nous aurons recours à une révision du système comme le décrit le point suivant.

3.6.4 - Révision des plans du système et de la structure conceptuelle de données

Comme nous l'avons vu, les diagrammes de transitions représentent la dernière étape de la phase d'analyse. Ils sont la représentation des scénarii décrivant le comportement du système-cible à réaliser. Ils permettent donc selon Moulin (MOULIN B. [1988]) de visualiser la "*cohérence des hypothèses de conception faites au cours des étapes de modélisation des plans et de la structure conceptuelle de données. (...). Cette étape de révision des plans et de la structure conceptuelle de données consiste à vérifier que les plans proposés au cours de la conception permettent de mettre en oeuvre des scénarii cohérents et satisfaisants pour les futurs usagers du système, interlocuteurs des analystes. Si des incohérences apparaissent, on modifiera les plans et/ou les scénarii jusqu'a ce que la simulation du comportement du système-cible à partir des scénarii soit satisfaisante.*"

On voit que le processus de modélisation par raffinements successifs proposé par la méthode EPAS permet un un "feed-back" (retour en arrière) afin de modifier les plans et/ou scénarii jusqu'au moment où le concepteur atteint le niveau de précision souhaité.

Nous aurons obtenu au terme de cette étape la forme définitive des plans et diagrammes de transitions du système-cible et de la structure conceptuelle de données. Le point suivant en donne un résumé.

3.7 - Vision globale des plans, de la mémoire et des scénarii

En guise de conclusion de cette phase d'analyse, donnons la syntaxe globale unifiant dans un seul diagramme les plans et les accumulations, la mémoire ou structure conceptuelle dynamique de données et les diagrammes de transitions.

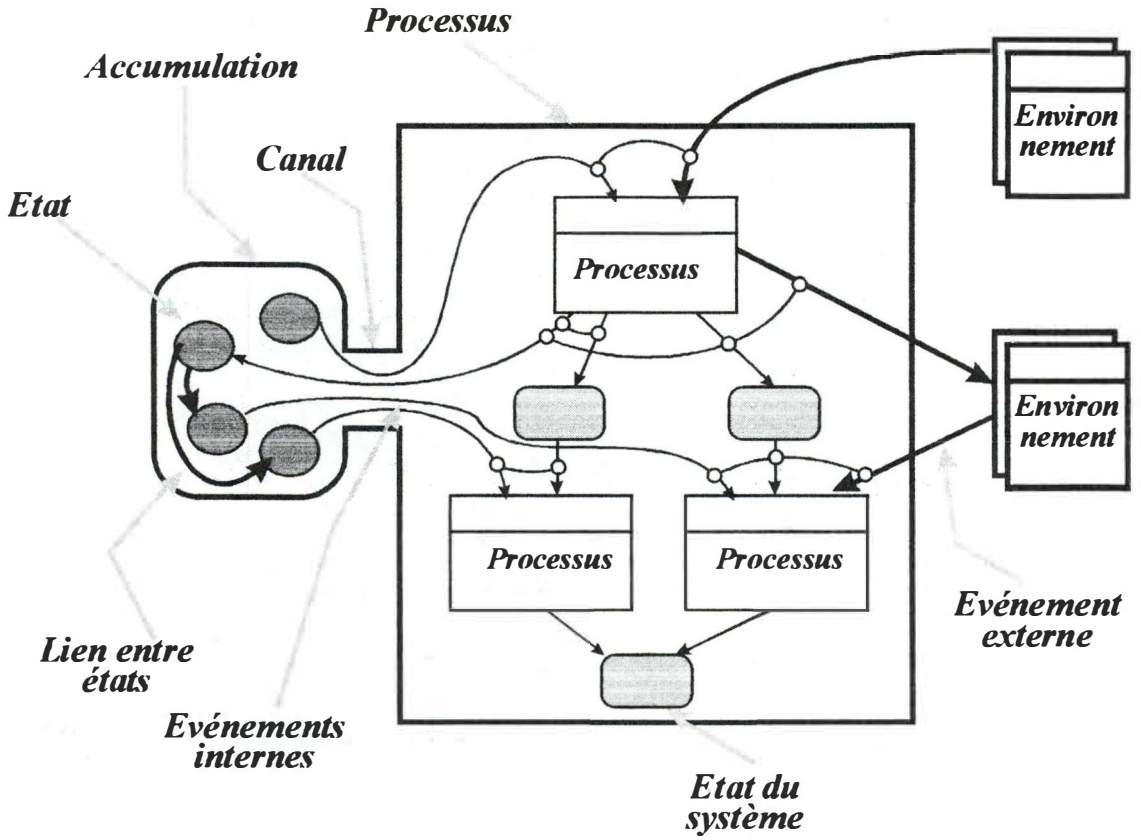

Figure 3.20 Illustration de la fusion des concepts

Une telle présentation montre bien l'intérêt des différentes étapes de la modélisation. En effet, on apperçoit les accumulations, processus et environnements identifiés lors de l'élaboration des plans. Nous voyons aussi que les événements internes, issus des états identifiés à l'étape des cycles de vie, parcourent les canaux situés entre les activités et les accumulations. Et pour terminer, nous pouvons apercevoir en détail grâce aux événements externes les interactions d'un processus avec son environnement. Grâce à ces événements, un séquencement dans l'activation des processus est rendu possible.

Section 4 - Conclusions

Nous avons d'abord montré dans ce chapitre qu'on pouvait modéliser le comportement d'un système au niveau conceptuel sous la forme de plans. Ceux-ci permettent de représenter de façon atemporelle les activités et les états (effectifs et potentiels) du système dans un contexte donné (contraintes d'environnement).

Nous avons par la suite représenté la mémoire du système sous forme de réseaux conceptuels qui décrivent l'évolution des connaissances que possède le système de son environnement et de lui-même, toujours indépendamment du temps.

La dernière étape de la phase d'analyse nous a permis de tenir compte de la dimension temporelle dans nos diagrammes. Ce sont les scénarii qui ont introduit cette notion de temps, car ils ordonnnent les activités au sein des diagrammes et placent des points de synchronisation afin que le déclenchement des activités ne soit pas anarchique.

A ce stade du développement, nous avons modélisé le comportement du système-cible visé et avons identifié les différents modules qui interagiront dans le système.

Nous avons présenté un outil puissant de planification d'un problème. Cet outil, la méthode de modélisation EPAS, permet d'isoler plusieurs sous-systèmes, et de déterminer leur coordinations. Sans doute, n'avons-nous répondu que partiellement au problème posé dans les premiers chapitres. En effet, si nous sommes en mesure de représenter les connaissances et les isoler en différents système ou sous-systèmes pour les implanter sur les différentes machines constituant un réseau, nous n'avons pas à ce stade répondu au problème de la négociation entre agents.

Dans le chapitre suivant, nous allons présenter une méthode permettant de modéliser la négociation entre agents dans un système distribué multi-agents. Le canevas de cette démarche est la méthode EPAS car elle convient à la schématisation des communications; encore faut-il trouver un moyen d'identifier les agents du système-cible. Comme on le verra, ce sont les méthodes orientées objets qui répondront à cette question.

CHAPITRE 4 : Démarche de modélisation orientée multi-agents

Section 1 - Introduction

Le chapitre 3 se termine et, avec lui, la présentation de la méthode EPAS, mise au point par Moulin (MOULIN B. [1988]). EPAS est une démarche de modélisation des interactions et des comportements de systèmes qui, de plus, se veut générale. Nous allons, dans notre cas, tirer profit de cette caractéristique et montrer comment EPAS peut être adaptée afin d'en faire une démarche modélisatrice de systèmes multi-agents.

Le but de ce chapitre est donc de présenter une méthode multi-agents pour la résolution de problèmes distribués, concepts déja énoncés dans le premier chapitre. Pour ce faire, elle débutera par la recherche des différents agents desquels il faudra modéliser la négociation. Comme le lecteur pourra s'en rendre compte, nous nous sommes fortement inspiré du domaine de la modélisation orientée objets pour la modélisation des interactions et des comportement des agents.

C'est donc une méthode hybride que nous proposons, construite par extraction de principes appartenant aux approches systémique et objet. Ainsi, nous disposerons d'une méthode rigoureuse mais néanmoins générale pour la modélisation d'un système multi-agents.

Ce chapitre présente la méthode en huit points conditionnée en amont par la méthode EPAS et en aval par un outil d'aide à la programmation. Afin d'illustrer chacun d'entre eux, nous recourrons à un exemple : le jeu FRIK, support à la vérification de la méthode. Ce jeu sera en partie présenté au fur et à mesure de ce chapitre.

Section 2 - Le Synopsis

Notre méthode, de résolution de problèmes pour un système distribué multi-agents débute par une étape qui consiste à décrire l'idée originale ou synopsis (de LOOZ-CORSWAREM G. ET JOURQUIN P., [1990]); un synopsis est "*l'idée originale sur laquelle s'étayera le scénario ou, si l'on en a déjà une vue plus précise, à exposer un bref résumé du scénario. Cette étape est naturelle car elle se trouve à la base de tout type de démarche de conception de scénario.*"

Le but est donc de donner une explication succincte des diverses tâches composant le scénario final. Comment dans notre cas allons-nous écrire le synopsis du jeu FRIK ?

Jackson (JACKSON M. [1983]) peut nous éclairer sur ce point. L'objectif est de "*modéliser la réalité*". Le message qu'il veut nous faire passer est que l'on va créer un modèle du monde réel par une simple "*description abstraite*" de la réalité. Nous aurons recours pour cela au principe d'abstraction, bien connu des concepteurs de génie logiciel, qui consiste à ne relever de la réalité que les concepts ayant leur pertinence au sein du modèle.

Sommerville (SOMMERVILLE I. [1989]) qui est du même avis, écrit : " *Le processus de planification est une création reposant sur la capacité, éducation, intuition et expérience du planificateur (...). Le planificateur doit proposer un nombre de planifications possibles, tester celles-ci, les itérer entre-elles jusqu'à ce qu'une solution acceptable soit trouvée.*

Ce processus doit démarrer quelque part et une approche suggérée par Abott(1980) et Booch(1987) est utile comme point de départ. Cette approche repose sur une brève et descriptive vue générale du système, exprimée dans le langage naturel et identifiant les noms (objets) et verbes (actions) pertinents."

Voilà ce que cela donne dans le cas du jeu FRIK :

Frik est un jeu de rôle permettant à plusieurs joueurs d'exercer des activités similaires à la réalité. Il faut tout d'abord que chaque joueur soit placé dans des conditions optimales de jeu. Il se verra attribuer, pour cela, une police d'assurance.

Chaque participant jouera à son tour, il lancera les dés, achètera et vendra des propriétés, des objets d'art, ou autres possibilités offertes par le jeu. Tout cela étant régi par les règles du jeu FRIK.

A tout moment, un joueur aura la capacité de négocier (vendre, acheter et/ou échanger) ses possessions avec un ou plusieurs joueurs. Il pourra aussi,

s'il le désire, manipuler ses séries (combinaison(s) particulière(s) de cartes de propriétés valorisant celles-ci), et avoir également accès à toute une série d'informations.

Chaque joueur sera réprésenté par un pion de couleur distincte des autres joueurs qu'il déplacera sur une plaque de jeu après avoir lancé les dés. Cette action l'amènera sur une des cases de la plaque de jeu déterminant un ensemble de tâches à réaliser avant de poursuivre sa route. Cette itération se déroulera jusqu'au moment où il ne restera plus qu'un joueur en lice.

Si le lecteur le désire, il se référera utilement aux annexes 1 où il trouvera une description extrêmement rigoureuse des règles du jeu FRIK.

Comme on peut le voir, certains mots du synopsis ont été soulignés. Ce sont en fait les *noms pertinents* de la spécification, décrits par Sommerville, et qui seront à la base de la constitution des agents du système distribué modélisant le jeu FRIK. Il est donc logique de constituer une section dont le but est de les décrire plus en profondeur; c'est à cette description des agents que s'attarde le point suivant.

Section 3 - Mise en évidence des agents et de leurs caractéristiques

La première étape permet d'exposer une approche succincte de la solution au problème posé; elle donne l'idée originale de la solution. Or Meyer (MEYER B. [1988]) a écrit : "*dans beaucoup de cas, le software est basé sur des aspects du monde réel qui ont leur pertinence pour l'application*". C'est ce qui à été fait dans le synopsis. Mais la totalité des tâches que devra effectuer chaque agent n'est pas extraite uniquement du synopsis, cela peut venir de l'expérience du concepteur, du cahier des charges, de ses multiples rencontres avec les utilisateurs finaux ou dans notre cas des règles du jeu que l'on trouve en annexe 1. On commencera par identifier les agents révélés dans le synopsis. Cela nous en donne une première identification, il faut ensuite spécifier leurs rôles, c'est-à-dire leurs tâches; celles-ci sont disponibles dans les règles du jeu FRIK. Notre tâche consiste à les répartir au sein des agents identifiés.

Comme nous le verrons au cours de la modélisation du jeu FRIK selon la méthode décrite dans ce chapitre, un agent pourra adopter plusieurs tâches distinctes comme par exemple initialiser ses données, jouer un ou plusieurs rôles au cours du déroulement du jeu ou encore exécuter sa tâche dans la clôture du jeu. Afin d'obtenir des spécifications claires, chacune de ces tâches sera modélisée au sein d'un module, nous aurons ainsi un module pour l'initialisation, un autre pour la clôture, et ainsi de suite. On entend par module, le regroupement, dans un ou plusieurs plans et par la suite un ou plusieurs diagrammes de transitions, de tous les processus nécessaires afin de réaliser entièrement une même tâche. De ce fait, un agent pourra être constitué de plusieurs modules.

Reprenons l'exemple du jeu et discutons des différents agents indentifiés. Nous décidons de ne pas reprendre l'agent **"plaque de jeu"**. En effet, l'entièreté des rôles que nous lui aurions attribués, nous l'avons répartie parmi toutes les cases. Ne nous étant plus d'aucune utilité, nous le retirons de la liste des agents.

Par contre un agent supplémentaire est identifié : le **meneur de jeu** (MJ). Son rôle est double. Tout d'abord il veille au bon déroulement de toutes les opérations d'entrée et de sortie d'argent de la banque, c'est-à-dire qu'il joue le rôle du grand argentier défini dans les règles du jeu. Ensuite, le MJ veille à l'exécution de toute une série de règles implicites: déterminer le tour du prochain joueur et lui donner l'ordre de jouer, gérer la mise en faillite d'un joueur, etc...

Une bonne description de cet agent nous est donnée par deLooz/Jourquin (de LOOZ-CORSWAREM G. ET JOURQUIN P., [1990]) : "*Concrètement, son comportement consiste à mettre en place le décor, à initialiser le domaine d'animation et à déclencher les comportements des agents, en respectant le scénario.*"

3.1 - Relevé des différents agents

- les joueurs (de 2 à 6)
- les cases (propriété, hasard, courtage, prêt, art, bourse, or, départ)
- le meneur de jeu
- les dés

3.2 - Identification du comportement des agents

3.2.1 - les joueurs

Le joueur est un agent qui est le seul à communiquer avec la personne qui joue, il en est également le porte-parole auprès des autres joueurs. Celui-ci est évidemment capable de remplir toutes une série de comportements tels que :

- recevoir plusieurs types de recommandations de son utilisateur
- communiquer avec les autres agents par des messages
- informer son utilisateur des messages qu'il reçoit des autres agents
- poser éventuellement des questions à l'utilisateur (celles-ci provenant du système)
- informe le système d'une éventuelle faillite

3.2.2 - case hasard

- suite à une requête du meneur de jeu, il initialise ses données
- lorsqu'un joueur tombe sur une case hasard, l'agent case hasard informe tous les joueurs du contenu de celle-ci
- il ordonne le paiement éventuel en fonction de la police d'assurance
- il peut devoir modifier le propriétaire d'une carte privilège suite à une négociation
- il informe les autres joueurs de ce qui se passe

3.2.3 - case courtage

- suite à une requête du meneur de jeu, il initialise ses données (mettre la valeur de la case à 0)
- lorsqu'un joueur tombe sur elle, la case courtage lui ordonne d'augmenter son portefeuille de la valeur qu'elle contenait, puis remet son contenu à 0

- il informe les autres joueurs de ce qu'il vient de se passer
- il peut répondre à une demande d'information

3.2.4 - case prêt

- suite à une requête du meneur de jeu, il initialise ses données (mettre les zones prêts à 0)
- lorsqu'un joueur tombe sur une telle case, il lui est permis de contracter un prêt
- si un joueur tombe sur une case prêt, et que celui-ci possède un prêt contracté sur cette case, il lui est permis de se faire rembourser
- il informe les autres joueurs de chaque opération prêt
- il peut répondre à une demande d'information

3.2.5 - case art

- suite à une requête du meneur de jeu, il initialise ses données
- lorsqu'un joueur tombe sur une case art, l'agent art lui permet d'acheter un tableau
- si un joueur tombe sur une case art, et que il possède un tableau , l'agent case art lui permet de le revendre
- il modifie le nom du propriétaire d'un tableau lorsque celui-ci a changé de main
- l'agent case art informe les autres joueurs de ce qui vient de se passer
- il peut répondre à une demande d'information

3.2.6 - case bourse

- suite à une requête du meneur de jeu, cet agent initialise ses données
- lorsqu'un joueur tombe sur une case bourse, l'agent lui permet d'acheter des actions
- si un joueur possède déjà des actions, l'agent lui permet de les revendre
- il modifie le nom du propriétaire d'actions lorsque celles-ci ont changé de main
- l'agent bourse informe les autres joueurs de ce qui vient de se passer
- il peut répondre à une demande d'information

3.2.7 - case or

- lorsqu'un joueur tombe sur une case or, l'agent or lui permet d'acheter un lingot; ensuite, après un lancement des dés, l'agent or ordonne l'encaissement adéquat au joueur
- l'agent or informe les autres joueurs de ce qui s'est passé
- il peut répondre à une demande d'information

3.2.8 - case départ

- lorsqu'un joueur tombe sur la case départ, ou passe sur celle-ci, le meneur de jeu augmente le fonds boursier de 100$
- ii signale toutes modifications du fonds boursiers à tous les joueurs

3.2.9 - case propriété

- suite à une requête du meneur de jeu, il initialise ses données
- lorsqu'un joueur tombe sur une case propriété, l'agent lui permet de l'acheter si elle est libre, et il envoie ensuite l'ordre de décaissement au joueur.
- l'agent propriété informe les autres joueurs de toutes actions exécutées
- si la case est occupée, l'agent ordonne au joueur de payer telle somme d'argent au propriétaire
- lors d'un achat libre ou semi libre, il ordonne au joueur de payer le courtage, et ordonne à l'agent case courtage d'augmenter son contenu de autant.
- si une propriété est semi-libre, le joueur doit payer le propriétaire; ensuite il peut acheter le morceau de propriété qui reste
- si le joueur n'achète pas la propriété sur laquelle il est tombé, une vente aux enchères est déclenchée pour vendre ce terrain
- il peut y avoir une modification de propriété lors d'une négociation
- il peut répondre à une consultation

3.2.10 - dés

- il reçoit une demande des joueurs
- il lance les dés

- il envoie la valeur obtenue par la somme des deux dés au joueur demandeur

3.2.11 - meneur de jeu

- il ordonne le positionnement des joueurs sur la plaque du jeu
- il ordonne l'initialisation de tous les agents cases
- il récolte de l'information sur chaque joueur (police d'assurance, nom)
- il envoie au joueur le résultat de la police d'assurance et la somme d'argent initiale
- il constitue la liste des joueurs avec un ordre de passage déterminé par les dés
- il détermine le joueur suivant, et lui donne l'ordre de jouer quand le joueur actif a fini son tour
- il reçoit le message de fin de tour des joueurs
- il reçoit les messages de faillite, et met la liste des joueurs courants à jour
- il informe les joueurs du résultat du jeu

C'est à cette étape que ressort le contrôle dans la résolution distribuée de problème, à savoir: coopération entre agents et leur organisation. De toute façon, la méthode est applicable quel que soit le choix adopté. Au vu de l'identification du rôle de chaque agent, le lecteur peut se rendre compte que notre choix est du style "chaque agent peut entrer en communication avec n'importe quel autre agent". Ce comportement a déjà été présenté au cours du chapitre 1.

En guise de résumé à cette section, nous dirons que l'identification des agents ne doit pas se faire d'emblée. En fait, il vaut mieux procéder à des remaniements des spécifications conduisant à une redéfinition des agents et de leurs rôles comme cela a été notre cas au cours de la modélisation, et au besoin il ne faut pas hésiter à réitérer ce processus plusieurs fois car pour résumer Booch (BOOCH G. [1991]), le processus de "design" ne se situe plus dans la traditionnelle procédure de modélisation dite de la "chute d'eau" (figure 4.1.A).

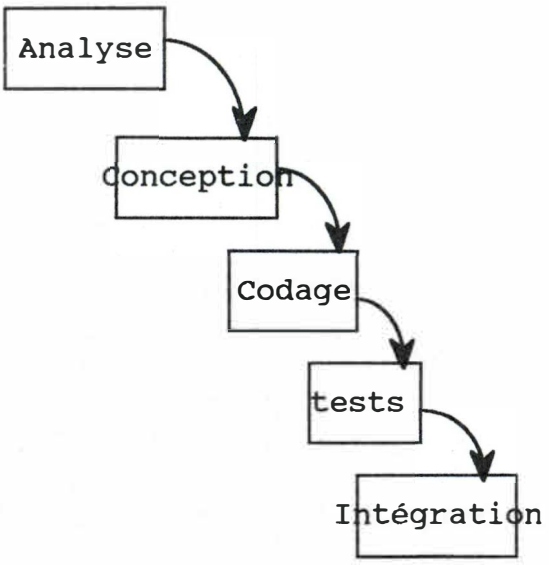

figure 4.1.A Modèle de la chute d'eau

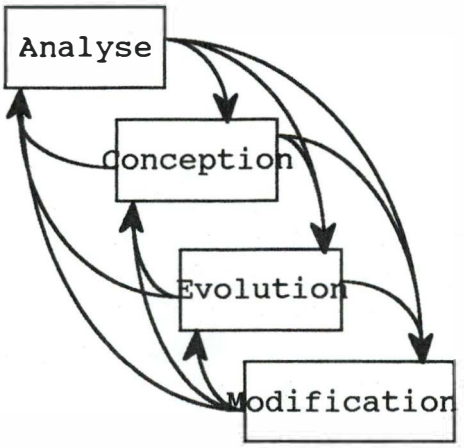

figure 4.1.B Modèle cyclique

En effet, il s'agit d'un processus cyclique dans lequel chaque étape de la conception peut entraîner des modifications à tous niveaux de la modélisation. Elle n'est donc ni "top-down" ni "bottom-up" (figure 4.1.B). Ainsi, au fur et à mesure de la réalisation du modèle, il est possible que de nouveaux agents soient identifiés, mais aussi, il sera parfois utile de décomposer un agent en divers modules. Avant de modéliser les interactions entre les agents, nous introduirons une étape intermédiaire qui découle du synopsis et met en évidence les comportements globaux du système-cible.

Section 4 - Plans généraux

Une grande partie des agents est mise en évidence; il reste à modéliser leur(s) comportement(s) et interactions à l'aide des plans et des diagrammes de transitions. Mais avant de se lancer dans cette modélisation, une étape intermédiaire nous paraît indispensable.

Il s'agit ici d'une étape qui nous est originale car non présente dans la méthode EPAS. Elle permet grâce aux diagrammes EPAS de modéliser la première approche dans la résolution du problème par raffinements successifs.

Ainsi grâce aux plans généraux, les grandes lignes des différents comportements sont décrits et seront détaillés par la suite. Le premier plan, le plus abstrait dans sa manière de modéliser les choses, sera tiré du synopsis. Son but étant un peu celui d'une table des matières, il donnera au lecteur une idée des différents éléments qui seront traités; il s'agit d'un fil d'Ariane indiquant au lecteur le chemin à suivre. Au niveau d'abstraction inférieur plusieurs plans globaux détaillant les comportements cités au niveau supérieur peuvent être introduits. Contrairement au plan du système des diagrammes EPAS, les plans généraux ne comportent aucune accumulation ni aucun environnement car ils ne font que décrire des comportements globaux. Il s'agit d'un fil conducteur à travers le scénario.

Il convient de spécifier la forme d'un plan général avant d'en donner un exemple. Etant donné qu'il s'agit d'un plan, nous reprenons le concept de diagramme de plan déja spécifié dans la méthode EPAS. Ce dernier comportera diverses boîtes que nous appellerons "descriptif" car ils sont là pour décrire des comportements généraux. Leur forme sera identique aux processus dont le rôle est similaire.

Nous pourrions ainsi exprimer le plan de ce mémoire où chaque descriptif, contiendrait le titre d'un chapitre. Au niveau inférieur, il y aurait autant de plans généraux que de chapitres énoncés dans le premier plan général; chacun de ces descriptifs comporterait le titre d'une section.

Illustrons ce produit par un deuxième exemple tiré des spécifications du jeu FRIK.

Le premier plan général est celui que nous tirons directement du synopsis. Ainsi ce plan (figure 4.2) est composé de trois grandes parties les **"initialisations"**, les divers comportements pouvant être activés durant la partie nous lui avons donné le nom de " **faire un tour** " et la troisième étant la **"clôture"** du jeu.

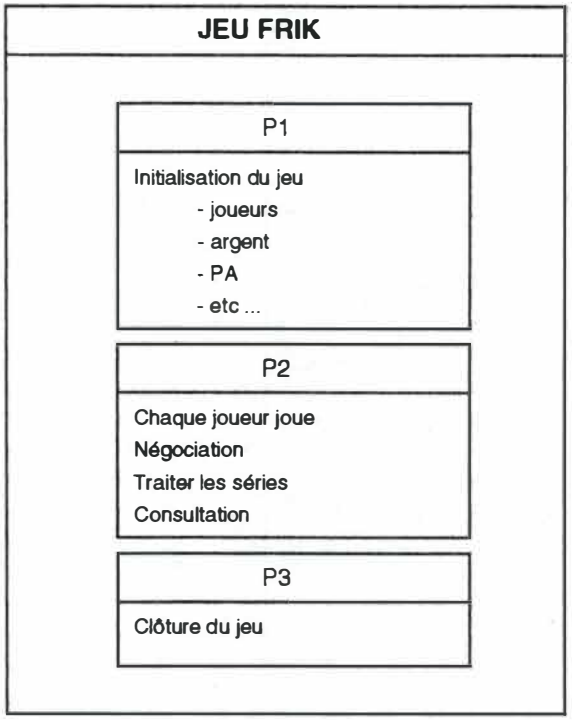

figure 4.2 Plan général

Nous passons ensuite au deuxième niveau d'abstraction où nous trouvons de façon détaillée d'une part la décomposition du descriptif "P1 initialisation" (figure 4.3) et, d'autre part, les divers comportements qui pourront être déclenchés à tout moment du jeu par un joueur (figure 4.4).

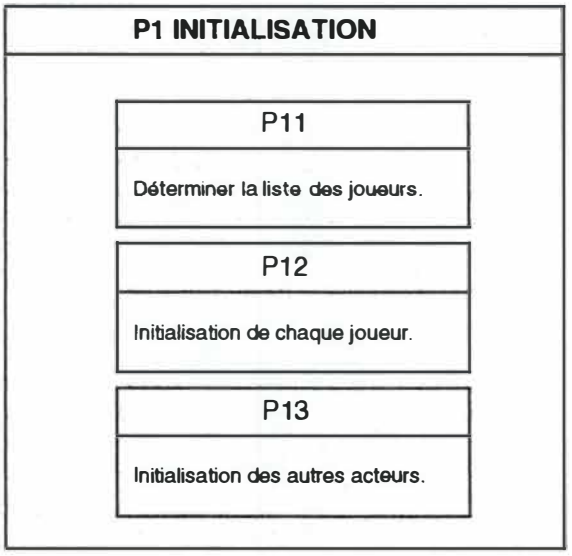

figure 4.3 Plan général : initialisation

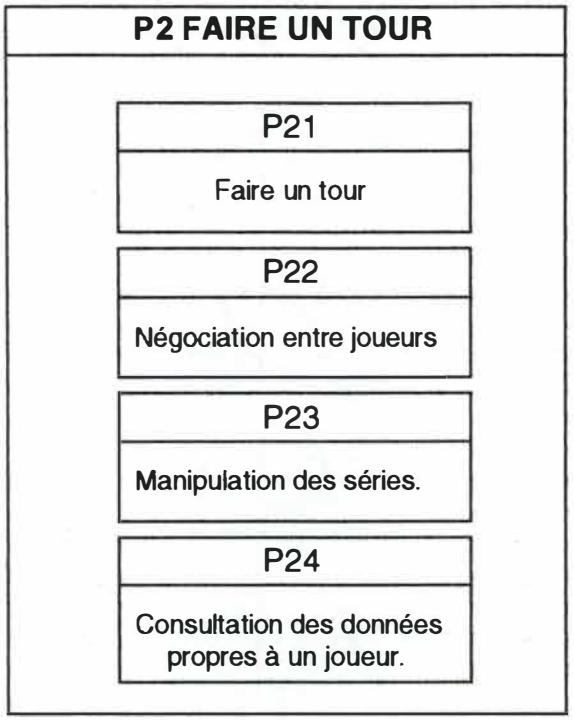

figure 4.4 Plan général : faire un tour

A ce stade, les grandes lignes du scénario sont dévoilées; on connaît le point de départ, les comportements à exécuter et le point de chute. Il s'agit à présent de détailler les rôles des agents identifiés lors de l'étape précédente tout en tenant compte des comportements qui viennent d'être exposés. C'est à cette tâche que s'attaque l'étape suivante.

Section 5 - Plans (diagrammes EPAS)

Pour mener à bien leur rôle, les agents développent un ensemble plus ou moins complexe de tâches composant leur comportement. De plus, lorsqu'ils en ressentent la nécessité, ils communiquent avec leur environnement, c'est-à-dire qu'ils échangent des informations via émission et réception de messages. Quant aux informations qu'ils receuillent, ils les placent dans leur mémoire comme cela a été expliqué au chapitre précédent.

L'étape précédente a permis de présenter les divers comportements de ces agents. Elle a servi d'étape de transition entre la découverte des agents, étape essentiellement à caractère objet, et les étapes suivantes dont le canevas est la méthode EPAS. Cette quatrième étape, quant à elle, est destinée à modéliser ces comportements au sein du système-cible.

Ce sont les plans qui modélisent au mieux les comportements des agents. En effet les processus décrivent les tâches; les flux expriment un échange de données entre l'environnement externe et l'agent; la mémoire est schématisée par les accumulations et, enfin, les canaux accèdent à cette mémoire (tous ces concepts empruntés à la méthode EPAS ont déja été définis au chapitre 3, point 3).

Mais avant de détailler les tâches des agents du jeu FRIK : les dés, le meneur de jeu, le joueur et les différentes cases, nous présentons les adaptations apportées aux concepts de la méthode EPAS dans le but d'obtenir une similitude avec les concepts d'agents et de messages spécifiés dans les chapitres 1 et 2 de ce travail.

Afin d'illustrer cette étape, on extrait des spécifications du jeu FRIK la tâche de clôture du jeu exécuté par le **MJ**. Dans un premier temps, on expose le plan de clôture du jeu. Ce plan sera critiqué au cours des différents points pour aboutir à la présentation d'un plan de clôture du jeu tel qu'il a réellement été implémenter.

Tout d'abord, énonçons en quoi consiste le comportement de clôture que doit exécuter le meneur de jeu:

> Lorsque le meneur de jeu doit clôturer le jeu, deux formes de clôture peuvent se présenter. Tout d'abord quand il n'y a pas assez de joueurs, cela ne vaut pas la peine d'entamer une partie. Il faut alors prévenir le demandeur d'une partie de jeu que celle-ci n'aura pas lieu car personnes n'est intéressé par une partie. Ensuite lorsque la partie se termine, il convient, dans ce cas, d'informer tous les joueurs du vainqueur.

Afin de savoir comment clôturer le jeu, le MJ doit prendre connaissance de la façon dont se termine la partie. Pour cela, il retire de l'accumulation **jeu** l'information qui le lui

apprendra. Puis dans un deuxième temps, il informe le ou les **joueur(s)** de la fin du jeu. Cela donne le plan de la figure 4.5.

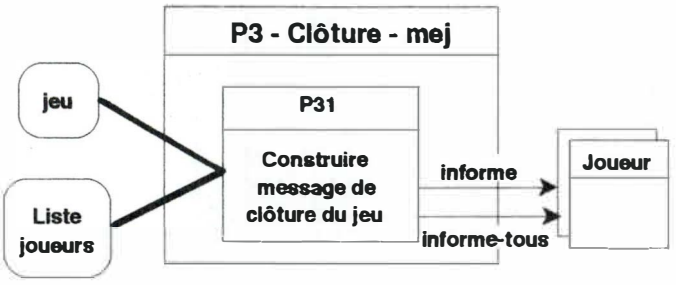

Figure 4.5 Plan de clôture du jeu version MJ

5.1. Message vs flux

Etant donné que, dans un système multi-agents tel que nous l'avons choisi, le mode de commmunication des agents sont les messages, ces derniers doivent être représentés spécifiquement.

La méthodologie EPAS met à notre disposition les flux en provenance et à destination d'un environnement. Or qu'est-ce qui peut mieux représenter un message qu'un flux puisque, l'un comme l'autre, représentent un transfert de données, une circulation d'informations entre deux systèmes en interaction. C'est pour cette raison que nous utilisons la notion de flux pour en faire, dans notre méthode, une représentation des messages.

Les messages sont représentés par la flèche qui est aussi la manière d'exprimer un flux d'informations entre deux éléments. De plus la flèche présente un avantage : elle montre immédiatement le sens de propagation de l'information.

5.2. Agent vs environnement

Si les diagrammes EPAS autorisent la modélisation des interactions qu'a un système particulier avec les environnements composant les autres systèmes qui l'entourent, il faut trouver le moyen de représenter la communication entre agents.

Si le concept de flux a été transformé en message, il ne reste plus qu'à renommer celui d'environnement en agent. En effet l'objectif de l'environnement, selon la méthode EPAS (chapitre 3, point 3.3.3.), est de représenter l'environnement extérieur avec lequel un processus échange de l'information. Or dans notre cas, l'environnement extérieur à un agent est composé de tous les autres agents. Ainsi, dans le jeu FRIK, l'agent **joueur** peut demander à l'agent **dés** de lancer les dés et de transmettre au

retour la valeur de chaque dé.

La structure de représentation graphique est identique à celle de l'*environnement* définie au chapitre 3. Cependant nous l'appelons ***Agent***.

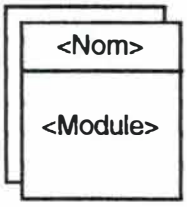

Figure 4.6 Structure d'un Agent

Règle de nomenclature

Dans la partie supérieure, l'identifiant est constitué du nom de l'agent. Quant à la partie inférieure elle sert à identifier une des rôles de l'agent, c'est-à-dire à identifier parmi les divers modules du même agent celui concerné dans une interaction avec un processus.

Ainsi si l'on prend le cas où un joueur s'arrête sur une case propriété, l'agent **case propriété** va être sollicité afin de régler le cas prévu par les règles. C'est-à-dire que l'agent **case propriété** ordonnera au joueur de payer les revenus de la propriété aux différents propriétaires, s'il en existe; puis il convient de prévenir les autres joueurs de la (les) transaction(s) qui vienne(nt) de s'effectuer. C'est ce que présente la figure 4.7.

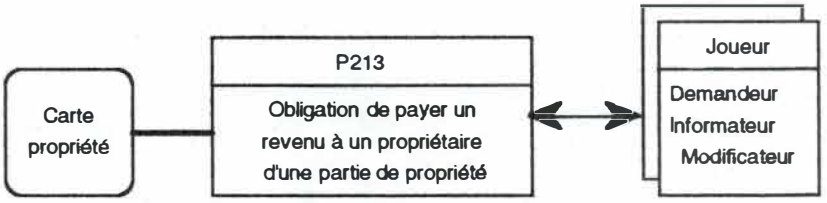

Figure 4.7 Illustration de la représentation d'un agent

Cette représentation présente les avantages suivants :

1) Cela permet au fur et à mesure de l'évolution de cette étape, de ne plus devoir tenir une liste des numéros identifiants, déjà attribués aux agents, dont le rôle est d'empêcher que des numéros identiques ne soient donnés à des agents différents. Ainsi dans l'exemple de la figure 4.8, nous distinguons les agents Art, Or, Serveur et Dés sans devoir nous soucier du numéro de chaque agent contrairement à ce qui était préconisé par la méthode EPAS.

2) Cela donne une meilleure lisibilité des diagrammes EPAS pour deux raisons. Tout d'abord, on connaît directement le nom de l'agent émetteur ou récepteur d'un message, ce nom est inscrit dans la partie supérieure de l'agent. Deuxièmement, si cet agent est composé de plusieurs modules, c'est-à-dire plusieurs tâches distinctes, on connaîtra immédiatement celles concernées. Cela présente une fois encore un gain de lisibilité si le nom des tâches est bien choisi. On peut en voir un exemple à la figure 4.7 dans lequel les modules **Demandeur**, **Informateur** et **Modificateur** de l'agent Joueur sont sollicités par le processus P213.

3) Cela apporte plus de clarté aux diagrammes de plans et par la suite aux diagrammes de transitions. En effet, lorsqu'un processus doit émettre un même message à destination de plusieurs agents, en lieu et place de cette nuée d'agents, il n'y en aura qu'un seul. Cette rationalisation n'est possible que si l'on a recours à une astuce.

Afin d'illustrer cette astuce, on prend l'exemple présenté aux figures 4.8 et 4.9 extrait des spécifications du jeu FRIK. Dans le processus **P13** de la partie initialisation (Figure 4.8), on peut voir que le MJ envoie quatre fois un même message : **lancez-votre-initialisation**, aux agents : **serveur**, **dés**, **case or**, **case propriété**, etc ... Dans le même plan (Figure 4.9), on ne distingue plus qu'un seul message partant du processus P13 à destination d'un seul agent.

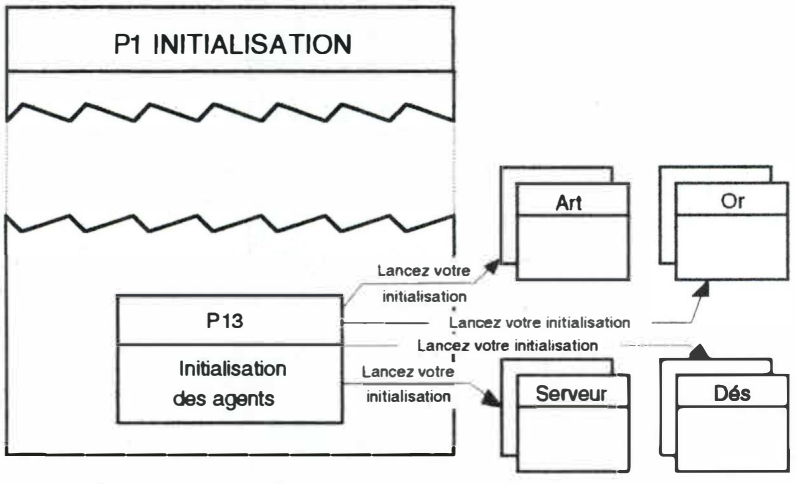

Figure 4.8 Plusieurs agents, plusieurs représentations

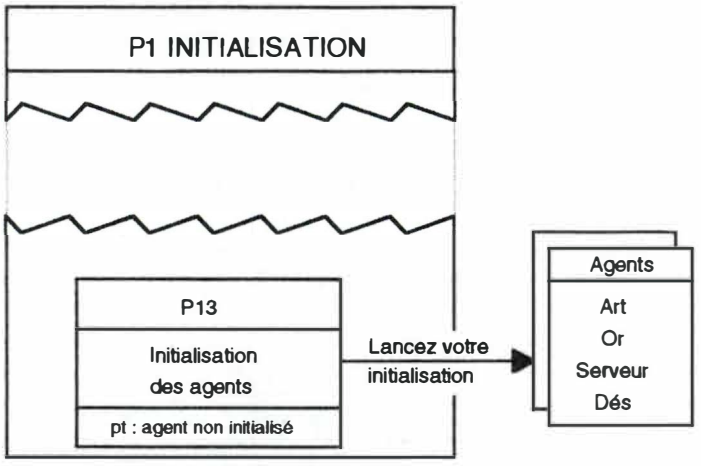

Figure 4.9 Plusieurs agents, une seule représentation

L'astuce utilisée est la suivante :

A l'endroit ***<Nom>***, nous plaçons le mot **Agents**, cela pour indiquer qu'il y a interaction entre plusieurs agents du système distribué; dans la partie inférieure de l'agent, on inscrit les divers agents récepteurs du message. Ainsi dans le plan **P1**, il y a un agent ***Agent*** englobant les agents serveur, dés, case or, case art.

Il faut remarquer la structure particulière du processus P13 de la figure 4.9. La partie inférieure est destinée à montrer que le message doit être émis pour chaque agent. Cela signifie qu'au niveau physique quatre messages seront émis sur le réseau mais, d'un point de vue de la représentation, il n'y a plus qu'un seul message avec un seul agent. Les diagrammes y gagnent en clarté et en lisibilité.

5.3. Interactions multiples entre deux agents

Il sera parfois nécessaire d'exprimer un échange intense de messages entre deux agents, nous appelons cela une *négociation*, ce concept est de même nature que celui présenté au chapitre deux mais la différence tient dans le nombre d'agents participant à une négociation. En effet au chapitre deux, plusieurs agents pouvaient participer, tel était le cas du réseau de contrats, alors que dans notre cas la négociation ne se fait qu'entre deux agents: elle schématise un dialogue. Afin de l'exprimer visuellement, nous aurons recours à l'arc (flèche) bidirectionnel (Fig 4.10). Ce n'est qu'au niveau inférieur que seront détaillés les messages échangés par les agents et les proccessus traitant ces messages.

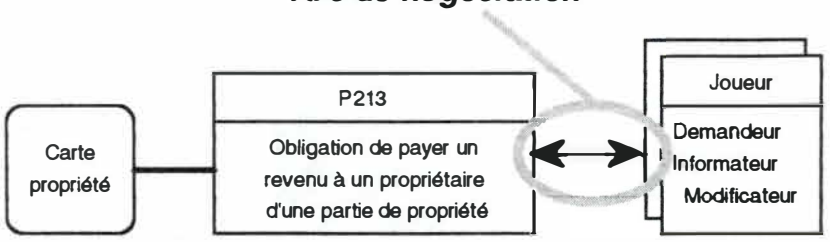

Figure 4.10 Illustration d'une négociation

Règle de nomenclature

Une négociation sera représentée par une flèche bidirectionnelle. Il n'est pas nécessaire de la nommer car elle englobe un ensemble de messages de noms différents. Elle ne sera détaillée qu'au niveau inférieur.

On peut voir dans la figure 4.10 un bel exemple de négociation. En effet, lorsqu'un joueur tombe sur une case propriété, un échange intense de messages a lieu afin de régler les différents revenus des joueurs possédant une partie de la propriété. Cela entraîne des décaissements des avoirs du joueur et un encaissement de revenu par le propriétaire.

5.4. Découverte d'agents/de modules à un agent

Cette étape d'élaboration des plans permet de découvrir de nouveaux agents non encore révélés lors de l'étape d'identification des agents (chapitre 4, section 2). On peut aussi, pour chaque agent, dégager plusieurs modules.

Ces découvertes entraînent des modifications dans les plans de niveaux supérieurs ou une restructuration de la liste des agents. Il ne sera pas rare de devoir opérer de la sorte dans une optique de conception apparentée aux méthodes orientées objets.

Afin d'illustrer cette étape, on extrait deux exemples du jeu FRIK. Ainsi, lors de la modélisation, nous avons découvert qu'il serait bon d'introduire deux nouveaux agents, le **serveur** et l'**utilisateur**. La tâche du serveur est de gérer les connaissances partagées entre plusieurs agents. En effet, nous avons remarqué que certaines accumulations (plateau du jeu, fonds boursier, carte courtage et liste joueurs) devaient être distribuées entre plusieurs agents. Nous avons isolé deux modes de résolution de se problème: la multiplication de ces entités (chaque agent en possède une copie) et la technique d'un serveur.

Soit nous multiplions ces accumulations mais, dans ce cas, lorsqu'un agent modifie le contenu d'une accumulation, il doit émettre un message afin que tous les autres

agents agissent de même dans le but de posséder une réplique exacte de cette accumulation. Avec une telle optique de fonctionnement, nous obtenons vite des goulots d'étranglement sur le réseau ralentissant considérablement le déroulement des tâches des agents. Nous optons donc pour la deuxième solution qui consiste à créer un agent supplémentaire, le **serveur**.

Grâce à la technique du serveur gérant les accumulations communes à plusieurs agents, nous éliminons le risque de voir travailler ces agents sur des données non mises à jour, ainsi que celui du goulot d'étranglement des lignes physiques. Les agents ne peuvent désormais avoir accès aux données que via ce nouvel agent.

Un deuxième agent a été indentifié: l'**utilisateur**. Le rôle de cet agent est très simple, il avise la personne jouant une partie de jeu en présentant des informations à l'écran et il saisit les réponses introduites par l'utilistaeur humain faisant suite aux questions posées par l'agent joueur. Il sert donc d'interface entre l'agent joueur et la personne.

Il faut les ajouter à la liste des agents et leur attribuer les tâches qu'ils devront accomplir.

5.4.1. Identification du rôle des agents

5.4.1.1. serveur

- il communique aux autres agents les éléments d'information contenus dans les accumulations plateau du jeu, fonds boursier, case courtage et liste joueurs

- il veille à la mise à jour des accumulations communes aux différents agents suite à un message de modification qu'il reçoit de ces même agents

5.4.1.2. utilisateur

- il affiche à l'écran les message informatifs ou interrogatifs que l'agent joueur lui ordonne d'afficher.

- il saisit les informations introduites, par la personne participant à une partie de jeu, qu'il transmet au joueur.

D'un autre côté, la planification des divers comportements du joueur a permis de mettre en évidence plusieurs modules : **informateur**, **consultation**, **faire tour**, **modificateur**, **fonctions**.

Ces modifications et découvertes autorisent une reformulation du plan de clôture de la figure 4.5.

5.4.2. Reformulation du plan de clôture

L'accumulation **liste joueurs**, contenant un ensemble d'informations faisant partie des connaissances partagées entre plusieurs agents, disparaît du plan de la figure

précédente. Le MJ doit dès lors négocier avec le **serveur** afin d'obtenir la liste des joueurs. De plus le joueur se trouve composé de plusieurs modules parmi lesquels on trouve celui intitulé **informateur**. Ces modifications apportent une refonte du plan de clôture du jeu (figure 4.5) qui débouche sur les figure 4.11. et 4.12.

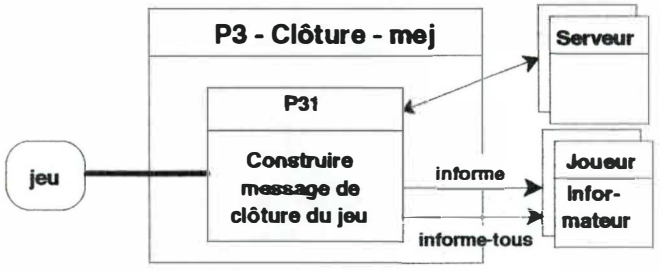

Figure 4.11 Plan de clôture version MJ

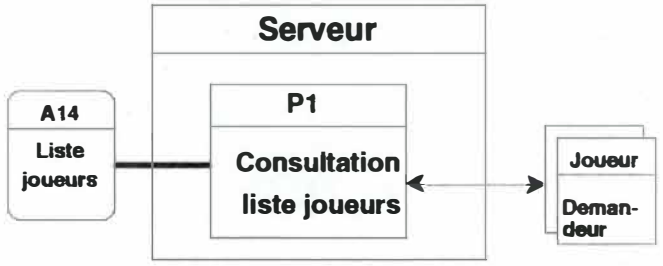

Figure 4.12 Plan de consultation de la liste des joueurs version serveur

Nous avons pu schématiser les divers comportements des agents grâce aux diagrammes EPAS, ce qui nous a permis de découvrir deux nouveaux agents, le serveur et l'utilisateur, et grouper certaines tâches dans un même module comme on l'a fait pour le joueur. Les connaissances ont été réparties entre les agents et ont été représentées par les accumulations. Il faut à présent donner une structure à la mémoire de chaque agent, c'est-à-dire exprimer la manière dont on structure les accumulations.

Section 6 - Structure de données

L'étape précédente a permis de modéliser les tâches de chaque agent grâce aux plans et les accumulations ont rendu possible la visualisation de leur mémoire. Il convient à présent de spécifier la forme de leur connaissance au sein du système. C'est l'étape de formulation de la structure conceptuelle des données qui permet d'organiser les informations composant la connaissance des agents.

Cette étape, comme on l'a vu au chapitre 3, permet de donner une structure à l'organisation de la mémoire des agents chargée d'enregistrer et retrouver des éléments d'information. Selon cet objectif, on identifie l'ensemble des attributs composant chaque accumulation. Ensuite, on crée un réseau conceptuel en liant ces accumulations entre elles, on obtient la *structure conceptuelle des données*.

Cette étape sert aussi à partager les connaissances entre les agents: il faut entendre par là une répartition de la connaissance entre quelques agents ou entre tous les agents. Il y a donc soit une concentration de la connaissance d'où résulte une multiplicité de messages sur le réseau provenant des agents ne possédant aucun élément d'information, soit une distribution de la connaissance allégeant du même coup les lignes de transmission de messages. Cela a déjà fait l'objet d'un point au chapitre deux.

Un exemple de réseau conceptuel de données est présenté à la figure 4.13. Elle montre la structure relative à une négociation.

Une transaction de négociation ne peut porter que sur un seul objet. Ainsi le joueur pourra négocier un tableau, une propriété, des actions ou un privilège se trouvant en sa possession. La personne initialisant la négociation fera une proposition de vente dont la valeur est enregistrée dans l'attribut "*prix*". Par la suite, les différentes personnes désireuses d'acquérir l'objet feront leur offre. Seule l'offre la plus avantageuse aux yeux de l'initiateur sera retenue. La meilleure des propositions d'enchères est mémorisée dans l'attribut "*Proposition-enchère*".

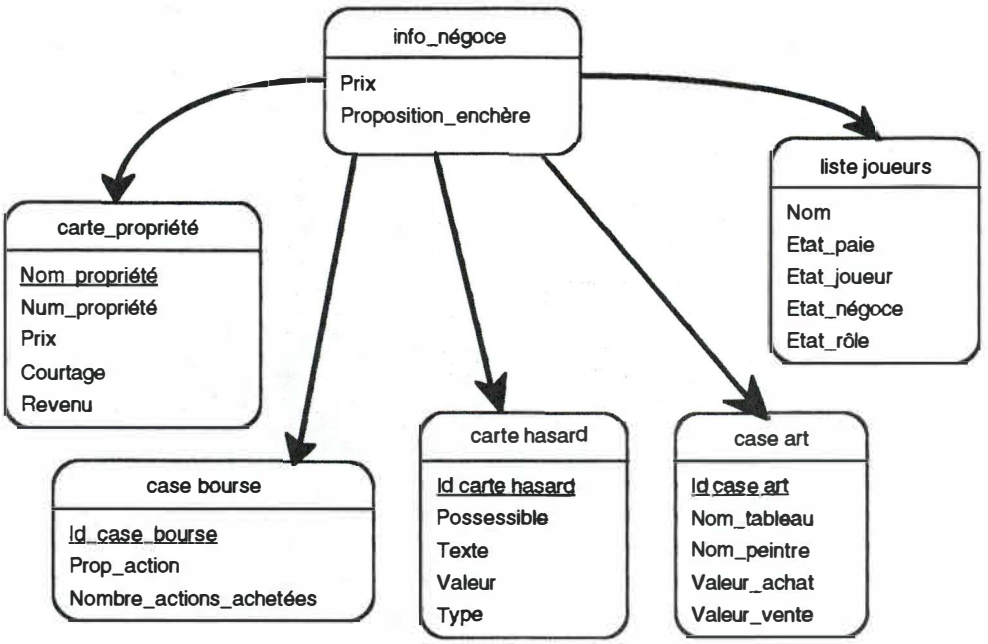

Figure 4.13 Structure conceptuelle d'une négociation

Il ne faut pas oublier la raison qui nous pousse à élaborer ce travail. Nous sommes à la recherche d'un méthode permettant de résoudre un problème distribué suivant une approche multi-agents, c'est-à-dire qu'il nous faut centrer chacune des étapes de la modélisation sur l'agent. Selon la méthode EPAS, la structure conceptuelle intègre au sein d'un même réseau conceptuel plusieurs entités appartenant à des environnements différents, ces concepts ayant été définis au chapitre 3. Or dans les systèmes distribués multi-agents, soit que quelques agents possèdent toute la connaissance, soit que chaque agent possède sa base de connaissance, partie de la connaissance globale, qui lui permet de raisonner sur un problème ou sur les intentions d'autrui (cfr. Chapitres 1 et 2); pour notre part, nous choisissons la deuxième possibilité.

Ainsi dans la figure 4.13 l'entité **info_négoce** constitue une partie de la mémoire de l'agent **Joueur**, tout comme **carte_propriété** est à **Propriété**, **case_bourse** est à **Bourse**, **carte_hasard** est à **Hasard**, **carte_art** est à **Art** et **liste_joueurs** est au **Serveur**.

Il convient donc de rendre à chaque agent ce qui lui revient de droit, c'est-à-dire qu'il faut trouver une méthode permettant de décomposer ce réseau conceptuel en sorte de faire disparaître les liens de références entre entités. On trouvera dans (HAINAUT J.-L., [1986]) un ensemble de règles de décomposition. En appliquant ces règles au réseau conceptuel, on obtient, pour la structure de données **info_négoce**, une structure (Figure 4.14) dont le formalisme de présentation est emprunté à Hainaut (HAINAUT J.-L., [1986]). En appliquant les règles de décompositions proposées, nous maximisons l'indépendance de chaque agent grâce au minimun de redondance obtenu

entre entités.

En appliquant ces règles de décomposition à l'entité info négoce du réseau de la figure 4.13, on obtient l'entité décrite à la figure 4.14. Afin de symboliser l'objet, d'identifier le propriétaire actuel de cet objet et la personne émettant l'offre la plus alléchante, l'entité sera constituée respectivement des attributs "***Type_objet***", "***Propriétaire***" et "***Offrant***". Il reste à placer le champ "***Etat*** " qui est nécessaire pour montrer les différents stades par lesquels passe la structure info_négoce lors du traitement d'une négociation.

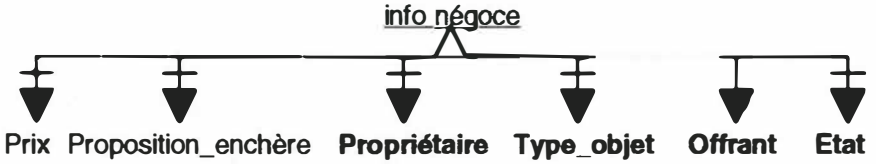

Figure 4.14 Structure de l'entité *info_négoce*

Nous avons réussi à isoler l'entité info_négoce et à déterminer chacun des attributs composant sa structure. De cette manière, l'agent Joueur peut poursuivre son raisonnement lors d'une négociation, en extrayant de sa mémoire les éléments d'information nécessaires en cas de négociation. Il ne doit plus échanger de message avec les autres agents; de même, il ne doit pas attendre les réponses des autres agents afin de poursuivre sa tâche.

La structure de la connaissance de chaque agent est identifiée de façon identique. Mais on remarque que certaines entités possèdent l'attribut "**Etat**" parmi leurs attributs alors que d'autres ne l'ont pas. Cela signifie que les agents, qui possédent de telles entités, voient leur mémoire évoluer. Le but de la section suivante est donc de rendre compte de cette évolution en présentant un moyen de représentation de l'évolution des connaissances des agents.

Section7 - Cycles de vie

Dans l'étape précédente, nous nous sommes attardés à présenter une méthode de structuration des connaissances des agents. Nous sommes arrivés à la conclusion que certains agents ont une mémoire évolutive. Cette section présente la façon de modéliser cette évolution.

Le chapitre précédent a montré que les cycles de vie servent à schématiser l'évolution de la mémoire du système. Cette mémoire transite par différents états qui sont représentés par un graphe de transitions d'états, c'est-à-dire un graphe orienté déterminant l'ordre de passage d'un état à un autre.

Ainsi, si on reprend l'exemple énoncé dans l'étape précédente (Figure 4.14), on peut se demander quels sont les états par lesquels la mémoire de l'agent joueur passe lors d'une négociation ? La réponse à cette interrogation vient en grande partie de la connaissance que le concepteur a d'une négociation mais, encore une fois, il est possible de découvrir de nouveaux états lors de la réalisation des diagrammes de transitions entrainant une modification du cycle de vie. Nous exposons ci-après le déroulement d'une négociation telle qu'elle apparaît après l'étape de réalisation des diagrammes de transitions :

Lorsqu'elle est créée, l'entité *info_négoce*, acquiert l'état **"NON ENGAGEE"** afin de spécifier qu'il n'y a encore aucune négociation en cours. Par la suite, un joueur, le "*demandeur*", va solliciter l'exécution d'une négociation portant sur un objet qu'il possède. Les initialisations vont se dérouler et arriver à leur terme; l'état devient **"FIN INIT"**. Grâce à cela, le meneur de jeu pourra débuter la négociation en signifiant à la première personne qu'elle peut fixer son offre. La négociation est alors en cours et l'état de l'entité *info_négoce* est placé à **"EN COURS"** tant que plusieurs négociateurs restent en concurrence. Au moment où il ne reste plus qu'un négociateur, l'état devient **"FIN TRT"**. Le meneur de jeu prend alors le relais et veille au bon déroulement d'une fin de négociation, c'est-à-dire qu'il envoie un message au "demandeur" afin de savoir s'il est d'accord sur les termes finaux de la négociation. Celui-ci répond à la requète du meneur de jeu par une confirmation ou une infirmation. La négociation reprend mais cette fois entre le "demandeur" et le dernier négociateur. L'état redevient **"EN COURS"** et s'il y a eu confirmation, on assistera au transfert de l'objet. Après cela la négociation se termine ce qui déclenche la transition à l'état **"NON ENGAGEE"**.

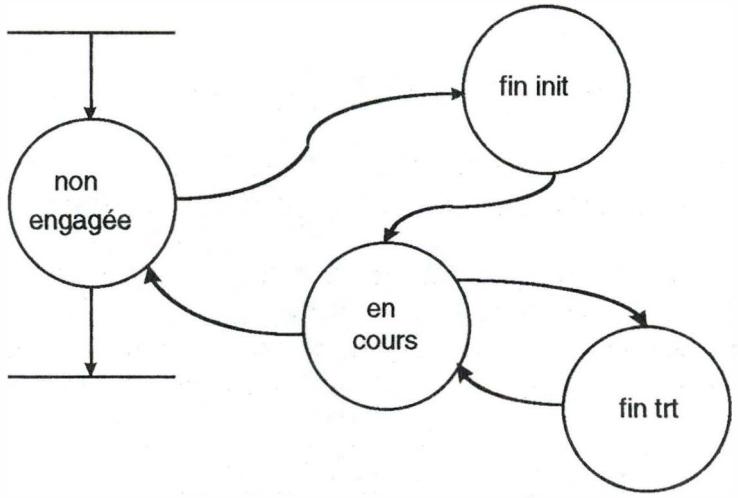

Figure 4.15 Illustration du cycle de vie de l'entité info-négoce

Nous introduisons à ce point un deuxième exemple qui présente un problème auquel nous avons été confronté lors de la modélisation du jeu FRIK selon notre démarche. Le problème apparaît lorsqu'une entité est constituée de plusieurs attributs d'états. En effet, il est possible de se trouver dans une telle situation suite aux concaténations d'accumulations et à la décomposistion du réseau conceptuel des données de l'étape précédente.

Prenons l'exemple de **liste_joueurs**, celle-ci dispose de quatre états: **Etat_joueur** qui informe le meneur de jeu de l'état du joueur à tout moment du jeu, **Etat_négoce** qui permet de savoir si un joueur est en négociation ou non, **Etat_rôle** qui présente le rôle d'un joueur négociateur dans une négociation et **Etat_paie** qui autorise le joueur à ne pas payer de revenu s'il possède la carte de privilège "exemption de payer".

Après avoir analysé la situation, on peut se rendre compte que ces quatre états ont été introduits afin de mener à bien les différentes tâches du meneur de jeu. Ainsi, "Etat_joueur" permet le bon déroulement de la tâche "faire un tour", tout comme "Etat_négoce" et "Etat_rôle" sont à "négociation" et "Etat_paie" concerne le paiement de revenu dans "faire un tour" lorsque le joueur tombe sur une case propriété. La description complète de chacune de ces tâches se trouve dans les annexes.

Nous proposons la solution qui consiste à présenter trois cycles de vie. En effet, le cycle de vie, comme on l'a déjà dit, schématise l'évolution de la mémoire de l'agent. Si sa mémoire évolue, c'est parce qu'il développe divers comportements qui ont pour résultat une modification de ses connaissances. De plus, les comportements développés par l'agent sont autonomes les uns des autres; c'est-à-dire que si au cours d'une partie, le joueur décide, par exemple, d'entamer ou de participer à une négociation, il peut le faire à tout moment.

Cette décomposition en plusieurs cycles de vie nous autorise à dire que l'on gagne en clarté et lisibilité des schémas exposés. Le lecteur peut s'en apercevoir en regardant les figures 4.16 à 4.18.

Etat_joueur

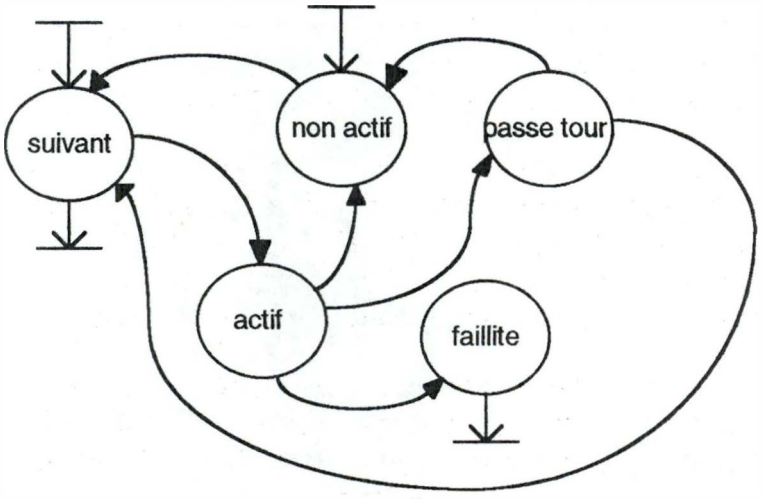

Figure 4.16 Cycle de vie de *liste_joueur*

Etat_négoce + Etat_rôle

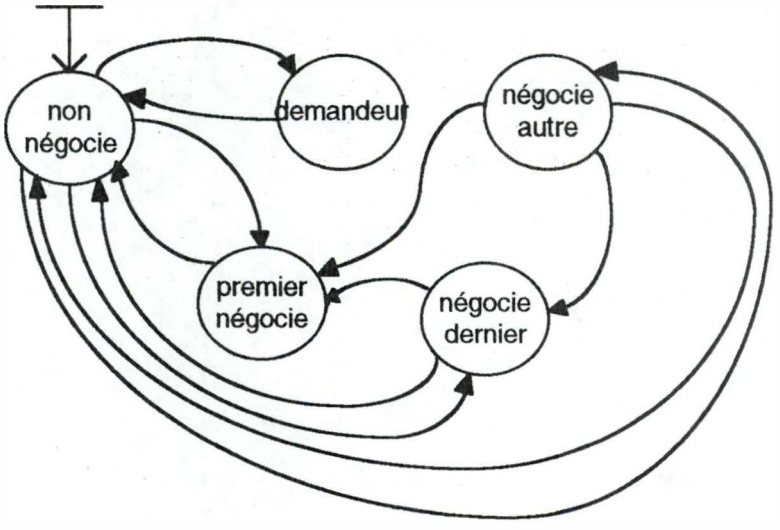

Figure 4.17 Cycle de vie de *liste_joueurs*

Etat_paie

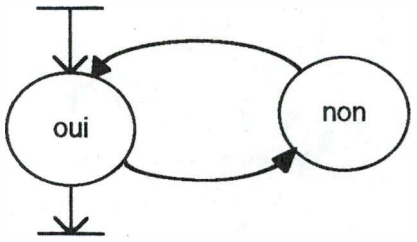

Figure 4.18 Cycle de vie de *liste_joueurs*

Après avoir détaillé la structure de données de la mémoire ainsi que son évolution pour chaque agent et résolu le problème de multiplicité des états d'une entité, il faut présenter dans une dernière étape la combinaison des plans et de la structure de données statique et dynamique. La section suivante introduira en plus une dimension temporelle par un séquencement des tâches d'un comportement, c'est-à-dire en définissant l'ordre d'activation des processus. Cet ordre d'activation est rendu possible en partie grâce aux transitions d'états de la mémoire qui viennent d'être définis pour chaque agent.

Section 8 - Diagrammes de transitions

Jusqu'à présent, les différentes étapes de la démarche proposée ont permis d'identifier les agents du système distribué ainsi que leurs tâches. Les plans ont permis de mettre en évidence les éléments d'information constituant la connaissance de chaque agent dont on a décrit par la suite la structure et aussi l'évolution grâce aux cycles de vie.

A partir du moment où l'on veut décrire le comportement d'un système en introduisant la dimension temporelle, on élabore des scénarii exposant les comportements des systèmes. La représentation de ces scénarii se fait au moyen des diagrammes de transistions (voir chapitre 3).

Un scénario permet de déterminer l'ordre d'activation des tâches. Il crée des points de synchronisation conjuguant des événements internes ou externes avec des états du système constituant de cette manière les conditions en entrée ou en sortie d'activités.

8.1. Applications des diagrammes de transitions aux agents

Les diagrammes de transitions tels qu'ils ont été définis dans le chapitre 3 sont d'application lorsqu'il s'agit de gouverner les interactions entre systèmes. Dans un premier temps, nous allons adapter les différentes notions vues au chapitre 3 afin de les utiliser dans des systèmes multi-agents.

Les concepts énoncés au point 3.6 du chapitre 3 entrant dans la composition d'un diagramme de transitions sont : l'activité, l'environnement, les états du système et les événements interne et externe. Les notions d'environnement, d'état du système et d'événement externe sont à revoir, les activités et événements internes ont, pour leur part, déjà été définis au chapitre 3.

Tout d'abord, le concept d'agent remplace celui d'environnement (voir point 4.2) et celui de message se substitue à la notion d'événement externe. Reste à déterminer le concept d'état du système. Il a été présenté au chapitre 3 comme un élément de représentation des états des entités ou comme accumulation locale. Pour notre part, nous lui réservons le rôle d'élément de synchronisation entre les tâches d'un même agent et accessoirement celui d'accumulation locale retenant des éléments d'informations provisoires dans la mémoire de l'agent. Il constitue de ce fait la *mémoire à court terme* (MCT) des agents comme nous le verrons dans ce qui suit.

8.2. : Ajouts aux diagrammes de transition par rapport à EPAS

Nous allons apporter ci-après plusieurs ajouts aux diagrammes de transitions tels qu'ils sont définis dans la méthode EPAS. Nous verrons que ces ajouts nous autorisent à élaborer des diagrammes de transitions contenant moins d'indéterminations et plus facilement lisibles que ceux de la méthode EPAS. Si nous avons été quelque peu influencés par le support logiciel d'implémentation, le système générique de groupes transitoires (SGGT), lors de l'élaboration de certains de ces nouveaux concepts, il n'en reste pas moins que nous nous autorisons une généralisation; nous devons en effet rester à un niveau de conceptualisation logique ne tenant compte d'aucune caractéristique physique car cela tronquerait la démarche adoptée. En effet, Jackson (JACKSON M. [1983]) écrit : "*si la modélisation est bien faite, il ne faut pas la changer lorsqu'on passe d'un support d'implémentation à un autre dû à un changement technologique ou autre. Il ne faudra remodéliser que la phase d'implémentation.*"

Avant d'aborder les concepts ajoutés, commençons par présenter un exemple. Soit le scénario "**déterminer liste joueur**" (figure 4.19) dont le but est de déterminer la liste des joueurs participant à une partie de jeu FRIK. Afin de constituer la liste des joueurs, le MJ communiquera avec les différents **joueurs** par envoi de messages. Chaque joueur lui confirmera son intention de (ne pas) jouer. Le MJ recueille alors les réponses et peut ainsi constituer la liste des joueurs qu'il transmet au **Serveur**.

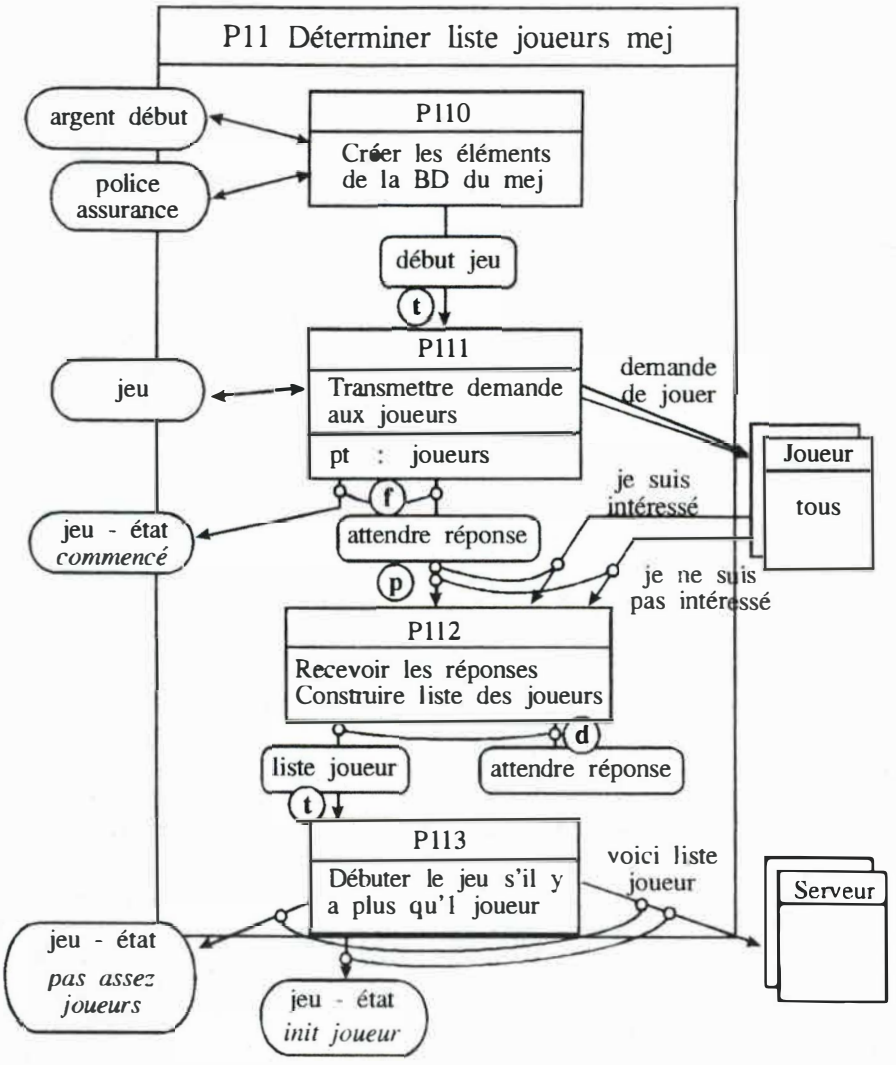

Figure 4.19 Illustration d'un diagramme de transitions

Sur base de cet exemple, nous expliquerons, tout d'abord, une façon originale d'exprimer la combinaison des événements constituant les conditions en entrée et sorties d'activités. Par la suite, nous introduirons les nouveaux concepts s'appliquant aux évolutions de la MCT de chaque agent depuis leur création jusqu'à leur destruction en passant par leurs modifications. Et enfin, nous toucherons un mot du nouveau type de tâche dégagé dans cet exemple.

8.2.1 - Conditions en activation et désactivation de tâches

Toujours dans le même esprit d'augmenter la rapidité de compréhension d'un diagramme par une amélioration de la lisibilité nous décomposerons une condition en plusieurs traits reliant les événements.

Ainsi si nous reprenons la figure 3.20, nous substituons le mode de représentation d'une disjonction, c'est-à-dire le " **OU** " de la condition c1 en activation de la tâche P1, par une décomposition. Sur le schéma (Figure 4.20), nous présenterons cette disjonction par deux arcs de cercle reliant respectivement evt1 au trait partant de la MCT1 et reliant evt1 au trait partant de la MCT2. Nous connaissons ainsi à la lecture du diagramme la condition c1. La concjonction est représentée par un arc de cercle liant des événements internes et/ou messages entre eux en entrée ou sortie d'activité. On apperçoît trois conjonctions dans le schéma ci-dessous, par exemple, une d'entre elles lie MCT3 à MCT4.

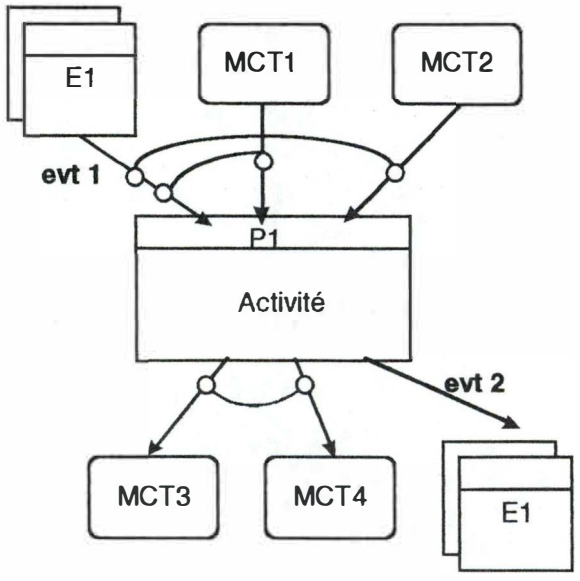

Figure 4.20 Activation / désactivation d'activités.

Cette méthode de représentation des conditions aussi bien en activation que en désactivation des activités a l'avantage de présenter immédiatement, à la simple vue du diagramme de transitions et contrairement à la méthode E.P.A.S., les différentes combinaisons d'événements et de messages. Il s'agit donc d'un moyen de rapidité de lecture accrue des diagrammes.

Ainsi dans l'exemple présenté à la figure 4.19, on apperçoit très nettement les deux conditions en entrée de l'activité **P112**. Le premier point de synchronisation combine l'événement interne provenant de la MCT **"attendre réponse"** et du message **"je suis intéressé"**. Le deuxième point de synchronisation se compose du même événement interne et du message **"je ne suis pas intéressé"**. Il existe en sortie de l'activité **P113** une double condition.

Nous avons exposé une autre manière de représenter les disjonctions en entrée comme en sortie d'activités. Comme les mémoires à court terme entrent dans certaines de ces conditions, nous allons, au point suivant, analyser leur longévité c'est-à-dire leur persistance dans le temps.

8.2.2 - Concepts relatifs à l'évolution de la mémoire des agents

La mémoire des agents et plus particulièrement la mémoire à court terme (MCT) constitue un des éléments constituant la connaissance des agents. On aurait pu leur donner une structure comme il en a été des accumulations, mais étant donné leur caractère éphémère dû à leur spécificité, il n'y a pas de raison pour qu'elles soient considérés au même titre que la mémoire à long terme constituant la connaissance longue durée des agents.

Entre leur création et leur destruction, les MCT, entrant dans la composition des conditions d'activation des tâches, sont classées en deux catégories. La première concerne les MCT qui, dans un point de synchronisation, ne participent qu'à une seule activation puis sont oubliées de l'agent (Figure 4.21, evt1), ce sont les MCT *transitoires*.

Dans la deuxième catégorie (Figure 4.21, evt2), les MCT sont impliquées dans plusieurs activations successives d'une tâche; ce pourrait être, par exemple, un compteur. Ces mémoires ont la caractéristique de subsister pendant un certain laps de temps; elles sont *permanentes* pendant cette période de temps plus ou moins longue. Elles peuvent donc à nouveau participer à la réalisation du même point de synchronisation ou être un des éléments d'un autre point de synchronisation. Nous avons décidé d'acompagner l'événement issu de cette MCT d'un '**p**' entouré d'un cercle dans le but de montrer au lecteur le caractère répétitif de cet événement déclencheur.

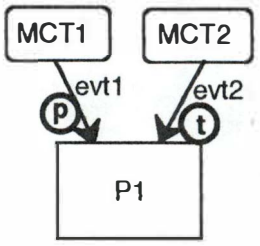

Figure 4.21 Evénément permanent/transitoire

Il est à noter que dans le deuxième cas, l'événement aurait pu être accompagné d'un **"d"** cerclé signalant la *destruction* de la MCT. Nous jugeons préférable de laisser ce symbole pour une autre fonction. En effet, nous dirons par convention que si un évènement est accompagné de ce sigle en sortie d'un processus alors le but de cet évènement est la destruction de la MCT (Figure 4.22).

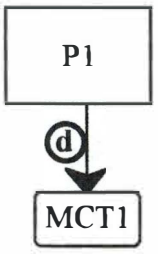

Figure 4.22 Destruction d'une MCT

En règle générale, un événement destructeur agit sur une MCT qui avait un caractère permanent dans la condition d'activation d'une activité. C'est, de plus, l'activité qui détermine le moment de la destruction de la MCT. Ainsi si l'on suppose que l'on a une MCT dont la fonction est de déclencher l'activité P1 **émettre facture** toutes les heures jusqu'à une certaine heure, l'activité P1 s'exécutera à chaque heure. Lorsqu'elle remarquera qu'il est l'heure d'arrêter (17 h), elle détruira cette MCT comme on peut le voir à la figure 4.23.

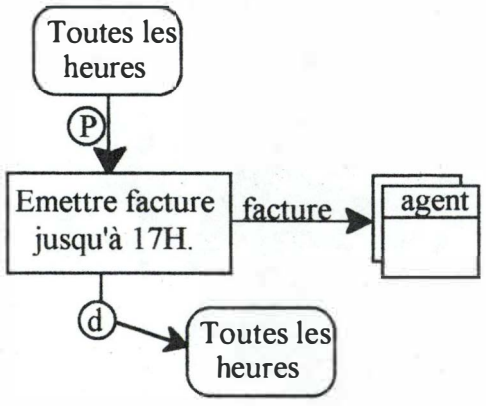

Figure 4.23 Illustration d'une destruction

Un troisième symbole peut être ajouté à la liste, mais celui-ci concerne aussi bien les MCT que les éléments de la structure de données. Il s'agit cette fois d'un changement de valeur de l'état. Nous parlons de modification et il est donc logique que l'événement en sortie d'un processus soit accompagné du sigle "**m**".

Une petite modification de l'exemple précédent permet d'illustrer ce type d'événement. La MCT n'est plus une information "toutes les heures" mais "à telle heure", c'est-à-dire qu'elle contient une heure bien précise. On ajoute une tâche supplémentaire au processus qui consiste à placer l'heure de sa prochaine activation. Il doit pour cela modifier le contenu de la MCT "à telle heure" lors de l'envoi de la facture. On obtient le schéma suivant :

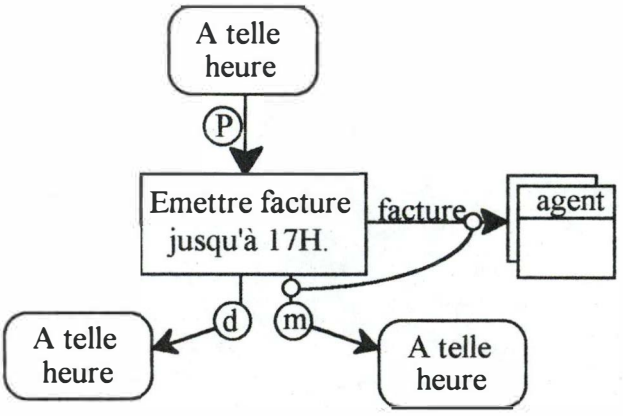

Figure 4.24 Illustration d'une modification

Les événements internes composant les conditions d'activation en entrée de processus et les effets en sortie de processus étant présentés de façon plus explicite, on peut se demander s'il n'existe pas une manière de présenter plus clairement les processus. En effet, certains d'entre-eux servent à émettre plusieurs fois un message à des agents différents, ce cas a déjà été rencontré au point 5.2. La solution à ce problème est exposée au point suivant.

8.2.3 - Evenements multiples occasionnés par une tâche.

Jamais dans la description des diagrammes EPAS du chapitre précédent nous n'avons vu apparaître la notion d'événement multiple. Il s'agit d'un mécanisme de représentation d'itérations engendré par un agent. En effet, au cours de son déroulement, une activité peut exprimer la nécessité d'émettre plusieurs messages identiques à des agents différents: par exemple un joueur envoie un message d'information à d'autres joueurs.

De cette façon, nous introduisons le concept d'*événement multiple* qui est un évènement qui sera émis à plusieurs reprises lors de l'exécution de l'activité. Ainsi le processus **P111** de la figure 4.19 est un processus à événement multiple. L'événement répété à plusieurs reprise lors du déroulement de l'activité est le message "**demande de jouer**" émis à destination des joueurs.

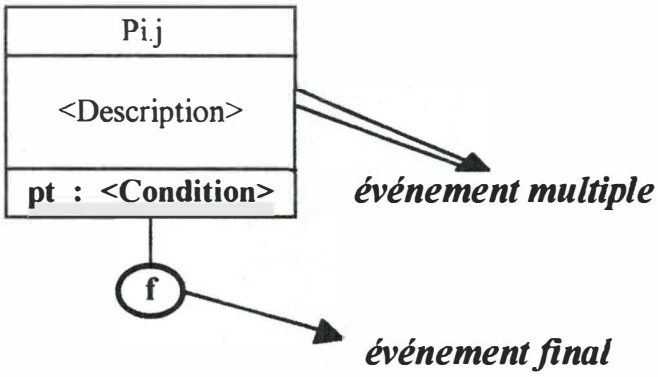

Figure 4.25 Evénement multiple

On peut remarquer dans cette représentation qu'il existe une ligne supplémentaire divisant en trois la boîte d'activité. Nous inscrivons dans la nouvelle partie ainsi créée un signe conventionnel "**PT :**" signifiant par définition "Pour Tout". Il faut par la suite indiquer la condition sur laquelle porte l'itération. Ainsi dans l'exemple précédemment cité, le message "**demande de jouer**" est envoyé aux différents joueurs mais pas à tous les joueurs, seulement ceux dont le nom figure dans la liste "**joueurs**" située dans la partie inférieure du processus. Cette liste constitue de ce fait la condition d'émission du message.

Encore faut-il spécifier la forme qu'auront les évènements émis par l'activité à événement multiple. Tout d'abord, il y a l'évènement émis par le corps de la boucle à chaque itération, l'événement multiple, puis l'événement déclenché en sortie de boucle, l'événement déclenché par la fin de l'itération, l'événement final.

Dans les deux cas, nous représentons les évènements par une flèche. Afin de distinguer l'événement *multiple* de l'événement *final*, nous traçons, respectivement, une flèche à double traits et une flèche simple accompagné du symbole "**f**" pour événement *final* .

Dans la figure 4.19, lorsque l'activité **P111** s'est exécutée, elle se termine par la mise en oeuvre de l'événement final qui est la conjugaison d'une modification de la mémoire "**jeu**" et la création de la mémoire "**attendre réponse**".

On a présenté plusieurs ajouts et modifications rendant la lecture plus aisée des diagrammes de transitions. Nous y avons, entre autres, exprimé la mémoire à court terme. Le point suivant propose un méthode de représentation des connaissances d'un agent, c'est-à-dire de sa mémoire permanente.

8.2.4 - Représentation de la connaissance des agents

Dans le chapitre 3, nous avons présenté les états du système comme des éléments de la représentation à la fois de la structure de données et des éléments de la mémoire du système lui permettant de retenir des éléments d'information et de faire la synchronisation des processus dans le temps, c'est-à-dire leur ordonnancement.

Au point 8.1 de cette partie, nous avons introduit la notion de (MCT) dans le but de distinguer parmi les états du système la mémoire à court terme de la mémoire à long terme que nous appellerons mémoire par la suite. Il reste donc à exposer le mode de représentation de sa mémoire à long terme formant la base de connaissance du raisonnement de l'agent. Pour augmenter la lisibilité et gagner en compréhension, nous introduisons une distinction entre ces deux formes de mémoire qui ont une fonction différente. La première sert essentiellement d'élément de synchronisation alors que la deuxième permet à l'agent d'emmagasiner des informations qui lui serviront dans d'autres raisonnements.

Ainsi, lorsque le meneur de jeu construit la liste des joueurs désireux de prendre part à une partie de jeu FRIK, diagramme de transition "P11 déterminer liste joueurs mej" (Figure 4.19), on peut constater une différence dans la représentation de ces deux mémoires. Nous plaçons dans la catégorie court terme: **début jeu**, **attendre réponse** et **liste joueur**, et dans la catégorie long terme : **argent début** , **jeu** et **Police assurance**.

Nous avons choisi de représenter la mémoire de l'agent par un rectangle dont les deux extrémités sont des demis-cercle. Cela permet une légère distinction avec la MCT dont seuls les coins sont arrondis. La mémoire est liée aux processus soit par un arc bidirectionnel, soit par un arc unidirectionnel. La signification du premier cas, flèche bidirectionnelle, est très simple; cela signifie que l'activité consulte la mémoire de l'agent afin d'en extraire les éléments d'information nécessaire pour poursuivre son exécution ou qu'elle y enregistre des éléments d'information dont elle aura besoin ultérieurement.

Dans le cas d'un arc unidirectionnel, nous distingons deux sous-cas. Soit l'origine de l'arc est la mémoire et la destination un processus, soit l'inverse. Cette distinction est importante car, dans le premier sous-cas, l'arc est un événement qui tire son origine de la valeur de l'état de la mémoire et entre dans la constitution d'un point de synchronisation. Le second cas correspond à une transition d'états (cfr. cycles de vie); en effet, l'arc correspond à un événement en sortie d'activité dont le but est la transition vers un autre état, spécifié dans le cycle de vie de la mémoire de l'agent (section précédente).

Dès lors, on se rend compte de l'importance de l'étape de constitution des cycles de vie, car l'évolution des connaissances de l'agent, décrite précédemment, joue sur l'activation des activités.

Figure 4.26 Structure d'une mémoire à long terme

Règle de nomenclature

La mémoire à long terme est identifiée par un nom attribué lors de l'étape "structure conceptuelle de données". Dans le cas de l'évolution de la mémoire de l'agent, on ajoute le nom de l'état ainsi que sa valeur que l'on inscrit en italique. Cela permet de faciliter la lecture des diagrammes.

Un bel exemple nous est fourni dans les diagrammes "P11 D**éteminer liste joueurs mej**" (Figure 4.19) et "**P3 clôture mej**" (Figure 4.27). Grâce à ces deux diagrammes, on peut connaître le procédé de raisonnement de l'agent meneur de jeu. En effet, supposons qu'il n'y ait aucun joueur désireux de jouer, le meneur de jeu ne peut pas constituer la liste des joueurs (P112) et il place (P113) dans sa mémoire l'information "*pas assez joueur*". Cet état de mémoire déclenche automatiquement l'activation de l'activité (P31) afin de clôturer la partie de jeu en informant le demandeur qu'il est le seul à vouloir jouer (P32).

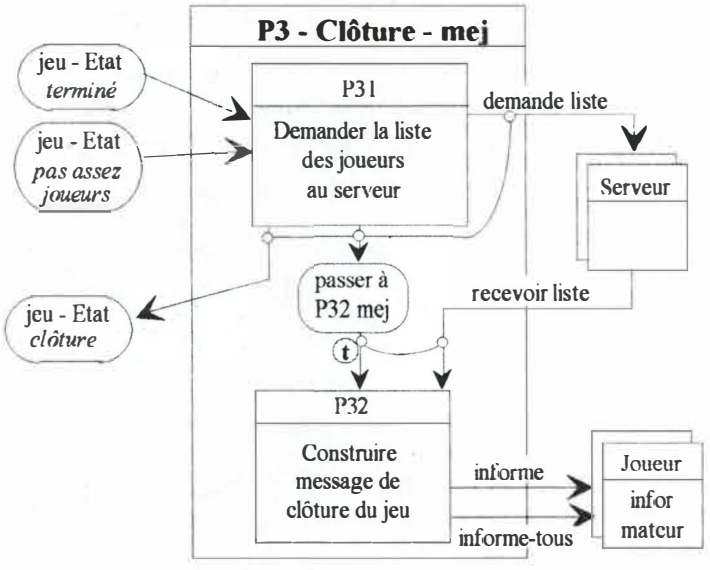

Figure 4.27 Diagramme de transition: clôture du jeu

Nous nous trouvons au terme de la modélisation des comportements de chaque agent. Tous les concepts ont été exposés, il est dès lors possible d'analyser les diagrammes de transition et de les comprendre. Nous invitons le lecteur à prendre connaissance de la deuxième partie de ce travail exposant l'ensemble de la résolution du jeu FRIK selon la méthode que l'on vient de présenter et qui se trouve placée dans les annexes 2 (second volume).

Néamoins, il subsiste un petit problème. En effet, on a développé pour chaque agent un ensemble de diagrammes selon l'approche par raffinements successifs propre à EPAS que nous avons ponctuée par des "feed-back" sur certaines étapes. Cette possibilité de modification des spécifications faites auparavant fut inspirée par les méthodes orientées objets. L'inconvénient d'une telle approche résulte dans un amalgame de schémas des agents répartis selon la découpe des comportements par raffinements successifs et pas toujours selon les comportements propres à chaque agent. C'est pourquoi la section suivante propose de rétablir l'ordre au sein de chaque agent en proposant une méthode présentant chaque agent ainsi que ses comportements et interactions.

Section 9 - Regroupement par agent

Nous voici de nouveau confronté à la contrainte de modélisation orientée agent. Si la méthode EPAS nous offre la possibilité de schématiser les comportements, de planifier les relations et dialogues de plusieurs agents interagissant dans un ou plusieurs systèmes, elle nous laisse sans réponse quant à l'organisation des diagrammes de transitions entre eux, comme cela a été présenté à la fin de la section précédente. En effet, à l'ultime étape de la phase d'analyse, nous obtenons un amalgame de diagrammes de transitions qu'il convient de réordonner au sein de chaque agent. Comment représenter chaque agent? Voilà la question qu'il nous reste à élucider dans cette démarche. Quels sont les modèles existant à l'heure actuelle nous autorisant de constituer un squelette que nous reproduirons pour chaque agent ?

Pour résumer la situation, chaque agent développe un ou plusieurs comportement(s). Afin de les mener à leur terme, il emmagasine des informations dont il peut se servir à tout moment. Il dispose à cette fin d'une mémoire formant ses connaissances. Au cours des multiples activités composant un comportement, un agent émet des messages vers d'autres agents. Grâce à cette communication, il lui est possible de résoudre des problèmes auxquels il est confronté et pour lesquels il n'est pas armé, c'est-à-dire des problèmes pour lesquels il ne possède aucun élément d'information en mémoire.

Encore une fois c'est la bibliographie traitant d'approche orientées objets qui procure une solution. En effet, en recoupant les différentes méthodes de présentation d'un objet proposées par différents auteurs, nous obtenons un structure générale de la forme : objet, variables, méthodes et parfois les interactions de l'objet avec les autres objets.

Nous décidons de dériver le squelette de présentation des agents de cette structure générale. En effet, chaque agent est constitué d'un nom, d'une ou plusieurs entités constituant sa mémoire, de divers comportements identifiés tout au long de la modélisation et enfin des interactions avec d'autres agents. Nous optons pour le squelette présenté ci-dessous que nous nommons **fiche signalétique de l'agent** en raison de son caractère à renseigner le lecteur sur les caractéristiques de l'agent.

Agent : <nom de l'agent>

Mémoire :

<nom de la ou des entité(s) constituant la mémoire de l'agent>

Comportements :

<liste des diagrammes de transitions dégagés pour chaque agent>

Interactions :

<liste des agents avec lesquels il y a échange(s) de messages>

Voyons à présent ce que cela pourrait donner dans le cadre du jeu FRIK. Prenons par exemple les fiches signalétiques des agents joueur, case hasard et utilisateur; les fiches signalétique des autres agents se trouvent en annexe 2.

Agent : Joueur

Mémoire : joueur

Comportements :

Initialisation

P11 det liste jou joueur

P12 init chaque jou jou

Faire un tour

Module CONSULTATION

P21 consultation bd jou

Module COEUR

P21_Faire un tour_jou

Module MODIFICATEUR

P21_Modificateur_jou

Module INFORMATEUR

P21_Informateur_jou

Module FONCTIONS COMMUNES

P21_Fonctions_communes_jou

Interactions :

Utilisateur

Case prêt, hasard, départ, bourse, propriété, art, or

mej

Serveur

Agent : CASE Hasard

Mémoire : carte hasard

Comportements :

Initialisation

P13 init case hasard

Faire tour

P21 faire tour hasard

P212 cata pa has

P212 pas rej has

P212paiement has

Interactions :

Joueur

Serveur

mej

Agent : Utilisateur

Mémoire : néant

Comportements :

Module INFORMATEUR

P1 Utilisateur

Module FONCTIONS

P21_Informateur_jou

P2 fonctions utilisateur

Interactions :

Joueur

Grâce à cette manière de représenter les agents, nous rendons la lecture des spécifications plus compréhensible car le lecteur peut voir pour chaque agent ses divers comportements et interactions. Nous pouvons aussi affirmer que cela permet une meilleure maintenance car si l'on veut modifier un des comportements de l'agent, on se référera à cette fiche signalétique et l'on connaîtra immédiatement les divers autres agents dans lesquels il pourrait y avoir du changement.

Section 10 - Conclusions

Le but recherché avant d'aborder ce chapitre était de trouver une méthode de modélisation de la solution à un problème distribué selon une approche multi-agents comme cela a été expliqué dans les deux premiers chapitres de ce travail.

Le travail réalisé par de Looz/Jourquin et la bibliographie traitant des méthodes orientées objets, nous ont permis d'aborder les deux premières sections de ce chapitre au terme desquelles une partie des agents du problème a été identifiée. Dans un deuxième temps, il fallait modéliser leurs comportements. La méthode mise au point par Moulin semblait prometteuse pour la modélisation de systèmes distribués. La troisième section de ce travail nous a permis de créer un lien entre la modélisation orientée objet et la modélisation de systèmes. En effet, elle présentait sous forme de plans généraux les grands comportements qui se dégageaient de la spécification, c'est-à-dire les comportements de base du système-cible (le jeu FRIK) englobant les rôles des agents. A partir de ce point, la modélisation des tâches des agents a été possible grâce à la méthode EPAS dont on a adapté le vocabulaire à celui présenté dans la taxonomie des sytèmes distribués multi-agents.

Vu l'évolution des systèmes informatiques actuels, nous sommes en droit de penser que les méthodes de modélisation de systèmes multi-agents possèdent un bel avenir. En effet, avec l'avènement des systèmes à objets d'où découlent les agents présentés dans ce travail et l'expansion voire l'explosion des réseaux, nous verrons de plus en plus apparaître de logiciels distribuant leur connaissance sur plusieurs machines dans lesquelles un ou plusieurs agents se la partageront. Cela aura pour effet d'aboutir à un système multi-agents où chaque agent aura sa propre connaissance et la fera évoluer. On s'appercoit que pour modéliser de façon optimale un tel système il convient de disposer d'une méthode rigoureuse permettant de gérer les comportements et interactions de chaque agent afin de traiter et faire évoluer sa connaissance.

Il reste à montrer comment on doit procéder pour implémenter un système multi-agents sur un réseau distribué et surtout comment passer des diagrammes de transition à un programme codé dans un langage de programmation. Le chapitre suivant répond à cette question en présentant un outil à deux faces. Tout d'abord il sert d'outil d'aide à la programmation en aidant le concepteur à introduire les diagrammes de transitions obtenus au terme de la modélisation mais, dans un second temps, il permet de gérer la communication entre les agents. Nous verrons que cet outil est un support adéquat pour représenter les diagrammes de transitions modifiés tels que nous les avons présentés.

CHAPITRE 5 : Le SGGT, un outil d'aide à la programmation

Section 1 - Introduction

Lorsqu'on se trouve en possession de diagrammes de transitions obtenus à l'aide de notre méthode, il ne reste plus qu'à concevoir un programme exécutable. Or, à partir de ceux-ci, toute personne capable de programmer peut en sortir des lignes de code sans trop de problème, voire même dans le langage de son choix. Grâce au SGGT (Système Générique de Groupe Transitoire), ce passage à la programmation peut se faire d'une manière beaucoup plus aisée. En effet, après un simple travail d'encodage, il ne reste plus qu'à programmer le contenu de chaque processus, ce qui revient à quelques lignes de codes par processus. Ceci peut être réalisé dans le langage de son choix, voire même par plusieurs personnes différentes, entraînant du même coup un gain de temps certain dans l'élaboration du programme. Cet outil de programmation va être détaillé dans les lignes qui suivent, pour faire ressortir l'intégration des diagrammes de transition dans le SGGT.

Section 2 - Le système générique de groupes transitoires

Le chapitre 4 a présenté une méthode de modélisation de problèmes distribués selon une approche multi-agents. Les agents voient en général leurs comportements et leurs structures d'informations se raffiner au cours des différentes étapes de la modélisation. Cette démarche aboutit donc aux diagrammes de transitions, c'est à dire à un ensemble de tâches schématisant les différents comportements, et dont le séquencement est ordonné par les différents états des agents. Ces états dépendent de la mémoire à court terme de chaque agent et/ou de l'arrivée de messages en provenance d'autres agents.

Un outil permet de gérer ces tâches c'est à dire qu'il activera l'une d'entre elles, lorsque tous les éléments faisant partie de sa précondition d'activation seront réunis, ou qu'il créera tous les éléments spécifiés en sortie de cette activité. Cet outil est le Système Générique de Groupes Transitoires (SGGT) mis au point par Cloutier (CLOUTIER L. [1991]).

Dans les points suivants, nous allons définir le SGGT, donner ses caractéristiques, montrer les éléments qu'il traite et la manière dont il faut définir les tâches de sorte qu'il puisse les gérer convenablement. Le tout sera illustré au moyen d'exemples extraits des diagrammes de transitions du jeu FRIK.

2.1 - Présentation

Comme son nom l'indique, le SGGT est un outil d'aide à la création et à la gestion des groupes transitoires. Un groupe transitoire est un élément composé de préconditions d'activation, constituées des éléments en entrées de tâches, d'une transition, nom donné aux processus par Cloutier (CLOUTIER L. [1991]), et d'effets, composés des éléments entrant dans les conditions en sortie de tâche. En effet, lorsque les préconditions d'activation, les effets et les transitions ont été définis, le SGGT veille au déclenchement des transitions pour lesquelles la précondition d'activation est vérifiée. De plus, il assume la coopération entre agents car il assure la réception et l'émission des messages entre les agents.

Le SGGT présente un avantage supplémentaire car c'est un outil d'aide à la programmation. En effet, à partir de la définition des tâches schématisant le comportement d'un agent, il génère le squelette du programme final contenant les processus définis par le concepteur. Il s'agit d'un squelette car le SGGT ne gère que les activations des processus, c'est à dire qu'il analyse les préconditions d'activation et veille à la création des effets en sortie de tâche mais ne s'occupe pas de fournir le code pour le corps des processus.

2.2 - Principes de fonctionnement

Ainsi que nous l'avons vu dans la présentation de cet outil, le SGGT est un outil d'aide à la conception des programmes. En effet, à partir de la définition des éléments constituant un groupe transitoire, il écrira le squelette d'un programme. Le concepteur ne devant plus que remplir le corps de chaque processus de ce programme. L'objectif des points suivants est de présenter les trois composants d'un groupe transitoire et de montrer la manière de les formuler.

Avant d'aborder les différentes définitions, nous présentons un exemple. Celui-ci est tiré des spécifications du jeu FRIK. Cela pour deux raisons, tout d'abord il agrémentera chaque définition de concept, mais il nous permettra également de montrer au lecteur la motivation qui nous a poussé à porter des modifications et ajouts aux diagrammes de transitions tels qu'ils existaient dans la méthode EPAS. L'exemple (Figure 5.1) est celui présenter au chapitre précédent (point 8.2), il concerne la création de la liste des joueurs participant à une partie de jeu.

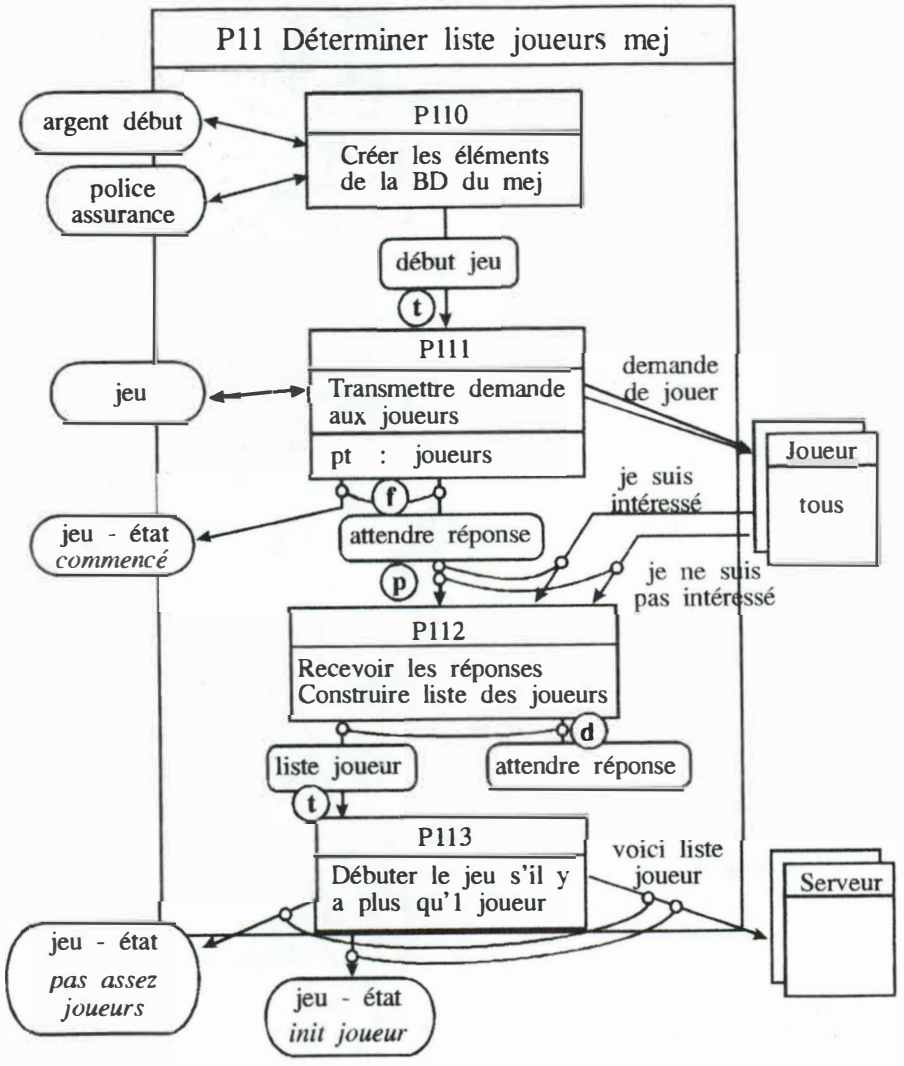

Figure 5.1 Diagramme de transition Déterminer liste joueurs

2.2.1 - Définition des groupes transitoires

Afin que le SGGT puisse constituer le squelette du programme, le concepteur doit définir l'ensemble des groupes transitoires apparaissant dans les diagrammes de transitions; un groupe transitoire est en effet constitué de l'ensemble des éléments rattachés à une tâche. Ces composantes sont les préconditions d'activation, la transition et les effets comme le montre la figure 5.2.

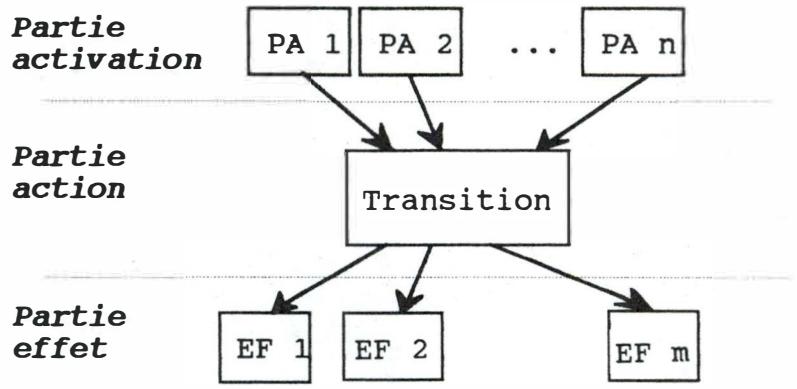

Figure 5.2 Schéma d'un groupe transitoire

On voit distinctement sur le schéma ci-dessus les 3 parties constituants un groupe transitoire. La partie activation composée d'une disjonction de préconditions d'activation (PA) où chaque PA est constituée d'une conjonction d'éléments déclencheurs dont le nombre varie entre 1 et n, et où n est un nombre entier quelconque. Un élément déclencheur est un élément créé par une transition et dont la survenance a pour but de remplir une condition d'activation d'une transition. Ainsi dans le schéma 5.1, si on prend, par exemple, la tâche P112, on aperçoit très distinctement deux PA constituées de deux éléments déclencheurs chacune. La première est composée des éléments déclencheurs : la MCT **"attendre réponse"** et le message **"je suis intéressé"** et la deuxième comprend la même MCT mais le message est **"je ne suis pas intéressé"**.

La deuxième partie est constituée de la transaction, c'est à dire de la tâche qui sera exécutée lorsque tous les éléments déclencheurs à une PA auront été réunis. Pour reprendre le même exemple, la tâche **"P112"** consiste à recevoir les réponses de tous les joueurs potentiels et, de là, à construire la liste des joueurs effectifs.

La dernière partie est la partie effet qui est composée d'une disjonction d'effets. De cet ensemble d'effets, seulement un seul sera déclenché par transition. De plus, il est composé d'une conjonction d'éléments déclenchés dont le but est d'entrer dans une précondition d'activation dans laquelle ils seront éléments déclencheurs. Toujours dans le même exemple avec la même tâche, on aperçoit un seul effet composé de deux

éléments déclenchés : **"liste joueur"** et **"attendre réponse"**.

2.3 - Eléments déclencheurs, déclenchés et transitions supportés par le SGGT.

2.3.1 - Eléments déclencheurs.

Les éléments déclencheurs sont les éléments dont la combinaison permet de constituer les préconditions d'activation d'une transition. Le SGGT gère trois types d'éléments déclencheurs qui sont : consultation d'un état du système (ES), consultation d'un état de la base de données (BD), arrivée de messages.

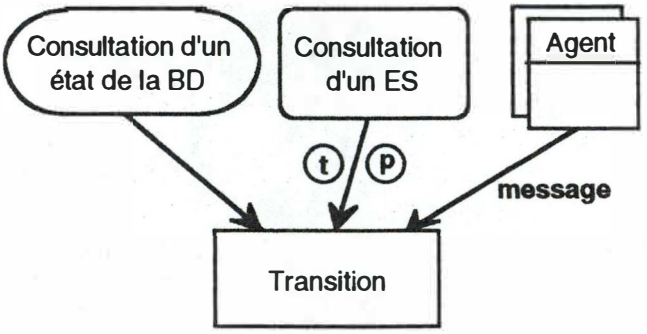

Figure 5.3 Eléments déclencheurs

I Consultation d'un état du système

Les ES décrits ici correspondent au concept de mémoire à court terme d'un agent présenté au chapitre précédent, par la suite nous utiliserons indifféremment MCT et ES. Si cette MCT est créé au terme d'une autre transition, elle entre dans la PA d'une transition et est dès lors accessible du SGGT pour une consultation, c'est à dire pour prendre connaissance" des éléments d'information enregistrés.

On notera les deux signes conventionnels de part et d'autre de l'arc indiquant si la MCT est persistante ou transitoire, auquel cas le SGGT l'efface de la mémoire après consultation; ces concepts ont été définis au chapitre 4, point 7.2.1.

On peut en voir un exemple dans le diagramme de transition "demander liste joueurs". En effet, la mémoire à court terme **"liste joueur"** est la PA de la tâche P113, cela signifie que cette tâche prend connaissance des informations contenues dans cette MCT. Comme ce sont les noms des joueurs qui y sont enregistrés, elle en aura connaissance.

II Consultation d'un état de la base de donnée

Dans le cas d'une consultation de la BD, le SGGT examinera en permanence l'attribut d'état d'une entité. D'après sa valeur à un moment donné, le SGGT le placera dans la précondition d'activation de telle ou telle PA comme élément déclencheur vérifié.

Pour en avoir un exemple, nous reprenons le diagramme de transition illustrant la clôture du jeu présenté à la section 7 du chapitre précédent. En effet, si on prend la tâche P31, on remarque qu'elle est activée de deux façons qui dépendent toutes deux de l'état de l'entité **"jeu"**. Soit que le jeu se termine de façon normale, dans ce cas l'état est à "*terminé*", soit que le jeu se termine car il n'y a pas assez de joueurs, et dans ce cas, la valeur de l'état est "*pas assez joueurs*".

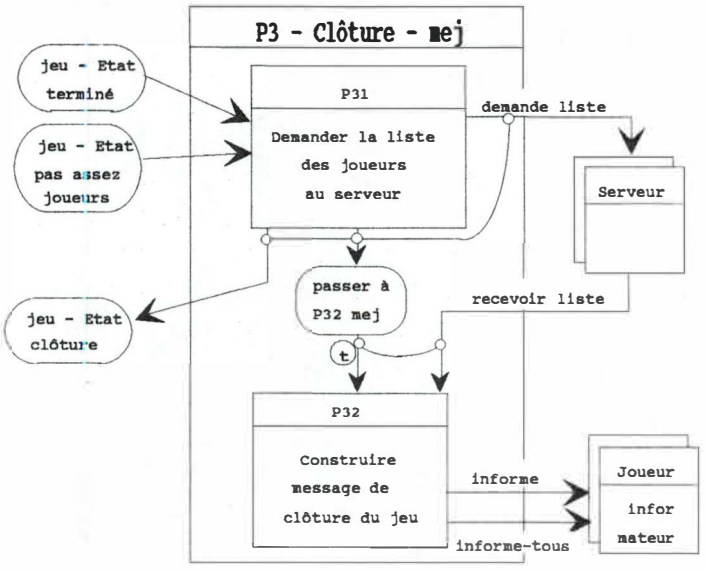

Figure 5.4 Diagramme de transition: clôture du jeu

III Arrivée d'un message

Lorsqu'un message en provenance d'un environnement extérieur survient, il est aussitôt placé dans la précondition d'activation dans laquelle il avait été spécifié comme élément déclencheur. C'est le cas avec les deux messages **"je suis intéressé"** et **"je ne suis pas intéressé"**.

2.3.2 - Eléments déclenchés

Les éléments déclenchés sont les éléments dont la combinaison permet de constituer les effets d'une transition. Le SGGT est conçu pour gérer cinq types d'événements dont trois concernent les états du système (ES) : modification d'un état de la BD, création, modification et destruction d'un ES et, enfin, émission d'un message.

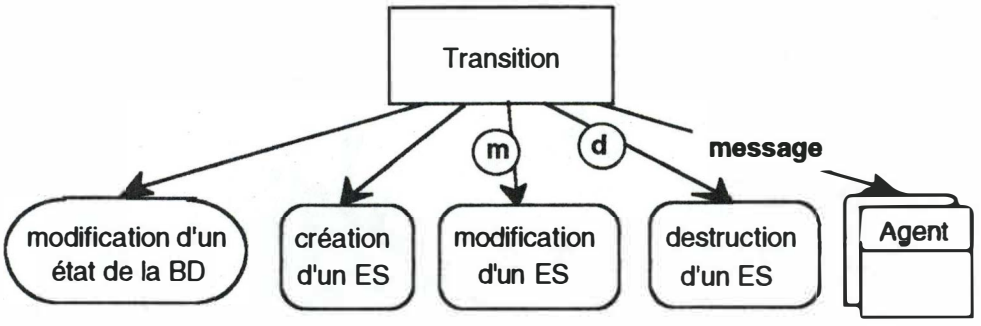

Figure 5.5 Eléments déclenchés

I Modification d'un état de la BD

Lorsque cet élément déclenché est activé, il a pour conséquence la modification de l'état de la BD spécifiée. C'est grâce à cet événement que les transitions d'un état à un autre peuvent se dérouler comme spécifié dans les cycles de vie.

On en a plusieurs exemples dans le diagramme de transition de la figure 5.1. Si on prend la dernière tâche "P113", on remarque que dans chaque effet, ce type d'événement survient. Dans le premier effet, l'état devient "***init joueur***" afin de débuter l'initialisation des joueurs et donc d'une partie de jeu. Dans le second cas, l'état devient "***pas assez joueurs***" qui a pour but d'activer le processus "P31" de clôture du jeu par manque de joueurs.

II Création d'un ES

La conséquence au déclenchement de cet événement est la création d'un état du système, c'est à dire la création d'une MCT dont le rôle est soit de stocker des éléments d'informations utiles à une autre transition, soit de veiller au séquencement des groupes transitoires.

On peut remarquer que c'est le cas avec, par exemple, la MCT "**début jeu**" qui est créée par la tâche "**P110**" afin de déclencher la tâche "**P111**".

III Destruction d'un ES

Le but de cet élément déclenché est, comme son nom l'indique, la destruction d'un ES créé par un autre groupe transitoire. En effet, lorsqu'une MCT possède le statut permanent, cela signifie qu'il sera consulté à plusieurs reprise par une transition et donc lorsqu'il ne lui sera plus nécessaire, elle le détruira.

On trouve un cas de destruction d'une MCT dans le diagramme de transition qui consiste à déterminer la liste des joueurs (Figure 5.1). La tâche **P112** détruit la MCT **"attendre réponse"** lorsqu'elle à reçu toutes les réponses. On remarque que cette MCT possède le statut permanent car tant que toutes les réponses n'auront pas été rendues, la tâche P112 attendra l'arrivée de messages.

IV Modification d'un ES

Le but de cet élément est de changer la ou les valeur(s) contenue(s) dans une MCT créé par un autre groupe transitoire.

2.3.3 - Transitions

Les transitions dont on parle ici correspondent aux tâches ou processus présentés aux chapitres 3 et 4. Le SGGT supporte deux types de transitions : les transitions simples et les transitions composées à événements multiples. Nous ne définirons pas ici ces deux concepts car ils ont déjà été présentés au chapitre 4.

Les divers éléments supportés par le SGGT viennent d'être présentés. Pour pouvoir être introduit et donc compris par le SGGT, il faut se donner un formalisme de représentation; c'est précisément l'objet du point suivant.

2.4 - Définition des groupes transitoires

Comme on l'a vu précédemment, les groupes transitoires sont formés de trois parties: la partie des préconditions d'activation, la partie reprenant la transition et la partie des effets. Les PA et effets sont des ensembles composés d'un ou plusieurs éléments déclencheurs et, respectivement, déclenchés.

L'introduction des éléments formant un GT commence par la définition des éléments simples, c'est-à-dire les éléments déclencheurs et déclenchés, car ils ne sont pas la combinaison d'autres éléments. A partir de ces éléments simples, on définira les éléments composés que sont les PA et les effets. Enfin, suivra la définition du groupe transitoire. Celle-ci est une disjonction de PA et d'effets. Nous commencerons par décrire les éléments déclencheurs puis les éléments déclenchés et nous terminerons

par les transitions.

2.4.1 - Eléments déclencheurs

I Les messages

Un message est défini au moyen de cinq paramètres: le type, la force, le destinataire, l'envoyeur et le nom du message.

Les deux premiers paramètres ont été définis selon la théorie des actes du discours. Ainsi le **type** peut être de différentes natures : directif comme une *demande* ou une *commande* ou encore assertif. Dans ce second cas, le message est de type *informe*. Le deuxième paramètre, la **force**, concerne l'importance du message. Cette force est cotée de 1 (impératif) à 5 (ne presse pas). Le paramètre **destinataire** est destiné à recevoir le nom du récepteur du message. L'**envoyeur,** quant à lui reçoit le nom de l'émetteur du message et enfin le **nom du message** dans lequel est placé le message ainsi que la liste des variables qui seront transmises avec lui.

Afin d'illustrer la description d'un message, on reprend l'exemple utilisé depuis le début de cette section (Fig. 5.1). Le message servant à illustrer la théorie est celui issu de la tâche **"P111"** dont le nom est **"demande de jouer"**.

La description du message est :

Type : informe
Force : 1
Destinataire : Joueur
Envoyeur : mej
Message : demande de joueur()

On remarque que le paramètre du destinataire "Joueur" a été défini avec une majuscule, cela afin de déclarer au SGGT qu'il s'agit d'une variable qui prendra des valeurs différentes, celle des joueurs potentiels, puisque le message doit leur être envoyé.

Les messages jouent évidemment un rôle très important lorsqu'on sait que c'est le moyen choisi pour la communication entre les agents. Jusqu'il y a peu, le moyen utilisé en intelligence artificielle était la méthode de blackboard, mais depuis qu'on essaie d'intégrer les croyances et les intentions des agents, l'utilisation des messages devient évidemment plus appropriée. On se rapproche en fait de plus en plus du langage naturel, et en intelligence artificielle, ce n'est évidemment pas à négliger.

II Consultation d'un état de la BD

La consultation d'un état de la BD se définit avec le nom de l'entité concernée, les attributs précédent l'état, l'état proprement dit et les attributs suivant l'état tel que le

montre la structure suivante :

état <Nom de l'entité> **(** <liste des attributs>, <état>, <liste des attributs>**)**
suivant l'état suivant l'état

Appliquons ceci à un exemple. L'activité **P31** du diagramme de transitions **"P3 clôture mej"** est activée après consultation de l'état de l'entité jeu. Deux valeurs sont possibles; il y a donc deux déclarations :

état : jeu (Paramètres, *pas assez joueurs*)

état : jeu (Paramètres, *terminé*)

III Consultation d'un ES

On définira une consultation à partir du nom de l'état ainsi que les variables destinées à recevoir les éléments d'information à transmettre à une autre transition et son caractère transitoire ou permanent afin que le SGGT sache s'il doit détruire ou non cette MCT.

La tâche P113 (Fig. 5.1) accomplit une consultation d'une MCT qui à deux rôles. Le premier consiste à déclencher la tâche P113 et le deuxième à retenir la liste "*Liste*" des joueurs effectifs. Liste qui sera prise en considération par la tâche P113 grâce à une consultation puis sera transmise au serveur via le message "voici liste joueur".

2.4.2 - Eléments déclenchés

I Message

Les messages en tant qu'éléments déclenchés se définissent de manière identique aux messages en tant qu'éléments déclencheurs.

Afin d'illustrer la description d'un message, on reprend l'exemple utilisé depuis le début de cette section (Fig. 5.1). Le message servant à illustrer la théorie est celui issus de la tâche **"P113"** dont le nom est **"voici liste joueur"**.

La description du message est :

Type : informe
Force : Force
destinataire : serveur
Envoyeur : mej
Message : voici liste joueur (Liste)

Il y a deux points à signaler. Tout d'abord le paramètre Force, il n'a pas été défini par un chiffre car nous voulons signaler le fait que l'importance de cet acte du discours ne nous intéresse pas. Elle peut donc être n'importe quoi car pour le SGGT une variable commence toujours par une majuscule. Ensuite, dans la déclaration du nom du message, il apparaît une autre variable "*Liste*", c'est un paramètre à envoyer au serveur et qui contient la liste des joueurs effectifs.

II Création et destruction d'un ES

Dans ces deux cas, il n'y a qu'un seul paramètre à déterminer qui est le nom de l'état. On peut y adjoindre des variables si lors de la création, elles ont été déclarées.

Nous commençons par déclarer la création **"liste joueur"** puis la destruction de **"attendre réponse"**, deux MCT du diagramme de transition "déterminer liste joueurs".

Création d'une MCT :

Type : création d'une MCT

Etat : liste joueur(Liste)

Destruction d'une MCT :

Type : destruction d'une MCT

Etat : attendre réponse

III Modification d'un ES

Pour une modification d'un ES, il est nécessaire de donner deux paramètres: l'ancien état, c'est à dire l'état dans lequel se trouve l'ES avant la transition et le nouvel état qui est celui dans lequel se trouve l'ES après la transition.

IV Modification d'un état dans la BD

Il est important de spécifier sur quelle entité de la BD on travaille avant de donner la valeur initiale de l'état. La dernière étape consiste, alors, à déterminer quelle sera la valeur finale de l'entité.

Nous illustrons ce point ci par la modification de l'entité jeu en sortie de la tâche P111. Voici la façon de définir une modification :

Type : modification de la BD
Etat actuel : jeu(Paramètres, EtatJeu)
Nouvel état : jeu(Paramètres, commence)

On peut remarquer que lors de la déclaration, l'état passe de "EtatJeu", qui est une variable, à "commencé", qui est écrit avec une minuscule car il s'agit d'une valeur définie; en d'autres mots, cela signifie que peu importe la valeur initiale, la valeur finale sera "commencé".

2.4.3 - Préconditions d'activation et effets

Pour définir les PA et les effets, il suffit de spécifier au SGGT quels sont les éléments déclencheurs et déclenchés qui en font partie et en donner le nom.

Squelette d'une PA :

Précondition d'activation : <nom de la PA>
Elément déclencheur 1 : <nom>
Déclaration de l'élément déclencheur tel qu'on l'a vu précédemment
Elément déclencheur n : <nom>
Déclaration de l'élément déclencheur.

Squelette d'un effet :

Effet : <nom de l'effet>
Elément déclenché 1 : <nom>
Déclaration de l'élément déclenché tel qu'on l'a vu précédemment
Elément déclenché m : <nom>
Déclaration de l'élément déclenché.

La déclaration de la PA de la tâche P112. Il s'agit ici d'un cas intéressant car il y a deux préconditions d'activation. Nous dirons que pour en définir le nom, nous inscrivons les lettres **"PA"** suivi du nom de la tâche **"P112"**. Dans ce cas, comme il y a deux PA, nous leur adjoindrons un numéro afin de les distinguer.

Précondition d'activation : PA1_P112_intéressé
Elément déclencheur 1 : MCT_attendre réponse
Type : consultation d'une MCT
Etat : attendre réponse
Type : persistant
Elément déclencheur 2 : msg_intéressé
Type : événement message
acte : informe
force : Force
destinataire : mej
envoyeur : Joueur
message : je suis intéressé()

Précondition d'activation : PA2_P112_pas_intéressé
Elément déclencheur 1 : MCT_attendre réponse
Type : consultation d'une MCT

Etat : attendre réponse
Type : persistant
Elément déclencheur 2 : msg_pas_intéressé
Type : événement message
acte : informe
force : Force
destinataire : mej
envoyeur : Joueur
message : je ne suis pas intéressé()

Afin d'éviter toutes confusions entre les effets, leur nom sera déclaré de manière analogue au nom des PA. Ainsi nous obtenons la déclaration suivante pour la tâche P112:

Effet : ef_P112
Elément déclenché 1 : créé_liste_joueur
Type : création d'une MCT
Etat : liste_joueur(Liste)
Elément déclenché 2 : détruire_attendre_réponse
Type : destruction d'une MCT
Etat : attendre réponse

2.4.4 - Définition d'un GT

Il faut un nom identifiant puis donner le type de la transition, c'est à dire spécifier s'il s'agit d'une transition à effet "simple"; dans ce cas, on note le mot "simple". S'il s'agit d'une transition à effet multiple, on inscrit le mot "multiple". Ensuite, on définit la liste des PA et la liste des effets. Un groupe transitoire est déclaré de la façon suivante :

Groupe transitoire : <nom>
Précondition d'activation : <nom>
Effet : <nom>

Nous disposons à présent de tous les éléments nécessaires à la déclaration des groupes transitoires constituant les diagrammes de transitions. Nous pouvons dès lors aborder la déclaration complète de la tâche P112.

Groupe transitoire : P112
Précondition d'activation : PA1_P112_intéressé
Elément déclencheur 1 : MCT_attendre réponse
Type : consultation d'une MCT
Etat : attendre réponse
Type : persistant

Elément déclencheur 2 : msg_intéressé
Type : événement message
acte : informe
force : Force
destinataire : mej
envoyeur : Joueur
message : je suis intéressé()

Précondition d'activation : PA2_P112_pas_intéressé
Elément déclencheur 1 : MCT_attendre réponse
Type : consultation d'une MCT
Etat : attendre réponse
Type : persistant

Elément déclencheur 2 : msg_pas_intéressé
Type : événement message
acte : informe
force : Force
destinataire : mej
envoyeur : Joueur
message : je ne suis pas intéressé()

Effet : ef_P112
Elément déclenché 1 : créé_liste_joueur
Type : création d'une MCT
Etat : liste_joueur(Liste)

Elément déclenché 2 : détruire_attendre_réponse
Type : destruction d'une MCT
Etat : attendre réponse

Section 3 - Conclusion

Ce chapitre a présenté l'outil qui nous a permis de passer sans problème des diagrammes de transitions à un squelette de programme,et ceci par une simple définition des groupes transitoires dégagés des diagrammes de transitions. Pour cela, nous avons exposé chacun des éléments composant les groupes transitoires, puis la façon de les définir pour qu'ils soient compris du SGGT. Il ne reste plus qu'à compléter le corps de chaque processus pour obtenir un programme. A l'exécution, le SGGT s'occupe de gérer les préconditions d'activations de chaque processus, ainsi que les messages nécessaires à la coopération des agents.

Ce dernier chapitre nous a permis de montrer le but recherché dans le chapitre 4. En effet, les ajouts apportées aux diagrammes de la méthode EPAS permettent de définir directement chaque processus, leurs préconditions d'activation et leurs effets selon la terminologie exigée par le SGGT.

Conclusions

Pour conclure ce travail, nous allons tout d'abord retracé le chemin parcouru. Le but de ce mémoire était de concevoir une méthode de modélisation pour la résolution de problèmes dans l'environnement particulier que sont les systèmes multi-agents.

Nous avons donc en premier lieu explicité les systèmes multi-agents pour permettre aux différents lecteurs de mieux comprendre l'environnement dans lequel nous nous trouvions.

Nous avons ensuite exposé les différentes approches pour pouvoir résoudre un problème de manière distribuée, chacune de ces méthodes pouvant, évidemment, être modélisées plus ou moins facilement à l'aide de notre méthode.

La méthode EPAS fut présentée par la suite car celle-ci s'adaptait relativement bien aux problèmes multi-agents, et parce que celle-ci avait déjà fait largement ses preuves en intelligence artificielle.

Arrive ensuite l'élaboration de notre méthode, une méthode hybride composée par extraction de principes des approches objets pour la modélisation et de la méthode EPAS. Nous avons modifié cette dernière de telle sorte qu'elle s'adapte parfaitement à la résolution distribuée de problème. Le dernier point important de notre travail était le fait que la méthode devait pouvoir déboucher sur un outil informatique qui existait, à savoir le système générique de groupes transitoires (SGGT).

Pour terminer, nous expliquons brièvement cet outil pour montrer les gros avantages fournis par ce dernier.

C'est donc grâce à notre méthode pour la modélisation et grâce à SGGT pour la programmation que les problèmes sont résolus d'une manière distribuée beaucoup plus aisément.

Ce mémoire a fait suite à un stage à l'Université Laval de Québec où nous avons élaboré la méthode. Nous avons également, durant cette période, développé une application à l'aide de notre méthode et à l'aide du SGGT pour tester son bon fonctionnement (cfr. annexes pour les spécifications et le programme du jeu). Cette application a mis en évidence la facilité d'utilisation de la méthode et l'aisance de la programmation.

Nous croyons donc avoir atteint nos buts principaux, à savoir, la réalisation d'une méthode de modélisation pour la résolution de systèmes distribués qui soit claire, simple et efficace et ,de plus, une méthode qui débouchait sur des spécifications directement applicables au SGGT.

Mais nous ne devons pas nous en tenir à ces objectifs. En effet, il aurait été vain d'élaborer un tel travail sans penser au futur. Actuellement, le domaine informatique est partagé entre les approches objets pour la modélisation et les réseaux distribués pour le support physique. Nous terminons par ce point car nous pensons que la méthode posséde un avenir dans ce domaine informatique.

Bibliographie

CAMMARATA S., McARTHUR D., STEEB R. [1983]
"Strategies of Cooperation in Distributed Problem Solving,"
Proc. 8th Int. Joint Conf. Artificial Intelligence, Karlsuhe, West Germany, pp.767-770, 1983.

CHAIB-DRAA B., MILOOT P. [1987]
"Architecture pour les systèmes d'I.A. Distribuée,"
Proc. IEEE Compint 87, pp 64-69, Montréal, Québec, 1987.

CHANDRASEKARAN B. [1981]
"Natural and Social System Metaphors for Distributed Problem Solving : Introduction to issue,"
IEEE Trans. Syst., Man, Cybern., vol. SMC-11, pp. 1-5, Jan. 1981.

CLINGER W. [1981]
"Foundations of Actor Semantics,"
MIT Artificial Intelligence Laboratory, Cambridge, MA, Tech. Rep. 633, 1981.

CLOUTIER L. [1991],
'Système Générique de Groupes Transitoires',
développé à l'Université Laval, Août 1991.

DAVIS R. [1980]
"Report on the Workshop on Distributed A.I.,"
SIGART Newsletter., pp 42-43, Oct. 1980.

DAVIS R. [1982]
"Report on the Second Workshop on Distributed A.I.,"
SIGART Newsletter., pp 13-23, Avr. 1982.

DAVIS R., SMITH R.G. [1983]
"Negociations as a Metaphor for Distributed Problem Solving,"
Artificial Intelligence, vol. 20, pp.63-109, Jan. 1983.

DECKER K.S. [1987]
"Distibuted problem-Solving Tecniqueq : A survey,"
IEEE Trans. Syst., Man, Cybern., vol. SMC-17(5), pp. 729-740, Sept. 1987.

de LOOZ-CORSWAREM G. ET JOURQUIN P., [1990]
'Méthode de spécification et d'implantation de systèmes d'animation graphique (volume i et II)',
Institut d'Informatique, Facultés Universitaires Notre-Dame de la Paix, Namur, 1990.

ERMAN L.D., HAYES-ROTH F., LESSER V.R., REDDY D.R. [1980]
"The Hearsay II Speech-Understanding System : Integrating Knowledge to Resolve Uncertainty,"
Comput. Surveys, vol.12, pp. 213-253, June 1980.

FEHLING M., ERMAN L. [1983]
"Report on the Third Annual Workshop on Distributed Artificial Intelligence,"
SIGART Newslett., pp. 3-12, 1983.

FERBER J. [1987]
"Des Objets aux agents : une architecture stratifiée,"
Actes du 6ième Congrès RFIA, 1987.

FERBER J., GHALLAB M. [1988]
"Problèmatique des Univers Multi-Agents Intelligents,"
Proc. Journée nationales du PRC/IA, Cepadues-Editions, 1988.

FOGELMAN-SOULIE F. [1985]
"Cerveau et Machines; des architectures pour Demain ?,"
Proc. Forum Cognitiva Cesta, Paris, Juin 1985.

FOX M.S. [1981]
"An Organisation view of Distributed Systems,"
IEEE Trans. Syst., Man, Cybern., vol. SMC-11, pp. 70-80, Jan. 1981.

GENESERETH M.R., GINSBERG M.L., ROSENSCHEIN J.S. [1986]
"Cooperation Without Communications,"
Proc. 5th Nat. Conf. Artificial Intelligence, Philadelpia PA, pp.51-57, Aug.1986.

GOMEZ F., CHANDRASEKARAN B. [1981]
"Knowledge Organisation and Distributuion for Medical Diagnostics,"
IEEE Trans. Syst., Man, Cybern., vol. SMC-11, pp. 34-42, Jan. 1981.

HAINAUT J.-L., [1986]
'Conception assistée des applications informatiques, conception de la base de données',
tome 2, Masson, 1986.

HAYES-ROTH F. [1985]
"A Blackboard Architecture for Control, "
Artificial Intelligence, vol 26, pp. 251-321, 1985.

HEWIT C. [1976]
"Viewing Control Structures as Patterns of Passing Message,"
MIT Artificial Intelligence Laboratory, Cmbridge, MA, Dec; 1976.

HEWIT C., KORNFELD B. [1980]
"Message Passing Semantics,"
SIGART Newsett., p. 48, 1980.

HEWIT C., LIEBERMAN H. [1983]
"Design Issues in Parallel Architectures for A.I.,"
MIT Artificial Intelligence Lab., Cambridge, MA, Memo 750, 1983.

JACKSON M. [1983],
'System Developpement',
Prentice-Hall international, INC., 1983.

KORNFELD W.A., HEWIT C.E. [1981]
"The scientific Community Metaphor,"
IEEE Trans. Syst., Man, Cybern., vol. SMC-11, pp. 24-33, Jan. 1981.

LAWRENCE P.R., LORSCH J.W. [1967]
Organisation and Environement.
Boston, MA : Harvard University, 1967.

LESSER V.R., ERMAN L.D. [1980]
"Distributed interpretation : A Model and Experiment,"
IEEE Trans.on Computers, vol. C-29(12), pp. 1144-1163, 1980.

MEYER B. [1988],
'Object-Oriented Software Construction',
Prentice-Hall international, INC., 1988.

MOIGNE (LE) J.-L. [1990],
'La théorie du système général : la théorie de la modélisation',
3° édition, PUF Paris, 1990.

MOULIN B. [1988],
'La méthode EPAS pour la modélisation et la conception de systèmes',
Département d'Informatique, Université Laval, Québec, 1988.

PARUNAK V.D.H. [1985]
"Manufacturing Experience with the Contract Net"
Proc. 1985 Distributed Artificial Intelligence Workshop, pp.67-91, Dec. 1985.

ROLLAND C. [1992],
Séminaire donné à l'Institut d'Informatique aux Facultés Universitaires Notre-Dame de la Paix dans le cadre de la Chaire Francqui, Namur, 1992.

ROSENSCHEIN J.S., GENESERETH M.R. [1985]
"Deals Among Rational Agents,"
Proc. 9th Int. Joint Conf. Artificial Intelligence, Pittsburg,PA, pp.115-119, 1982.

SHERIDAN T.B. [1985]
"Forty five years of man-machine Syteme : History and Tends"
2nd IFAC Congress : Analysis, design and evolution of man-machine systems, Varese, 1985.

SHIMANOFF S.B. [1984]
"Coordinating Group Interaction via Communication Rules,"
In *Small Group Communication*, L.A.S.R.S. Catheart, Ed. Dubuque, IA : Brown. 1984.

SIMON H.A. [1957]
Models of man.
New York : Wiley, 1957.

SMITH R.G. [1980]
"The contract-Net Protocol : High-Level Communication and Control in Distributed Problem solver,"
IEEE Trans. on Computer, vol. C-29(12), pp. 1104-1113, Dec 1980

SMITH R.G., DAVIS R. [1981]
"Frameworks for Cooperation in Distributed Problem Solving,"
UMI Research Press, Jan. 1981.

SMITH R.G. [1985]
"Report on the 1984 Distributed Artificial Intelligence Workshop,"
AI Mag., Vol.6, pp. 234-243, Fall, 1985.

SOMMERVILLE I. [1989],
Software Engineering,
International Computer Science Series, 3° édition, 1989.