UNIVERSITÉ DU QUÉBEC À MONTRÉAL

AGENTS ET SYSTÈMES MULTI-AGENTS : VERS UNE SYNTHÈSE DE CES CONCEPTS

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN INFORMATIQUE

PAR

ALEXANDRE GROULS

MAI 2013

REMERCIEMENTS

Tout d'abord, je tiens à remercier Monsieur Roger NKAMBOU, professeur à l'UQÀM (Université du Québec à Montréal) pour m'avoir suivi tout le long de ma maîtrise informatique à l'UQÀM. Il a su m'aiguiller sur les domaines qui m'intéressent et me donner l'opportunité d'explorer ces domaines et d'aller jusqu'au bout. Il a également su répondre à mes interrogations et me guider vers l'aboutissement du présent mémoire. Je lui suis également redevable de la grande patience qu'il a eue à mon égard, car quelques fois soumise à rude épreuve.

Je tiens également à remercier Lise ARSENAULT pour son soutien général durant mon cursus de maîtrise, mes professeurs et collègues de laboratoire qui m'ont permis d'acquérir des connaissances aussi diverses que variées mais nécessaires à la poursuite de recherches en informatique, ainsi que ma famille qui m'a soutenu jusqu'au bout dans la poursuite de mes idées, de mes recherches et de ma soif de connaissances.

Ce mémoire a été une lourde tâche dont je n'avais pas idée, et il m'a permis de découvrir tout un aspect de l'intelligence artificielle qui m'était inconnu jusque-là dans lequel il est fastidieux d'avancer.

À mon tour de remercier toutes ces personnes en leur présentant ce mémoire, qui est une forme d'aboutissement de mon cursus à l'UQÀM.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

RÉSUMÉ

Les systèmes multi-agents appartiennent à un domaine de l'intelligence artificielle et ce sont des systèmes que l'on appréhende très différemment de l'ingénierie informatique classique. Les systèmes multi-agents interviennent là où la résolution classique des problèmes grâce à l'informatique a ses limites.

Ce domaine est malheureusement peu exploité aujourd'hui compte tenu des possibilités qu'il offre dans de nombreux domaines comme les sciences sociales, sciences informatiques, sciences expérimentales ou encore l'industrie. Mais les limites des systèmes informatiques et industrielles actuels sont telles qu'il devient envisageable et même intéressant de développer des systèmes multi-agents pour répondre aux besoins croissants de nombreux domaines plus classiques, que ce soit en termes de temps, d'efficacité ou de productivité.

Nous allons tout d'abord commencer par le concept d'agent, qui est l'élément fondamental pour concevoir des systèmes multi-agents. Nous verrons les divers types et catégories d'agents, ainsi que les architectures typiques qui leurs sont associés comme BDI, IDA ou CTS.

Puis nous allons voir les notions concernant les systèmes multi-agents, comme la notion d'interaction qui est une des pièces maîtresses avec les agents pour concevoir un système multi-agents. Avec les interactions viennent des phénomènes d'auto-organisation, et on verra différent modèles d'organisation ainsi que plusieurs niveaux d'organisation dans les systèmes multi-agent.

Enfin nous verrons différents outils, plateformes et langages adaptés à la conception de systèmes multi-agents, pour ce qui est de la structure des agents ou de l'aspect interactions et communications. Puis nous ferons quelques recommandations méthodologiques concernant le développement de systèmes multi-agents dans leur globalité.

Mots-clés : agent, système multi-agents, systèmes adaptatifs, organisation émergente, cycle cognitif, intelligence artificielle distribuée.

INTRODUCTION

Ce mémoire a été écrit et rédigé dans le cadre de ma maîtrise en informatique à l'UQÀM (Université du Québec à Montréal).

L'intelligence artificielle est un domaine très vaste et il est facile de s'y disperser. Mais les objectifs de l'intelligence artificielle sont vraiment concrets. La réalisation de systèmes pouvant s'adapter à un environnement dynamique, comme de la robotique pour l'exploration spatiale, en fait partie. De même que des systèmes intelligents et autonomes pour la prise de décision, dans le domaine militaire ou de la prise de risque bancaire par exemple. La réalisation de systèmes experts en fait également partie, notamment dans le domaine de la médecine ou de l'industrie.

Il existe donc plusieurs courants dans le domaine de l'intelligence artificielle, dont un qui permet d'appréhender, de manipuler et de concevoir des formes d'intelligence : les systèmes multi-agents. Ce courant de l'intelligence artificielle a également pour objectif de permettre la distribution de l'intelligence, cela permettant une conception plus ouverte de systèmes dits intelligents. La recherche dans ce courant contribue à la compréhension de l'organisation de ce type d'intelligence et permet d'apporter des ouvertures sur la résolution de problèmes non conventionnels.

Ce mémoire est l'expression de mon intérêt pour le domaine de l'intelligence artificielle en général et des systèmes multi-agents en particulier, et ma tentative de transmettre cet intérêt à ceux portés vers la conception de solutions dites intelligentes. Il se veut une compilation de la littérature car celle-ci est très éparse et il est facile de s'y perdre. Il se veut également une sorte de guide pour aider dans les choix de tels systèmes : bien choisir la typologie et l'architecture des agents, puis bien choisir l'organisation de son système multi-agents, mais aussi choisir les outils appropriés.

Dans ce mémoire, ce sont les systèmes multi-agents que nous allons étudier. Nous allons tout d'abord commencer par l'élément fondamental de tout système multi-agents : l'agent, les définitions associées, ses typologies, ses architectures. Dans un deuxième temps nous allons changer de point de vue pour voir comment est défini un système multi-agents, en étudiant les types d'organisation, les différentes architectures, ainsi que l'importance des interactions et des stratégies de coordination au sein d'un tel système. Puis des recommandations seront émises, afin d'aider le lecteur dans ses choix pour la conception d'un système multi-agents.

Cette étude a pour but de mettre en évidence clairement tous les aspects des systèmes multi-agents afin de savoir par où commencer et comment faire pour mettre en œuvre ce type de système.

CHAPITRE I

AGENTS : DEFINITIONS, TYPOLOGIES, ARCHITECTURES, APPLICATIONS

Avant d'aborder l'aspect des systèmes multi-agents, il convient de se concentrer sur les agents, qui sont en quelque sorte les briques fondamentales de tout système multi-agents. Dans un premier temps, en section 1.1, nous allons voir chacun des concepts que la notion d'agent utilise, le tout illustré d'exemples. Ensuite, dans la section 1.2, nous allons nous intéresser aux différentes typologies et architectures que peut avoir un agent et aux différentes catégories auxquelles appartient un agent. Puis, dans la section 1.3, nous allons voir les différents modèles d'architecture pour des agents, selon leur type et les besoins que l'on a en termes d'agent et plus tard en termes de système multi agents. Dans la section 1.4, nous allons explorer différentes implémentations d'agents pour plusieurs types d'application. Enfin, dans la section 1.5, nous allons aborder les difficultés et limites aux agents.

1.1 Définitions

Dans cette première section nous allons voir chaque notion qui appartient au concept d'agent et qui permet de définir celui-ci. Nous allons voir ce qu'est un agent, puis un agent intelligent et ses différences avec un agent simple. Ensuite nous allons voir leurs caractéristiques, c'est-à-dire leurs capteurs et perceptions, leurs effecteurs, leurs comportements, mesures de performance, mais aussi la rationalité, les objectifs, l'autonomie, le raisonnement et la communication.

1.1.1 Agent

Tout d'abord, voici quelques définitions de ce qu'est un agent, selon différents auteurs :

Selon S. Russell et P. Norvig : « On appelle agent toute entité qui peut être considérée comme percevant son environnement grâce à des capteurs et qui agit sur cet environnement via des effecteurs. » (Russell et Norvig 2006).

Selon M. Wooldridge, « An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives. » Ce qui une fois traduit nous donne : un agent est un système informatique situé dans un environnement donné, et qui est capable d'agir de manière autonome dans cet environnement en fonction des objectifs qui lui sont définis (Wooldridge 1999).

J. Ferber a une définition très détaillée prenant en compte toutes les composantes d'un agent : « On appelle agent une entité physique ou virtuelle a) qui est capable d'agir dans son environnement, b) qui peut communiquer directement avec d'autres agents, c) qui est mue par un ensemble de tendances (sous la forme d'objectifs individuels ou d'une fonction de satisfaction, voire de survie, qu'elle cherche à optimiser), d) qui possède des ressources propres, e) qui est capable de percevoir (mais de manière limitée) son environnement, f) qui ne dispose que d'une représentation partielle de cet environnement (et éventuellement aucune), g) qui possède des compétences et des offres de services, h) qui peut éventuellement se reproduire, i) dont le comportement tend à satisfaire ses objectifs, en tenant comte des ressources et des compétences dont elle dispose, et en fonction de sa perception, de ses représentations et des communications qu'elle reçoit. » (Ferber 1995).

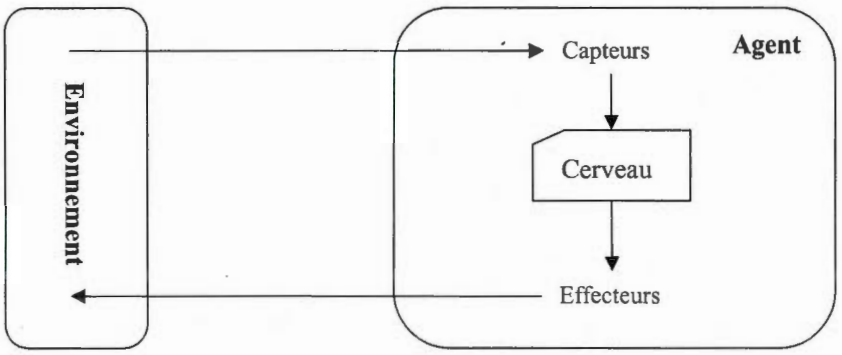


Figure 1.1 Un agent interagit avec son environnement grâce à ses capteurs et ses effecteurs.

Dans la littérature, chacun a une définition d'un agent plus ou moins large. Selon la définition de M. Wooldridge, un agent est purement limité à une entité informatique (Wooldridge 1999). Alors que la définition de S. Russell et P. Norvig est beaucoup plus ouverte et comprend les agents biologiques, contrairement à d'autres définitions restreintes au domaine de l'informatique ou du logiciel (Russell et Norvig 2006). On peut également noter que l'on n'évoque pas les agents intelligents, mais uniquement les agents. Cette distinction sera expliquée plus loin dans les sections 1.1.2 Agent intelligent et 1.1.6 L'autonomie d'un agent.

Une définition souple, voir la figure 1.1, d'un agent pourrait être : Un agent est une entité située dans un environnement, qui peut percevoir cet environnement grâce à des capteurs, qui peut agir sur cet environnement grâce à des effecteurs, et qui a un ou des objectifs.

Voici quelques exemples d'agents pour illustrer les précédentes définitions :

- Un robot d'exploration internet qui a pour objectif d'indexer du contenu pour un moteur de recherche, en naviguant de site internet en site internet et en analysant le contenu sémantique.
- Un virus biologique dont l'objectif est de se reproduire dans son environnement (un animal ou un être humain) en utilisant les cellules et leurs ressources.
- Un être vivant qui dispose d'yeux, d'oreilles (et d'autres organes sensoriels) comme capteurs, des membres supérieurs et inférieurs comme effecteurs, et qui a pour objectif de survivre et de se reproduire.

Il est intéressant de remarquer qu'un agent n'est pas un objet. On résumera l'exposé de M. Wooldridge qui revient à présenter trois différences fondamentales (Wooldridge 1999). La première différence est qu'une entité extérieure peut manipuler directement l'état interne d'un objet (avec des attributs publics par exemple) alors qu'on ne peut pas manipuler l'état interne d'un agent ni l'obliger à effectuer une fonction.
La deuxième différence est le caractère autonome du comportement d'un agent contrairement à un objet qui reçoit des ordres.
La troisième différence concerne l'exécution des comportements. Dans le cas d'un agent, l'exécution ne dépend que de lui. Alors que dans le cas d'un objet, cela dépend du système

(plateforme logicielle par exemple) auquel il appartient et qui planifie ses temps et durées d'exécution.

1.1.2 Agent intelligent

Parler d'agent intelligent nous amène à nous questionner sur ce qu'est l'intelligence, ainsi que sur la différence entre un agent intelligent et un simple agent. Pour M. Wooldridge, un agent est considéré comme intelligent lorsqu'il est capable d'effectuer des actions flexibles de manière autonome par rapport à ses objectifs (Jennings 1995, Wooldridge 1999). Et par flexible il signifie la réactivité, la pro-action, et la sociabilité :

- La réactivité : l'agent intelligent doit réagir dans un temps raisonnable aux changements de l'environnement.
- La pro-action : l'agent intelligent prend l'initiative d'un comportement tendant vers un but précis.
- La sociabilité : l'agent intelligent doit être capable d'interagir avec d'autres agents intelligents.

Il en ressort une caractéristique très importante et propre à un agent intelligent, c'est l'*autonomie*.

1.1.3 Environnement

L'environnement est l'univers dans lequel l'agent évolue et effectue des tâches. Et quand on parle d'environnement, on parle d'environnement de tâches. Et ces environnements de tâches sont en fait les problèmes que les agents vont devoir résoudre. O. Boissier synthétise plusieurs descriptions d'un environnement, qui peut être un espace de déplacement, un ensemble de ressources et de données disponibles, un milieu d'interaction avec ses propres lois, un espace où sont réalisées les actions, etc (Boissier 2001 : en ligne). En résumé, bien définir l'environnement revient à bien définir le problème, ce qui est primordial avant toute tentative de résolution de problème.

L'environnement joue donc un rôle très important en ce qui concerne le comportement d'un agent, selon L. Magnin qui identifie la dualité agent/environnement comme on peut le voir sur la figure 1.1 avec la partie cerveau qui commande les actions, et la partie agent qui est

l'enveloppe physique en contact avec l'environnement (Magnin 1996). Les caractéristiques des environnements doivent donc être spécifiées car on doit pouvoir être en mesure de distinguer ce qui concerne l'environnement et ce qui concerne le comportement de l'agent.

Nous allons tout d'abord voir comment définir clairement un environnement d'une tâche, notamment avec l'approche PEAS[1]. Puis nous allons présenter les différentes caractéristiques permettant de définir des environnements.

1.1.3.1 Approche PEAS

Lorsque l'on veut spécifier un environnement d'une tâche, il faut que l'on spécifie en son sein la mesure de performance, l'environnement, les capteurs et effecteurs de l'agent. On emploiera l'acronyme PEAS pour décrire cette spécification de l'environnement d'une tâche (Russell et Norvig 2006). Nous allons prendre en exemple un agent livreur robotisé de pizzas, dont l'environnement est plutôt complexe. L'agent est alors considéré comme un robot automatique chargé de livrer de la pizzeria aux clients. Le principal problème est la conduite, qui est une tâche plutôt ardue. Ajoutons aussi les objectifs d'économie de temps et de carburant. Le tableau 1.1 présente à titre d'exemple une description PEAS de l'environnement de tâche d'un livreur robotisé de pizzas.

Tableau 1.1 Description PEAS de l'environnement de tâche d'un livreur de pizzas

Type d'agent	Mesure de performance	Environnement	Capteurs	Effecteurs
Livreur robotisé de pizzas	Rapidité, respectueux du code de la route, sécurité, maximisation du profit	Routes, trafic (véhicules et piétons), clients	Accéléromètre, GPS, compteur de vitesse, caméras, sonar, thermomètre, sondes du moteur	Accélérateur, freins, direction, clignotants, klaxon, résistance, module de télépaiement

[1] Acronyme anglais utilisé dans la littérature (notamment par S. Russell et P. Norvig ainsi que O. Boissier) signifiant : Performance, Environment, Actuators, Sensors (soit les termes : Performance, Environnement, Capteurs, Senseurs).

1.1.3.2 Propriétés des environnements

Le nombre d'environnements de tâche possible est très grand, mais il est possible de catégoriser les environnements selon certaines propriétés, au nombre de six telles qu'elles sont présentées par S. Russell et P. Norvig (Russell et Norvig 2006). Ces propriétés jouent un rôle dans le type d'agent qui peut être conçu pour certains ensembles d'environnements.

- Entièrement observable / partiellement observable :
  Si l'agent a accès à tout instant à tous les états de l'environnement grâce à ses capteurs, alors son environnement de tâche est entièrement observable. Cela est intéressant car l'agent n'a pas besoin d'avoir une représentation interne de son environnement. Dans les autres cas, l'environnement de tâche est partiellement observable. Par exemple, l'environnement de tâche d'un robot d'exploration sur Mars est partiellement observable, car l'agent ne peut observer que dans un certain périmètre autour de lui.

- Déterministe / stochastique :
  Quand le prochain état de l'environnement est déterminé par l'action d'un agent et l'état courant de l'environnement, l'environnement est déterministe. Dans les cas où l'on fait exception des actions des autres agents, on dit qu'il est *stratégique*. Dans tous les autres cas, on dit qu'il est stochastique, et il comprend la notion d'aléatoire.

- Épisodique / séquentiel :
  Dans un environnement épisodique, le comportement de l'agent est découpé en épisodes atomiques. À chaque épisode est associée une action spécifique, et toutes les actions sont indépendantes, n'ont aucune influence entre elles. Dans un environnement séquentiel, les prochaines actions dépendent des actions précédemment réalisées. Les environnements séquentiels sont plus simples du fait que l'agent n'a pas à s'occuper de ce qu'il pourrait se passer après.

- Statique / dynamique :

  Si l'environnement ne change pas pendant que l'agent réfléchit ou agit, c'est un environnement statique, car l'agent n'a pas besoin de continuer d'observer son environnement et n'a pas de contrainte de temps. Au contraire, si l'environnement peut changer pendant la phase de réflexion ou d'action de l'agent, c'est un environnement dynamique car l'agent doit constamment prendre en compte les percepts qui peuvent influencer ses choix d'action. Il se peut que l'on puisse définir un environnement comme semi-dynamique, par exemple quand l'environnement ne change pas en fonction du temps mais quand la performance d'un agent dépend du temps (l'exemple du joueur d'échec avec chronomètre).

- Discret / continu :

  Un environnement est discret quand les états de cet environnement sont distincts, même s'il y en a une infinité. À titre d'exemple, une partie d'échecs est un environnement discret. La manière de gérer le temps ou de considérer les percepts et les actions d'un agent déterminent aussi si l'univers est discret ou continu. Dans notre exemple précédent du livreur robotisé de pizzas, l'environnement est continu car les états de l'environnement et le temps dans l'environnement sont continus, ainsi que les percepts de l'agent.

- Mono-agent / multi-agents :

  L'environnement est mono-agent lorsqu'un seul agent évolue dedans, et il est multi-agents lorsqu'il comprend au moins deux agents. Cette propriété de l'environnement est importante dans le cas où il est multi-agents car il va falloir détailler si l'environnement multi-agents est *concurrentiel* ou s'il est *coopératif*. Une partie d'échecs est un environnement purement concurrentiel car un seul des deux agents peut atteindre l'état recherché, la victoire, provoquant automatique la défaite et donc l'échec de l'autre. Un environnement où les agents évoluent sans s'affronter, comme dans un trafic routier par exemple, est coopératif. Il se peut qu'un environnement coopératif soit concurrentiel en partie, comme le trafic routier dans lequel le nombre de places de stationnement est limité et où les agents peuvent stationner. Dans les environnements multi-agents, on peut voir l'aspect de la *communication*, qui est alors un comportement rationnel.

Tableau 1.2 Quelques environnements de tâches avec leurs propriétés

Environnement de tâche	Observable	Déterministe	Episodique	Statique	Discret	Agents
Partie d'échecs	Entièrement	Stratégique	Séquentiel	Statique	Discret	Multi
Partie d'échecs chronométrée	Entièrement	Stratégique	Séquentiel	Semi-dynamique	Discret	Multi
Livreur de pizzas	Partiellement	Stochastique	Séquentiel	Dynamique	Continu	Multi
Mots-croisés	Entièrement	Déterministe	Séquentiel	Statique	Discret	Mono
Conseiller d'achats sur internet	Partiellement	Stochastique	Episodique	Statique	Discret	Mono

Il est alors aisé d'identifier les propriétés de l'environnement le plus difficile, à savoir : partiellement observable, stochastique, séquentiel, dynamique, continu et multi-agents ; ce qui représente un environnement très complexe. Le monde réel étant très complexe, il possède ses propriétés et est un environnement très difficile.

1.1.4 Capteurs et perceptions chez un agent

Les capteurs sont une caractéristique fondamentale des agents, et notamment pour la notion de perception. Les capteurs sont des mécanismes sensoriels permettant à l'agent de disposer d'un certain type d'information sur ce qui l'entoure. Voici quelques exemples de capteurs classiques par rapport à des sens connus chez l'être humain : des yeux ou des détecteurs infrarouges pour la vision, des oreilles ou un micro ou des détecteurs de pressions de l'air pour l'audition, des détecteurs de pressions pour le toucher, des détecteurs d'acidité ou de salinité pour le goût.

La fonction d'un capteur est de transformer un phénomène physique en un signal. Ce signal, qui est la forme d'un stimulus (ou de son absence) qui est perçue par un capteur, est ce que l'on appelle un percept. Nous utiliserons le terme de *percept* pour signifier ces données sensorielles reçues par l'agent, ou autrement dit ses perceptions.

Il est intéressant de noter la différence entre un percept et un stimulus. Le stimulus est le phénomène physique qui peut déclencher un percept, tandis que le percept est l'information

perçue suite à la conversion par le capteur. Ainsi, un stimulus peut ne pas être converti en percept, de même qu'un percept peut exister lors de l'absence de stimulus dans un cas précis.

1.1.5 Effecteurs

Les effecteurs sont, à l'instar des capteurs, une autre caractéristique fondamentale des agents. Ils permettent à l'agent d'agir sur son environnement. Selon le niveau de détail utilisé et selon le type d'agent étudié, la définition peut varier. Un effecteur peut être :

- Un outil, une perceuse par exemple.
- Un organe de préhension[2], comme une pince mécanique.
- Les membres locomoteurs, permettant à l'agent de se déplacer dans son environnement, sont aussi des effecteurs.
- Un morceau de code qui exécute une action menant à la modification de l'environnement.

Mais il est possible de donner une définition générale couvrant tous les types d'effecteurs : un effecteur est un dispositif propre à l'agent, lui permettant l'action, la locomotion ou la préhension dans et sur son environnement.

1.1.6 L'autonomie d'un agent

Pour comprendre ce qu'est l'autonomie chez les agents, nous allons voir ce qui concerne les connaissances ainsi que l'apprentissage, le tout était fortement lié à la rationalité telle que vue dans la section 1.1.7.2. Rationalité. Par ailleurs, comme nous l'avons brièvement vu dans la section 1.1.1. Agent, l'autonomie est une des caractéristiques qui différencie un agent d'un objet (Wooldridge 1999, Weiss 1999).

Tout d'abord il y a plusieurs manières pour un agent d'avoir des connaissances. Il y a celles qui lui sont innées, c'est-à-dire celles que les concepteurs auront implantées au préalable, et il y a celles provenant des percepts, que l'agent peut acquérir (Ferguson 1994). Pour acquérir

---

[2] Un organe de préhension défini un membre, organique ou mécanique, permettant la saisie d'objets dans l'environnement : la pince d'un robot ou la main d'un être humain sont des organes de préhension.

des connaissances, il faut un comportement rationnel permettant de récolter de l'information. L'*exploration* est un comportement rationnel d'un agent, lui permettant l'acquisition de connaissances sur la partie observable de l'environnement. Il est possible pour un agent d'agir avant d'avoir des informations, dans le but d'acquérir ces informations par la modification des futurs percepts, comme par exemple regarder avant de traverser une route, tout ceci ayant bien sûr pour but de maximiser la performance de l'agent. Ce procédé, qui est donc rationnel, visant à permettre la modification des futurs percepts est appelé *collecte d'informations*.

Ensuite, le concept de rationalité fait que l'apprentissage devient nécessaire, l'agent doit apprendre à partir de ces percepts. L'apprentissage permet d'accumuler des connaissances sur l'environnement, mais permet aussi de modifier des comportements, des actions, afin d'affiner les comportements et maximiser la performance. Cette accumulation de connaissances et de modifications des comportements définit l'*expérience* propre à un agent (Russell et Norvig 2006).

Moins un agent a d'expérience, et plus il dépend des connaissances innées (celles qui sont définies par ses concepteurs), alors il manque d'*autonomie*. L'autonomie d'un agent est donc liée à sa dépendance aux connaissances innées ou à celles qu'il se construit par lui-même grâce à ses percepts. En résumé, moins un agent dépend de connaissances qu'on lui écrit et plus il est *autonome*.

1.1.7 Comportement, mesure de performance et rationalité chez un agent

Pour S. Russell et P. Norvig, le concept de rationalité est lié à ce qui est défini comme un bon comportement chez un agent (Russell et Norvig 2006). L'objectif ici est de concevoir des agents rationnels, c'est-à-dire des agents qui sont capables de toujours effectuer la ou les actions appropriées pour chaque cas. Le fait d'analyser si une action est appropriée ou non nous amène à la performance d'un agent, et de la mesure de cette performance.

Dans un premier temps, nous allons voir en point 1.1.7.1 la mesure de performance. Cela va nous permettre de comprendre et définir la rationalité dans le point 1.1.7.2.

1.1.7.1 Mesure de performance

La mesure de la performance nécessite de connaître les capteurs et les effecteurs de l'agent, ainsi que son environnement. Cela va nous permettre de décrire la tâche que l'agent doit effectuer. On considère que la mesure de performance concerne toujours une tâche, et qu'il y a donc par ailleurs autant de mesures de performance que de tâches différentes.

Un critère est pris en compte pour mesurer la performance d'une tâche d'un agent : c'est le succès, ou non, de son comportement. Dans son environnement, l'agent reçoit des percepts et peut donc générer une séquence d'actions. Cette séquence d'actions définit en quelque sorte son comportement pour la tâche à effectuer. Et une manière simple de définir le succès d'un comportement est de voir si l'environnement est bien dans l'état recherché, ou passe bien par les états voulus, après l'application du comportement par l'agent. On en déduirait donc le bien fondé ou non du comportement en fonction de ces différents états de l'environnement. S. Russell et P. Norvig insistent fortement sur l'objectivité de la mesure de performance telle que définie par le concepteur d'un agent car la mesure dite subjective pourrait ne pas être en adéquation avec la mesure objectif (Russell et Norvig 2006). Bien entendu, il est tout à fait possible d'imaginer utiliser des mesures subjectives de performances différentes et complémentaires.

Il est important de bien définir les critères de la mesure de performance en fonction des comportements que l'on veut obtenir, car des comportements irrationnels et ne répondant pas à l'objectif souhaité pourraient apparaître tout en satisfaisant la mesure de performance. Or, ce qui nous importe le plus dans le comportement d'un agent est la réalisation d'un objectif (d'un changement d'état de l'environnement).

1.1.7.2 Rationalité

Plusieurs facteurs permettent de déterminer la rationalité à un instant donné :

- L'ensemble des connaissances (acquises ou innées) dont l'agent dispose sur son environnement.
- L'ensemble des actions qu'il peut effectuer dans cet environnement.
- La séquence de percepts à l'instant donné.

- La mesure de performance définissant le succès ou non d'une tâche.

Suite à l'énumération des quatre facteurs permettant de déterminer ce qui est rationnel ou non, nous pouvons clairement définir ce qu'est un agent rationnel tel que S. Russell et P. Norvig l'ont proposé : « Pour chaque séquence possible de percepts, un agent rationnel doit sélectionner une action susceptible de maximiser sa mesure de performance, compte tenu des observations fournies par la séquence de percepts et de la connaissance dont il dispose. » (Russell et Norvig 2006).

1.1.8 L'objectif chez un agent

Nous avons vu que les agents sont le moyen de résoudre un problème. Le besoin exprimé est la résolution d'une problématique, ce qui définit un outil permettant de satisfaire ce besoin. Cet outil est un agent, et il peut donc avoir un ou plusieurs objectifs (ou buts) car c'est ce qui définit sa fonction, sa raison d'exister.

Chez un agent, l'objectif est ce qui le pousse à agir. Nous pouvons nous demander ce que ferait un agent sans fonction, sans objectif à satisfaire. Il est possible qu'il ne fasse rien, il est possible qu'il agisse de manière chaotique ou incohérente, mais son comportement ne paraîtrait en rien rationnel pour l'environnement dans lequel il se trouve du fait de son absence de raison d'être.

1.1.9 La communication chez un agent

Chez les agents, la communication est une forme d'interaction (nous verrons plus en détails la notions d'interaction en section 2.1.2 Interactions). Cette possibilité de communiquer permet l'échange d'informations, et donc de connaissances.

La communication rend également possible la coopération entre agents, par exemple quand plusieurs agents ont des objectifs complémentaires.

Bien entendu, pour que la communication soit effective chez les agents, il faut qu'ils aient un protocole commun de communication ainsi qu'un canal commun et ce, afin d'assurer la cohérence dans les échanges d'informations.

1.1.10 Le raisonnement d'un agent

De manière générale, le raisonnement se définit comme étant un processus cognitif permettant de vérifier des prédicats ou bien d'en obtenir de nouveaux grâce à des mécanismes logiques comme l'inférence, l'induction, l'abduction, la déduction.

Dans la figure 1.1, la partie raisonnement peut être identifiée à la partie « Cerveau ». Chez les agents rationnels, le raisonnement peut utiliser ces différentes logiques à condition qu'ils aient été conçus avec cette possibilité. Nous verrons dans les sections 1.2 et 1.3 les typologies et architectures des agents, qui permettent à l'agent d'avoir des raisonnements adaptés à leur environnement de tâche.

1.2 Typologies des agents

Dans cette partie nous nous concentrerons sur les différentes catégories d'agents, en termes de modèle, ainsi qu'aux différents types de comportements propres à chaque catégorie d'agent, puis nous verrons chaque type d'agents ayant tel comportement et appartenant à telle catégorie. Ces classifications nous permettrons d'identifier plus facilement les idées propres à chaque architecture d'agent. Bien entendu, il existe des agents qui possèdent des caractéristiques des deux catégories. Ceux-là sont appelés des agents hybrides.

1.2.1 Catégories d'agents

Les agents sont répartis en deux grandes catégories : les agents réactifs et les agents cognitifs. La différence entre ces deux catégories d'agents concerne la représentation que peut avoir un agent de son environnement.

1.2.1.1 Agents réactifs

La catégorie des agents réactifs, décrite par R. A. Brooks, comprend tous les agents dont la représentation de l'environnement n'est que *sub-symbolique* (Brooks 1991). Cela signifie que la représentation provient uniquement des perceptions de l'agent, du monde « visible » à l'instant courant. Il n'y a pas de raisonnement à proprement parler dans cette catégorie d'agents puisque l'on est dans une configuration du type stimulus/action : stimulus → percept → action. L'agent réagit aux évènements dans l'environnement mais n'a

pas de mémoire et ne peut donc ni prendre le passé en compte, ni prévoir au-delà du court terme.

Le principe d'un agent réactif permet la construction de systèmes composés de nombreux petits agents, qui sont des automates. Les interactions des agents entre eux permettent l'émergence de structures d'une couche d'abstraction supérieure qui sont potentiellement observables (Lestel et al. 1994). Leur conception n'est pas si facile du fait de la difficulté de prévoir les phénomènes d'émergences.

1.2.1.2 Agents cognitifs

Contrairement aux agents réactifs, la catégorie des agents cognitifs comprend tous les agents disposant d'une représentation *symbolique* de leur environnement. Les agents cognitifs sont capables de raisonner sur leur représentation symbolique. Il existe des algorithmes pour manier les symboles, et donc les représentations symboliques, mais ils sont très complexes et limitent les capacités de ce type d'agents. Très souvent, cette classe d'agents s'inspire de théories de la cognition, principalement humaine. Ce sont ces différentes théories de la cognition qui induisent les comportements des agents. Ainsi, plusieurs architectures cognitives peuvent être envisagées pour la création d'agents cognitifs, comme nous le verrons plus loin en section 1.3.

1.2.2 Comportements d'agents

Les agents ont deux comportements types : le comportement réactif, ou réflexe, et le comportement téléonomique. On distingue ces deux types de comportements sur les causes des comportements et non pas les comportements en eux-mêmes.

1.2.2.1 Comportement réflexe

Le comportement réflexe est le plus simple dans la mesure où l'action découle directement du stimulus, sans raisonnement à proprement parler, mais selon une règle que l'agent possède ou peut apprendre (par exemple : si X alors Y ; si A alors B). Il s'agit d'un arc réflexe comme la figure 1.2 le montre. Le comportement de l'agent provient donc directement de l'environnement.

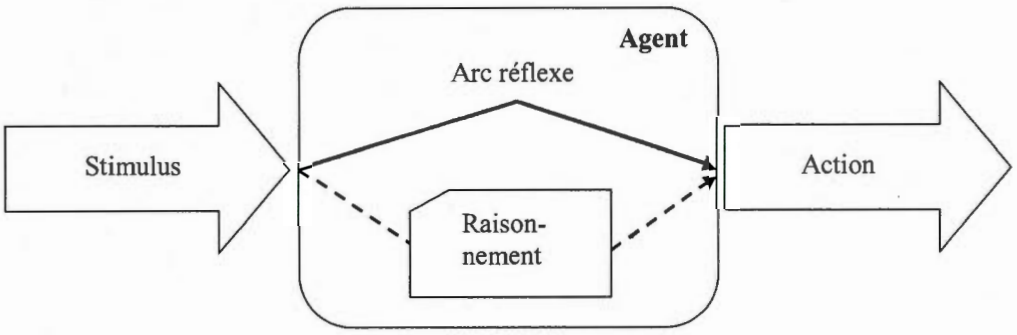


Figure 1.2 Opposition du comportement réflexe au comportement téléonomique.

1.2.2.2 Comportement téléonomique

Le comportement téléonomique est un comportement qui découle directement de l'objectif à atteindre pour un agent, de l'intention, de la finalité. On peut voir sur la figure 1.2, avec la flèche en pointillé, que l'agent a un percept provoqué par le stimulus, et qu'avant d'agir il raisonne (le détail de cette partie dépend de l'architecture utilisée pour concevoir l'agent). On est en totale opposition avec le comportement réflexe car ce qui cause le comportement est la partie de raisonnement, et non pas directement le stimulus. Le comportement téléonomique est un comportement guidé par les intentions et objectifs.

1.2.3 Classification des agents

De manière fondamentale, on peut identifier quatre familles principales d'agents dont chacune est une combinaison d'un comportement et d'une catégorie d'agent (Ferber 1995). Le tableau 1.3 synthétise cette classification contenant les familles des agents intentionnels, modules, pulsionnels et tropiques (Ferber 1995).

Tableau 1.3 Familles d'agents par catégorie et comportement, selon J. Ferber

	Catégorie d'agents réactifs	**Catégorie d'agents cognitifs**
Comportement réflexe	Agent tropique	Agent module
Comportement téléonomique	Agent pulsionnel	Agent intentionnel

1.2.3.1 Agent intentionnel

Les agents intentionnels sont des agents cognitifs dotés d'un comportement téléonomique. Ce sont des agents qui ont des objectifs qu'ils cherchent à accomplir. Pour atteindre ces objectifs, ils peuvent tout à fait accomplir des objectifs intermédiaires, qui ne sont que des étapes à franchir pour la réalisation de l'objectif final. Chez les agents intentionnels, il existe une intention forte, celle de réaliser les objectifs, ce qui fait intervenir une planification chez ce type d'agent.

1.2.3.2 Agent module

À l'instar des agents intentionnels, les agents modules sont des agents cognitifs et possèdent la même représentation symbolique de leur environnement. Cependant, ils n'ont pas d'objectifs, juste un comportement réflexe, ce qui fait qu'il est difficile de trouver des agents de ce type. Néanmoins, un rôle possible serait le rôle de ressource. C'est-à-dire qu'ils peuvent très bien servir de source d'information pour d'autres agents qui les sollicitent.

1.2.3.3 Agent pulsionnel

Les agents pulsionnels sont des agents réactifs dotés d'un comportement téléonomique. Ce type d'agent est particulièrement bien adapté au maintien d'un objectif, comme une mission. Si l'état de l'environnement s'éloigne trop de l'objectif, c'est-à-dire que si j'objectif n'est plus maintenu, l'agent pulsionnel déclenche un comportement (comme un réflexe à chaque fois qu'il y a un écart) dans le but de maintenir l'objectif. L'objectif est ici une source de motivation interne.

1.2.3.4 Agent Tropique

Les agents tropiques sont des agents réactifs uniquement dotés d'un comportement réflexe, ce qui en fait des agents parmi les plus basiques qui soient. Un agent tropique agit uniquement de manière réflexe à l'état local de l'environnement, et surtout à l'instant auquel cela se passe. Une image simpliste d'un agent tropique mais néanmoins vraie le représenterait comme un agent se laissant porter au gré du vent ou par les flots. Ce type d'agent n'ayant pas de but, il n'agit que lors d'un changement de l'état local.

Les agents tropiques étant très limités, une évolution a été proposée en les dotant d'une mémoire afin qu'ils puissent acquérir de l'expérience. Il était possible pour les agents de marquer l'environnement afin de stocker une information, mais avec la mémoire on s'affranchit de ce marquage. Ceci étant proposé en tant qu'évolution des agents tropiques afin qu'ils puissent être plus efficace. Ces agents tropiques étendus sont appelés des *agents hystérétiques* (Bergeret 2007).

1.3 Architectures d'agents

Dans la littérature, cinq architectures sont couramment utilisées pour construire des agents, notamment des agents informatiques. La plupart de ces architectures se fonde sur une théorie de la cognition, dans le but de concevoir le cycle cognitif de l'agent. On peut dire que le cycle cognitif est le processus de délibération général se situant entre la perception et l'action, comme illustré sur la figure 1.3.

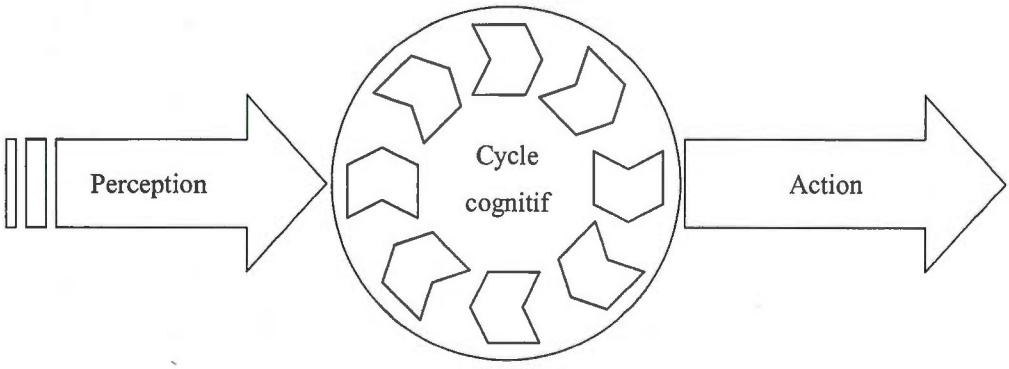


Figure 1.3 Situation du cycle cognitif chez l'agent

Nous verrons donc les cinq architectures suivantes : BDI, ACT-R, IDA, CTS et les modèles en couches avec notamment TouringMachines et InteRRaP. Ce sont ces différentes architectures qui servent à créer le cycle cognitif d'un agent.

1.3.1 BDI

L'architecture BDI[3] est un modèle d'architecture pour les agents basé sur trois concepts, ou attitudes mentales, que sont la croyance, le désir et l'intention. Ces trois concepts déterminent la rationalité particulière d'un type d'agent que l'on peut qualifier intelligent. Pour M. Bratman (Bratman 1987), ces trois concepts sont également centraux dans la théorie de la cognition sur laquelle se base l'architecture BDI, qui émet l'idée que les intentions jouent un rôle à part entière dans un raisonnement dit pratique basé sur des croyances et des désirs. Car ce type de raisonnement pratique permet de limiter les choix qu'un humain ou un agent peut faire à un moment donné. Cette théorie, formalisée dans une logique symbolique formelle, décrit :

- la nécessité d'actualiser le plus souvent possible voir en temps réel la base de connaissances issue des perceptions, et donc d'avoir une représentation valide de l'environnement en tout temps.
- un but ou un ensemble de buts afin de donner un sens à l'existence.
- un système de raisonnement, d'inférence par exemple, permettant le raisonnement à partir de la base de connaissances et des buts.
- une capacité à organiser des plans d'action, des comportements dans le but de satisfaire les objectifs, durant le cycle cognitif, ce que l'on peut appeler les intentions.
- une durée du cycle cognitif, du processus de délibération, processus qui peut être adapté au temps nécessaire pour percevoir les changements de l'environnement provoqués par les actions.

C'est en se basant sur cette théorie que certains chercheurs décrivent les trois concepts de l'architecture BDI (Georgeff et Rao 1991 et 1995).

Tout d'abord le B pour Belief, ou *Croyance* en français. Les croyances d'un agent sont tout simplement l'ensemble des connaissances que l'agent a sur son environnement. Le terme connaissance est abusif car ce sont des informations sur l'environnement qui sont issues des

---

[3] BDI est l'acronyme anglais de « Belief, Desire, Intention », pour l'architecture basée sur les concepts de croyance, désir et intention.

percepts de l'agent, et donc dépendent des capteurs. Ces informations peuvent donc ne pas être vraies ou correctes, mais le cycle cognitif de l'agent lui permet de mettre à jour ces informations pour que ses croyances sur son environnement soient les plus correctes et complètes possibles, par l'intermédiaire de ses perceptions mais aussi de ses actions.
Ensuite le D pour Desire, ou *Désir* en français. Les désirs d'un agent sont les états que l'agent cherche à atteindre. Ces états peuvent être ceux de l'environnement ou bien ceux de l'agent lui-même. Ces désirs sont l'ensemble des buts décrits dans la théorie résumée plus haut, et sont donc la raison d'être de l'agent basé sur une architecture BDI. Il est intéressant de noter que les désirs peuvent être contradictoires (comme vouloir manger et vouloir dormir), auquel cas le raisonnement permettra d'affiner les désirs afin de les rendre consistant[4].
Enfin le I pour *Intention*. Les intentions sont les choix d'actions décidés à l'issue de chaque cycle cognitif. Les désirs ne peuvent être réalisés que si l'agent planifie des actions pour les réaliser, lors du raisonnement. Bien entendu, tous les désirs ne peuvent pas être réalisés au même moment même s'ils sont consistants, et les intentions permettent d'ordonner la satisfaction de ces désirs, ou autrement dit la réalisation des objectifs.

La figure 1.4 schématise l'architecture BDI et son interprétation de la théorie de la cognition de M. Bratman, tout en prenant en compte l'environnement. Ici, la partie planification permet de structurer les intentions en exécutions organisées pour répondre aux buts.

L'architecture BDI permet donc de concevoir des agents dont le comportement est téléonomique du fait de l'intégration de la notion d'*intentions*. Les agents ainsi conçus étant du type cognitif car basés sur une théorie de la cognition, on peut dire que les agents BDI sont des agents intentionnels.

---

[4] Quand on parle d'un ensemble de désirs consistants, on parle d'un ensemble pouvant être réalisé sans contractions, au contraire d'un ensemble non consistant qui contient des désirs contradictoires. On peut donc choisir des comportements acceptables pour la résolution des objectifs.

Par ailleurs, il existe une extension à l'architecture BDI qui prend en considération un quatrième concept : l'obligation (Broersen et al. 2001). Les obligations sont des sources de motivation externes au contraire des désirs qui sont des sources de motivation internes. C'est le plus souvent une contrainte qui apparaît dans l'environnement par son état ou par le comportement d'autres agents. Ces obligations peuvent influer fortement le cycle cognitif, notamment selon l'ordre de traitement (priorité) mais aussi le poids (remplace ou non des croyances) au niveau du procès de délibération. Pour cela il existe six versions de cette extension, chacune définissant un type d'agent particulier : BOID, BODI, BIOD, BIDO, BDIO et BDOI. Ces six types d'agent sont également classés selon qu'ils soient stables, égoïstes ou sociaux. Cette extension à l'architecture BDI est utilisée pour la résolution de conflit entre les agents. On définit ce type d'agent comme normatif.

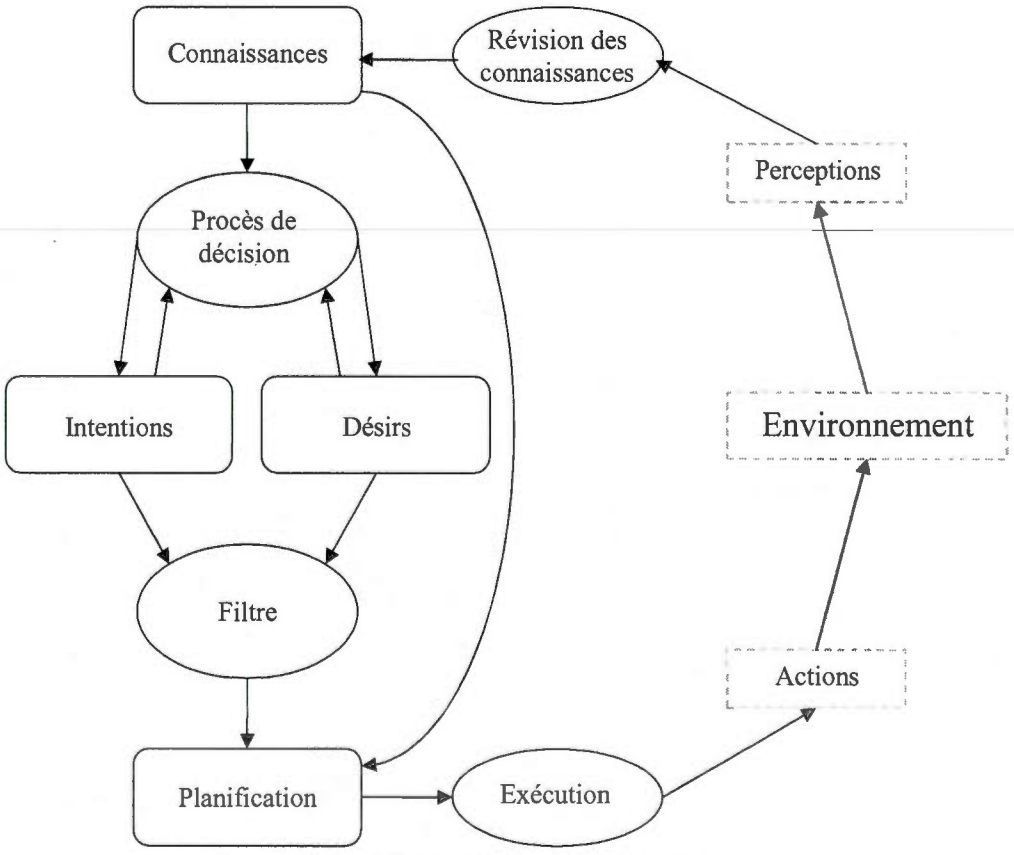


Figure 1.4 Architecture BDI simplifiée

1.3.2 ACT-R

ACT-R est une architecture cognitive créée et maintenue par J. R. Anderson depuis 1998 (Anderson et al. 2004, ACT-R : en ligne). Cette architecture est une évolution de la théorie cognitive ACT de J. R. Anderson développée en 1978 sur laquelle il ajoute le concept de la rationalité inhérente à la cognition humaine, et ACT est elle-même une évolution de HAM, toujours par J. R. Anderson et G. Bower en 1973 (Anderson et Bower 1973). ACT-R se base sur la théorie que les tâches effectuées par les humains sont une suite d'opérations atomiques issues de la perception et de la cognition, et ACT-R a pour but de définir ces opérations atomiques afin de créer des systèmes décisionnels s'approchant de la cognition humaine, en terme de raisonnement et de prise de décision.

ACT-R, de l'anglais « Adaptive Control of Thought-Rational », se définit comme une architecture de contrôle adaptatif de la pensée rationnelle. Cette architecture est basée sur une théorie cognitive définissant la production d'une cognition cohérente à l'aide de multiples modules, principalement classés dans les trois catégories suivantes : perception-motricité, buts, et mémoire. Le nombre des modules que l'on peut rajouter n'est pas limité et un module ajouté ne fait pas nécessairement partie d'une des trois catégories citées précédemment. L'ensemble des modules qui interagissent entre eux produit une cognition cohérente qui s'apparente à un système de production, notamment en ce qui concerne le processus délibératif.

Tout d'abord les modules de la catégorie perception-motricité sont généralement au nombre de deux : un module dédié aux perceptions de l'environnement, et un module dédié aux actions dans l'environnement.

Ensuite il doit y avoir au moins un module de la catégorie buts, et les modules de cette catégorie gèrent aussi l'aspect intentionnel. Il est intéressant de noter que ces modules peuvent être étendus dans les modules mémoires.

Enfin les modules de la catégorie mémoire sont également souvent au nombre de deux : un module de mémoire déclarative, comme une base de connaissances, de faits ou de règles, et

un module de mémoire procédurale, comme les connaissances inhérentes aux comportements que l'on peut aussi utiliser pour coordonner le cycle cognitif grâce à des règles de production.

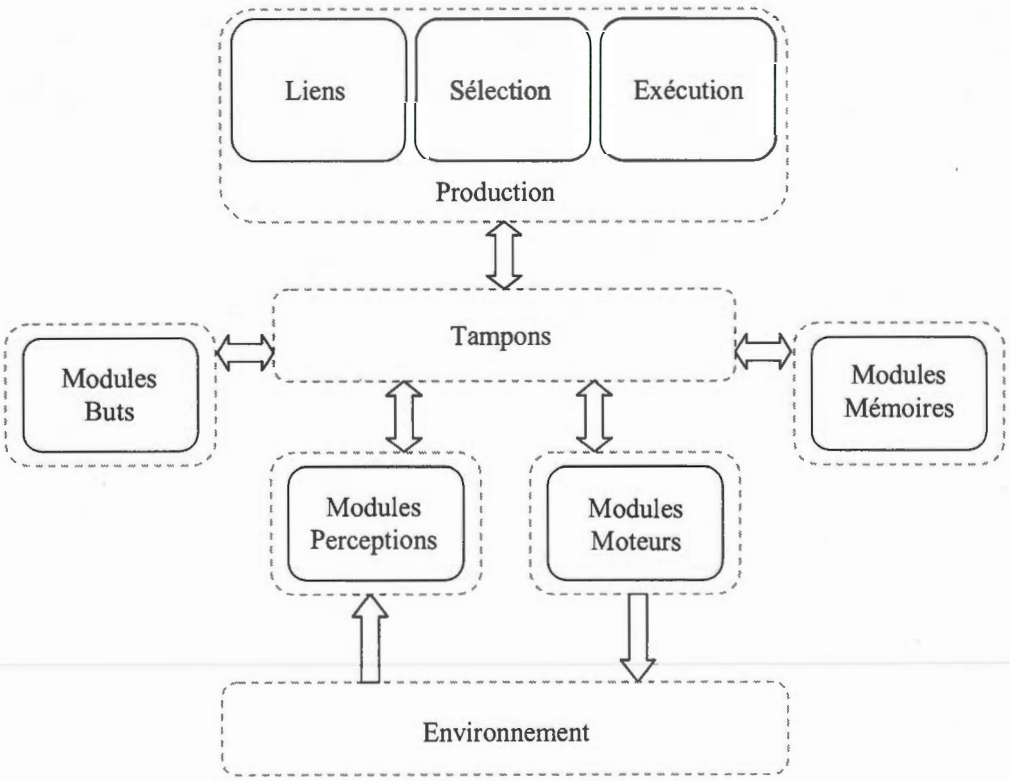


Figure 1.5 Architecture ACT-R simplifiée

Une notion particulière qui concerne le système *tampon*[5] se doit d'être prise en compte. En effet selon J. R. Anderson les modules traitent chacun une grande quantité d'informations et ce, de manière indépendante des autres modules. Le système central de production n'étant pas capable de traiter autant de données d'un coup, d'autant plus que toutes les données ne sont pas nécessaires au système de production selon les objectifs de l'agent, chaque module

[5] Le terme tampon provient de l'anglais buffer, terme régulièrement utilisé dans le domaine informatique pour ce qui est de définir un espace de travail pour des données temporaires, qui doivent par exemple transiter entre deux systèmes sans gêner le fonctionnement des dits systèmes.

dispose d'un tampon externe, ce qui permet au système de production d'avoir accès aux informations dont il a besoin lors du cycle cognitif sans gêner le fonctionnement indépendant des modules. Il est par ailleurs intéressant de noter que le concepteur peut jouer également sur la durée du cycle cognitif pour paramétrer l'agent par rapport à un environnement donné. Nous pouvons donc voir que le système de production fonctionne de manière séquentielle en utilisant une structure symbolique, et que les modules fonctionnent en parallèle et de manière totalement asynchrone utilise une structure sub-symbolique puisque leur fonctionnement interne influence le système de production par l'intermédiaire du tampon.

L'architecture ACR-T permet donc de concevoir des agents dont le comportement est téléonomique du fait de l'intégration d'au moins un module *buts*. Les agents ainsi conçus étant de type cognitif car basés sur une théorie de la cognition mais utilisant une structure symbolique ainsi que sub-symbolique, on peut dire que les agents basés sur ACT-R sont des agents hybrides.

1.3.3 IDA

IDA, de l'anglais « Intelligent Distribution Agent », est une architecture principalement développée par S. Franklin pour la Navy américaine (Franklin et al. 1998). Cette architecture a pour but de concevoir des agents logiciels dotés d'un certain niveau de conscience, tiré d'une théorie psychologique de la conscience appelée *atelier global* (Global Workspace Theory de Baars), leur permettant de s'adapter à des situations nouvelles. L'architecture IDA se base beaucoup sur les différentes mémoires que l'on retrouve dans la théorie de Baars : les mémoires de travail, les mémoires tampons, la mémoire sensorielle, la mémoire procédurale.

Dans l'architecture IDA, les cycles cognitifs s'enchaînent de manière ininterrompue afin de reproduire le processus cognitif humain. S. Franklin définit clairement la durée du cycle cognitif, soit 200ms. Pour chaque cycle, des parties de ces cycles peuvent être effectuées en parallèle, notamment au niveau des associations locales ou au niveau de l'apprentissage poussé par la conscience. La conscience maintient l'aspect séquentiel des décisions de l'agent, ainsi que l'ordre des règles à appliquer, afin d'assurer la cohérence des comportements de l'agent. Dans la figure 1.6 on voit la partie environnement dans IDA qui

peut être l'environnement extérieur dans lequel évolue l'agent, mais également l'environnement interne qui est en fait la perception que l'agent a de lui-même : sa conscience. Du coup, un cycle cognitif peut commencer sans perceptions venant de l'environnement extérieur.

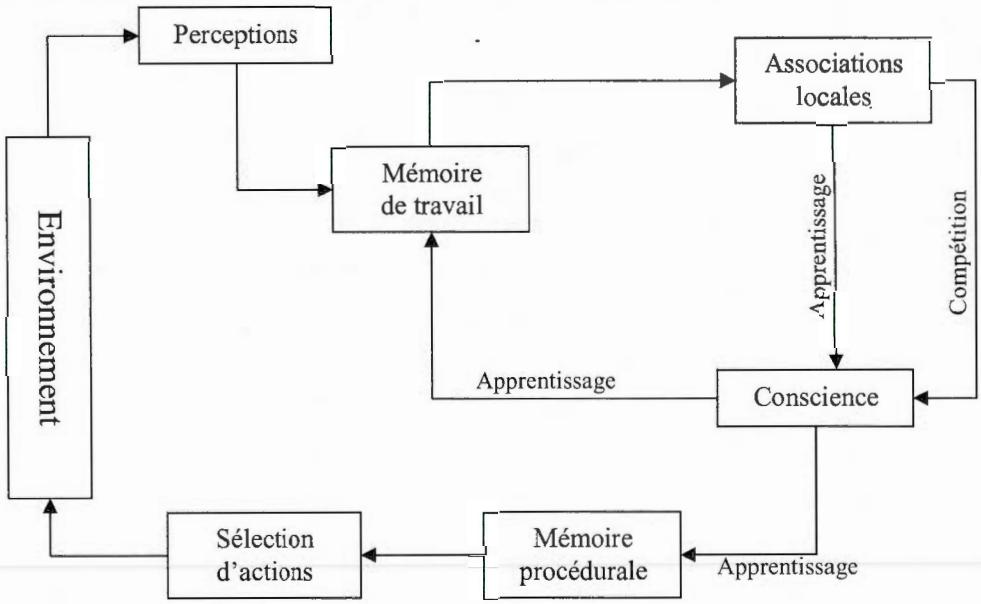


Figure 1.6 Architecture IDA simplifiée

L'architecture IDA a naturellement évolué vers l'architecture LIDA, pour Learning IDA, quand des procédés d'apprentissages ont été ajoutés à l'architecture. Ces procédés permettent notamment d'améliorer l'apprentissage épisodique, l'apprentissage à long terme et l'apprentissage procédural. Le cycle cognitif dans l'architecture LIDA peut être présenté en trois parties :

- La phase de compréhension, couvrant la perception, l'activation des mémoires et les associations créant une représentation symbolique chez l'agent.
- La phase d'attention, ou encore la conscience, manipulant les représentations symboliques et se focalisant sur les données sensorielles identifiées comme liées à ces représentation, puis activant un apprentissage sur les différentes mémoires.

- La phase d'action et d'apprentissage, qui conclut le cycle cognitif en sélectionnant le comportement le plus adapté et en effectuant l'apprentissage grâce aux actions sur l'environnement interne de l'agent.

LIDA se veut donc une architecture couvrant le plus possible la cognition humaine. Les différents mécanismes cognitifs dont s'inspire cette architecture définissent cette architecture, à l'instar d'ACT-R, comme hybride.

1.3.4 CTS

CTS, de l'anglais « Conscious Tutoring System », est une architecture pour un système tutoriel disposant d'une conscience[6] dite d'accès. D. Dubois, R. Nkambou et al. ont élaboré cette architecture à partir du modèle IDA, et plus particulièrement LIDA (Dubois et al. 2010). L'architecture CTS dispose donc du même cycle cognitif que l'architecture IDA et est basée sur la même théorie de Baars (Baars 1997).

La principale différence entre CTS et IDA est que CTS a été conçu pour être un agent tutoriel, un agent spécialisé dans la formation d'apprenant. Pour cela, la dimension émotionnelle est utilisée, et cette notion est fortement liée à la cognition et donc à la conscience d'accès dans cette architecture. Dans un premier temps, nous avons donc le cycle cognitif de CTS, et dans un deuxième temps nous verrons le réseau des actes de CTS, réseau qui complète le cycle cognitif.

Le cycle cognitif présenté dans la figure 1.7 est volontairement simplifié afin d'en permettre une meilleure approche, et comporte huit étapes qui ont été numérotées selon les routes décisionnelles. Ce cycle cognitif ordonnance les acteurs qui interviendront dans le réseau des actes pour la prise de décisions. Les stimuli sont perçus par les mécanismes (1) de perception. Les percepts deviennent (2) une partie de la mémoire de travail tandis que la mémoire déclarative (3) récupère des informations correspondantes. S'engage alors une compétition

---

[6] Ici on entend que la conscience d'un agent est un mécanisme qui lui permet d'avoir une compréhension de son existence, d'avoir une représentation de lui-même.

(4) pour l'accès à la conscience au sein de la mémoire de travail, puis la conscience d'accès diffuse (5) la plus forte. Ensuite les ressources comportementales (comme les codelets, voir la section 1.4.5. CTS) sont appelés, puis sélectionnés (7) afin d'en exécuter (8) les actes définis.

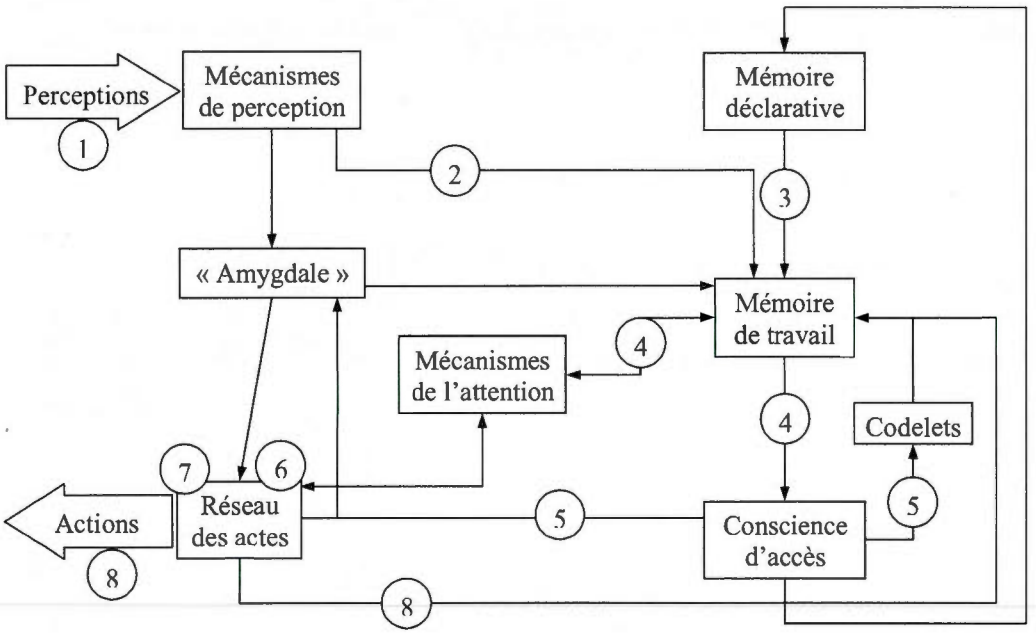


Figure 1.7 Cycle cognitif simplifié de CTS

Les nœuds que l'on peut voir dans la figure 1.8 ci-après sont en fait des actes tutoriels. Dans ce réseau des actes est géré un aspect complémentaire au cycle cognitif : l'émotion, ou plutôt la représentation émotionnelle que l'agent a à un instant donné. Toujours sur la figure 1.8, les flèches en pointillé ne signifient pas une information au sens classique, mais signifient une « énergie ». C'est cette énergie qui, en circulant, représente l'influence des émotions sur le raisonnement, comme on le détaillera ci-après.

Cet aspect complémentaire qui concerne l'influence des émotions dans la cognition humaine est étudié en neurobiologie, et E. Rolls suggère qu'il y a au moins deux routes de la perception à l'action. La première route principale correspond à un réflexe, fortement lié aux

émotions (complexe amygdalien[7]) et qui n'est pas interprété par les autres zones du cerveau. La deuxième route principale correspond à l'analyse par les différentes zones corticales[8] de l'environnement extérieur puis l'évaluation épisodique par les souvenirs, ensuite l'évaluation émotionnelle par le complexe amygdalien, et enfin la réponse (Rolls 1999).

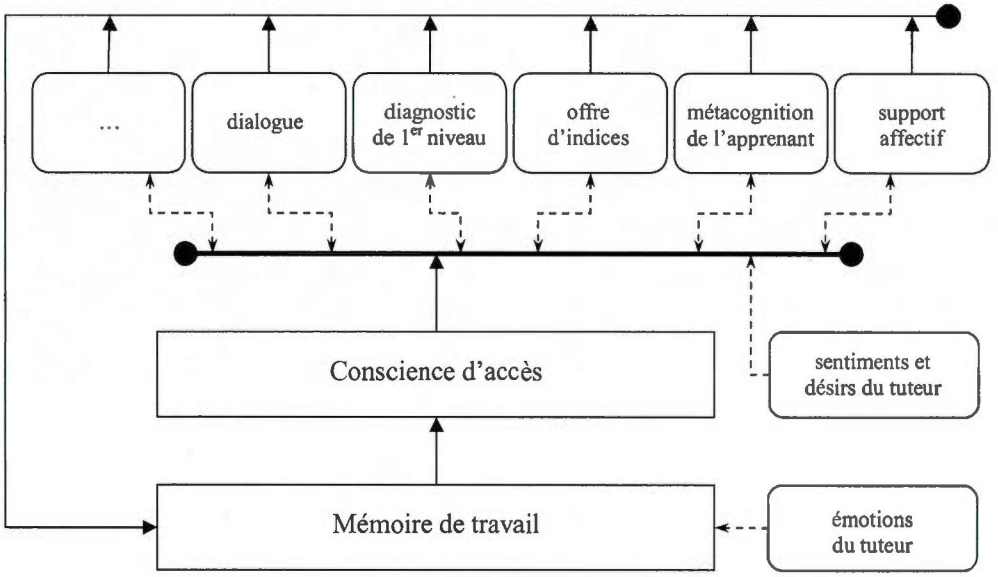


Figure 1.8 Partie du réseau des actes de l'architecture CTS

Dans l'architecture CTS, ces deux routes sont implémentées et fonctionnent en parallèle. Ce qui fait que dans CTS, les décisions conscientes de l'agent sont très fortement influencées par les émotions, et les émotions provoquent des actions réflexes. Les émotions ont un autre aspect au niveau de ce qui est appelé le renforcement émotionnel. En effet, lors de

---

[7] Zone du cerveau qui est une forme de système d'alarme, qui est impliquée dans l'évaluation émotionnelle stimulus extérieurs sensoriels.

[8] Zones du cerveau situées dans le cortex. Au sens cognitif du terme, elles sont impliquées dans les processus de raisonnement.

l'apprentissage, les actions sont influencées par les émotions et peuvent ainsi renforcer le chemin réflexe ou le chemin raisonné selon le contexte.

CTS, à l'instar de LIDA, est une architecture tendant à couvrir le plus possible la cognition humaine en comprenant également l'aspect émotionnel inhérent à la cognition humaine. De plus, avec son réseau des actes incorporé, CTS est très orienté vers l'apprentissage tutoriel. C'est une architecture hybride même si le nom laisse penser qu'elle est juste cognitive.

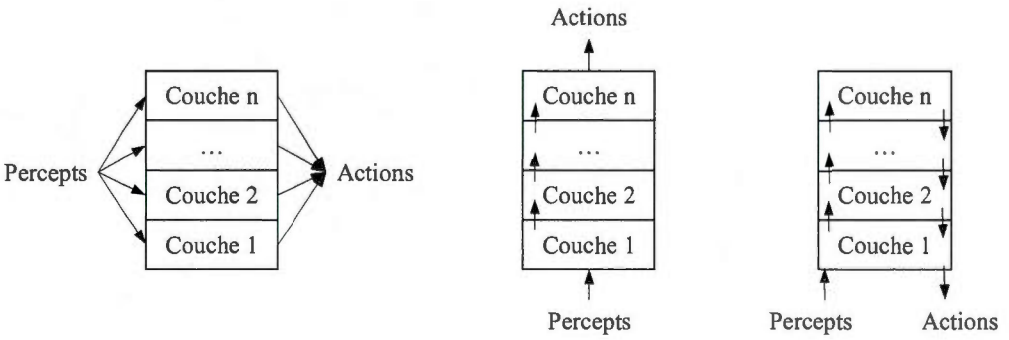
1.3.5 Modèle en couches

Les modèles en couches sont très simples en termes de conception : les percepts arrivent et sont traités par différentes couches pour ensuite sortir sous forme d'actions. Généralement on distingue deux types de modèles en couches, décrit par G. Weiss, dont la conception est illustrée par la figure 1.9 (Weiss 1999) :

- Le modèle horizontal dispose d'un nombre *n* de couches connectées directement aux capteurs ainsi qu'aux effecteurs, pouvant ainsi directement agir sur les actions. Chaque couche est en concurrence directe avec les autres (figure 1.9.a).
- Le modèle vertical dispose d'un nombre *n* de couches connectées entre elles à la manière d'une pile. Une seule couche au plus permet l'entrée des percepts ou l'émission d'actions. Ce modèle dispose de deux traitements différents de l'information selon que le circuit de l'information consiste en un seul passage (figure 1.9.b) ou en deux passages (figure 1.9.c).

Ces deux modèles permettent de gérer les flux d'informations selon le type d'agent désiré. L'avantage du modèle horizontal est que la conception est relativement simple, car on peut implémenter une palette de comportements dans une couche, et donc étendre les possibilités comportementales de l'agent juste en implémentant de nouvelles couches. Il faut quand même faire attention à ce que le comportement général de l'agent reste cohérent, du fait de l'aspect concurrentiel des multiples couches, ce qui nécessite l'implémentation d'un centre de contrôle pour gérer les actions effectives. Le modèle vertical permet d'éviter ce problème en partie car le nombre d'interactions dans les couches est restreint si on le compare au nombre d'interactions dans le modèle horizontal, que l'on utilise une passe ou deux passes. En revanche, le flux d'informations provenant des percepts passant par une succession de couches avant d'arriver à l'émission d'une action rend le modèle vertical intolérant aux

problèmes pouvant survenir dans l'une des couches, contrairement au modèle horizontale où le fait qu'une couche ait un problème ne coupe pas le flux d'informations. Nous allons donc voir deux architectures, l'une s'appuyant sur le modèle horizontal et l'autre sur le modèle vertical.



(a) couches horizontales (b) couches verticales (une passe) (c) couches verticales (deux passes)

Figure 1.9 Types des modèles en couches par G. Weiss

Le modèle en couches le plus célèbre est celui de I. Ferguson, appelé TouringMachine. L'architecture de TouringMachine, comme la figure 1.10 le montre, est relativement simple et est de type horizontal (Ferguson 1992). Elle est simple en partie par son âge mais principalement par l'intuition de I. A. Ferguson sur le lien entre l'intelligence et les comportements. Son intuition était que pour avoir un comportement intelligent, un agent n'a pas besoin d'être complexe, mais cette intelligence va se révéler dans son comportement dans un environnement avec d'autres agents.

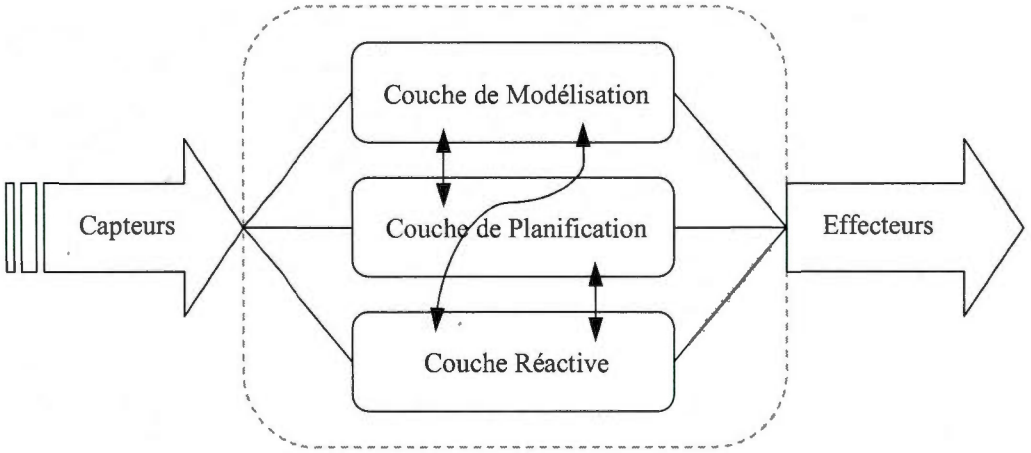


Figure 1.10 Architecture de la TouringMachine par I. A. Ferguson

L'objectif d'une architecture en couches est d'être capable de créer des agents robustes et souples à la fois. Robuste dans le sens où un agent doit être efficace et ne pas se bloquer, et souple dans le sens où un agent doit être capable de s'adapter aux situations. En effet, les agents sont censés produire un ensemble de comportements dans un environnement dynamique et partiellement observable : être capable d'agir selon le temps et les ressources disponibles, être capable de remplir des objectifs dans un temps raisonnable, être capable de s'adapter aux modifications de l'environnement provoquées par d'autres agents afin de toujours satisfaire ses buts.

De multiples modèles de cette architecture existent, et disposent d'un nombre différent de couches qui ont des fonctions ou objectifs différents. Les différentes couches peuvent s'échanger des informations, et il est important de noter que leur exécution se fait de manière concurrentielle (en parallèle). Les différentes couches perçoivent directement les stimuli et sont capables d'émettre directement des actions, vu qu'elles sont indépendantes. C'est pourquoi elles sont contenues dans un ensemble, un framework[9] par exemple, qui sera chargé

[9] Un framework est un terme anglais pour désigner un kit composé d'outils logiciels organisés pour faciliter le développement selon un modèle ou une architecture pour lequel il est conçu.

d'éviter les conflits dans les objectifs. Ici, la TouringMachine dispose de trois couches dont voici les propriétés :

- La couche réactive (ou « Reactive Layer » en anglais) correspond à l'arc réflexe (Figure 1.2) et s'apparente à un comportement réflexe. Cette couche contient un ensemble de règles de situation-action permettant un comportement réflexe. Par exemple, un agent qui se déplace et doit éviter soudainement un obstacle qui n'existait pas quelques instants avant. Bien entendu on parle d'une couche qui génère un comportement réflexe, la rationalité des actions n'est donc pas garantie. Pour corriger cela : à chaque action de la couche réactive, on interpelle la couche de modélisation qui pourra prendre en compte les actions effectuées par la couche réactive et alors adapter son raisonnement afin de satisfaire les objectifs de l'agent.
- La couche de planification (ou « Planning Layer » en anglais) est chargée d'ordonnancer les tâches et de gérer leur exécution. Les plans d'exécution générés par cette couche sont hiérarchisés à la manière d'un arbre à états, ce qui permet de s'adapter très rapidement à un changement soudain de l'environnement, et donc de re-planifier, car les ressources computationnelles[10] sont limitées dans un agent.
- La couche de modélisation (ou « Modelling Layer » en anglais) est chargée de permettre la réflexion et la prédiction chez un agent. Cette permet la construction de représentations du monde visible à partir desquelles un agent peut raisonner sur les comportements à avoir. Grâce à cet aspect de prédiction, l'agent est capable d'ajuster ses comportements en cas de conflits à venir. Les ressources computationnelles sont pré-allouées et bornées afin de garantir une prise de décision dans un temps raisonnable, et donc pas de latence dans l'enchaînement des actions de l'agent.

TouringMachine permet donc de créer des agents adaptés à des environnements dynamiques ainsi que multi-agents. TouringMachine permet aussi de produire, de manière délibérée ou

---

[10] Cela correspond aux ressources logicielles et matérielles auxquelles l'agent a accès par son implémentation et qui lui permettent de calculer. Ces ressources sont généralement limitées du fait du caractère embarqué d'un agent.

réflexe, des comportements orientés vers des objectifs définis. Cette architecture est capable de produire des comportements réflexes mais aussi téléonomiques, et donc sert pour créer des agents réactifs.

Un autre modèle connu, de type vertical à 2 passes, est InteRRaP[11] conçu par J. Müller. InteRRaP dispose de trois couches comme on peut le voir sur la figure 1.11, à l'instar de TouringMachine (Müller et al. 1993). L'intérêt des modèles en couche est la décomposition des fonctionnalités, ou plutôt des comportements, sous forme de couches. Pour InteRRaP, cela permet de prendre des libertés pour chaque couche.

Chaque couche contient une partie deux passes qui est une unité de contrôle, ainsi qu'une partie une passe qui est une base de connaissances. Ces trois couches sont connectées entre elles à la manière d'une pile et ont les spécificités suivantes :

- La couche comportementale est la couche connectée à l'interface de l'agent, c'est-à-dire son corps. Comme son nom l'indique, elle gère les comportements qui ont été effectués ou ceux à effectuer. Elle dispose d'une base de connaissance, une représentation du monde.
- La couche de planification est celle qui gère le raisonnement, la création de comportements ainsi que les objectifs de l'agent. Elle dispose d'une représentation mentale, des croyances que l'agent a sur lui-même.
- La couche sociale est la couche d'abstraction de plus haut niveau dans cette architecture. Elle permet le raisonnement concernant d'autres agents dans l'environnement. Elle permet également la communication avec d'autres agents dans l'environnement. Elle dispose d'une base de connaissance contenant des modèles d'interactions possibles, des « langages » permettant l'échange d'information, ou toute autre forme de connaissance sociale.

---

[11] L'architecture InteRRaP est hybride car elle comprend également une partie BDI, mais ce qui nous intéresse ici est la conception en couches verticales.

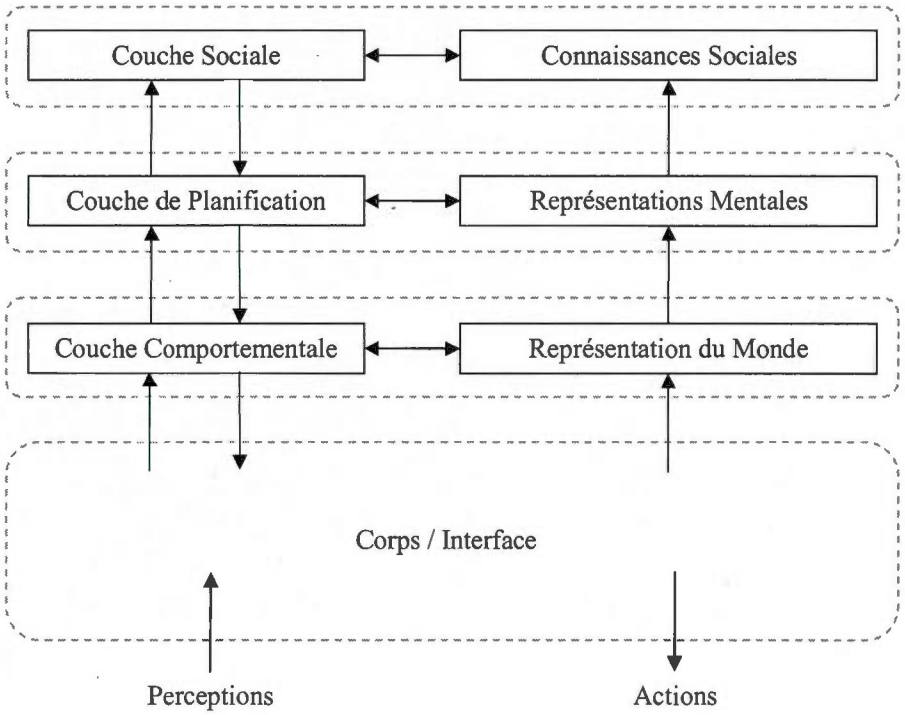


Figure 1.11 Architecture simplifiée d'InteRRaP

La différence avec le modèle horizontal est ici très claire, comme avec TouringMachine, en ce qui concerne les interactions entre les couches qui communiquent entre elles afin de donner un seul comportement à la fois, ce qui permet donc d'éviter les conflits internes comme avec le modèle horizontal où un contrôleur doit être ajouté.

Pour G. Weiss, cette approche de type vertical est très pragmatique, mais ne permet pas de créer de nouveaux comportements, qui doivent nécessairement être implémentés dans une couche, nouvelle ou existante (Weiss 1999). InteRRaP est donc une architecture hybride car elle réunit les composantes d'un agent tropique ainsi que d'un agent intentionnel (partie BDI incluse dans les couches).

1.4 Applications d'agents

Pour qu'un agent existe, il lui faut tout d'abord un corps. Ensuite, un agent ne peut exister que dans un environnement, sinon ce n'est qu'un modèle, il lui faut donc également un environnement (artificiel ou réel).

Nous allons tout d'abord voir des exemples d'agents dont les applications sont faites de nos jours, comme un thermostat, un service web et un robot aspirateur. Puis nous allons aborder quelques technologies d'implémentation, comme la librairie ACT-R et CTS, dont la plus grande partie sera abordée à la section 2.5 Implémentations et Applications du fait de l'aspect multi-agents de ces technologies. Enfin nous allons présenter une expérimentation d'une TouringMachine avec TouringWorld.

1.4.1 Agent thermostat

Un exemple d'agent simple est donné par M. Wooldridge, il s'agit d'un agent tropique, disposant donc d'un comportement réflexe, que tout le monde a déjà rencontré : un thermostat (Wooldridge 1999). Le thermostat est un système de contrôle permettant de gérer la température d'une pièce. Il dispose d'un capteur thermique dans l'environnement réel (une pièce) ainsi que d'un système décisionnel comprenant deux règles :

température inférieure à → allumage du chauffage

température supérieure ou égale à → extinction du chauffage

Le thermostat illustre très bien la catégorie des agents réflexes, et plus particulièrement les agents tropiques. L'agent n'a aucun objectif en soit, et ne peut donc offrir de garantie quant à la température de la pièce, mais il est conçu pour réagir à l'état de son environnement sur l'aspect de la température. Bien entendu, la complexité du système décisionnel et le nombre de règles dans celui-ci croît en fonction de la complexité des systèmes de contrôles sous forme d'agent que l'on veut concevoir.

1.4.2 Service web

Le cas des services web est également très courant car ils sont utilisés tous les jours. Ces agents sont purement logiciel, dans un environnement informatique. Ce sont des agents modules, car disposant d'un cycle cognitif qui leur est propre ainsi que d'un comportement

de type réflexe. En effet, un service web n'agit que sûr demande car il n'a pas d'objectifs à proprement parlé. Il sert de ressource dans le monde de l'Internet car il est sollicité par d'autres agents de son environnement, ici du système. Ces agents sont aussi appelés des oracles, du fait de leur rôle de ressource.

1.4.3 Robot aspirateur

Les robots aspirateurs sont un exemple d'application domestique pour le grand public d'agents autonomes destinés à remplir un objectif précis : faire le ménage en passant l'aspirateur tout seul. Pour ce faire, quoi de mieux que d'avoir un aspirateur autonome, c'est-à-dire qui est capable de se déplacer dans son environnement (les pièces d'une maison), de se nourrir (recharger ses accumulateurs), et de savoir quand il faut nettoyer ou non et à quel endroit. Bien des aspects de ce qu'on veut comme robot aspirateur désignent un type d'agent cognitif doté d'un comportement téléonomique : c'est un agent intentionnel. En effet, il doit évoluer dans un environnement dynamique (présence d'obstacles, objets déplacés) et d'avoir une représentation de son environnement, il a un objectif primaire et est capable de l'atteindre par l'intermédiaire d'autres objectifs, il doit savoir planifier ses actions afin de remplir ses objectifs ainsi que de se maintenir en activité, il doit être capable dans certains cas de communiquer ou coopérer avec d'autres agents (plusieurs robots dans une même maison), et il doit être capable de communiquer avec son propriétaire (lors d'un besoin de maintenance par exemple).

1.4.4 Librairie ACT-R

Parallèlement à l'élaboration de sa théorie et de son architecture ACT-R, J. R. Anderson a implémenté son architecture dans une librairie du même nom, qui comprend également la théorie (ACT-R : en ligne). Cette librairie permet donc l'implémentation d'agents basés sur l'architecture ACT-R. Elle est écrite en LISP[12], ce qui implique qu'un

---

[12] LISP (de l'anglais LISt Processing) est un langage de programmation de type impératif et de type fonctionnel, et est couramment utilisé dans les domaines liés à l'intelligence artificielle.

environnement LISP doit être installé pour pouvoir utiliser la librairie, et les modèles que la librairie utilise sont également écrits en LISP.

Le manuel de référence de la librairie ACT-R rédigé par D. Bothell précise la manière dont elle est conçue, et ce que l'on peut faire avec (Bothell : en ligne). Il y est présenté la manière de créer des hypothèses ainsi que des modèles et représentations qui incorporent le principe cognitif de la théorie ACT-R. Selon ce manuel, la librairie est principalement utilisée pour créer des modèles touchant à la mémoire et à l'apprentissage, la prise de décision, la résolution de problèmes, le langage et la communication, les processus d'attention et de perception, ainsi que le développement cognitif et les différences entre individus.

Cette librairie est également utilisée par PACT Center[13] de l'université Carnegie Mellon afin de développer des tuteurs cognitifs, qui font partie de la famille des systèmes tutoriels intelligents. Un des projets est CTAT[14], et celui-ci est un ensemble d'outils permettant de faciliter la création d'un système tutoriel intelligent en JAVA ou Flash par l'ajout d'un apprentissage par l'expérience, ce qui permet entre autre de créer rapidement une base de connaissances à partir d'ensembles de données ou d'expériences reproductibles plutôt que de concevoir à l'avance une base de connaissances ou de règles qui ne sera pas forcément pertinente dans le cadre du système tutoriel.

Il existe également jACT-R, qui est un projet dit « code source libre » (ou « open source » en anglais) implémentant l'architecture ACT-R (jACT-R : en ligne). Le projet est entièrement codé en JAVA, ce qui le rend portable très simplement et permet la création ou l'intégration de modules complémentaire codés eux-aussi en JAVA.

---

[13] Pittsburgh Advanced Cognitive Tutor Center, Carnegie Mellon University : http://pact.cs.cmu.edu/.

[14] Cognitive Tutor Authoring Tools : http://ctat.pact.cs.cmu.edu/.

1.4.5 CTS

La transmission de connaissances humaines par des professeurs nécessite des capacités humaines et beaucoup de projets en intelligence artificielle tentant de reproduire les mécanismes de décisions humaines ont été abandonnés à cause du coût computationnel et de la complexité. Une idée a été de découper le fonctionnement de la prise de décision en modules distincts et interagissant entre eux, comme des groupes de neurones biologiques peuvent se distinguer entre eux sous forme d'amas communiquant avec d'autres. Cette idée de penser une architecture avec ce principe permet d'approcher le fonctionnement de l'apprentissage humain et donc de reproduire un modèle de ce qu'est un apprenant et ce, afin de mettre en place un apprentissage efficace et adapté à l'apprenant.

En plus d'être une architecture cognitive hybride complètement orientée pour la création de systèmes tutoriels intelligents, CTS est également un agent tutoriel totalement implémenté et fonctionnel dit : agent cognitif « conscient ». Il a en premier lieu été développé en collaboration avec l'agence spatiale canadienne, sous le nom de CanadarmTutor, afin d'encadrer l'apprentissage de Canadarm2[15] des futurs astronautes (Nkambou et al. 2005). La problématique est qu'à la base il n'y a pas d'expert en manipulation du Canadarm2, que l'environnement réel est extrêmement sensible, et que l'interface du bras est très limitée : trois points de vue simultanés sur les douze possibles, sélection du joint à manipuler mais un seul à la fois. Il y a donc un besoin réel pour les astronautes de se former à l'aide d'un simulateur et d'un système permettant un tutorat efficace.

L'aspect émotionnel est une composante de CTS, notamment parce que les émotions jouent un rôle fondamental au niveau de l'apprentissage, mais également au niveau de la prise de décision. CTS est capable de se construire une représentation émotionnelle de apprenant, lui permettant d'orienter l'apprentissage en adéquation avec l'état émotionnel de l'apprenant. Par

---

[15] Canadarm2 est un bras robotisé en activité sur la station spatiale internationale qui permet la manipulation de modules ou de charges à l'extérieur de la station à partir d'une interface de contrôle à l'intérieur de la station.

exemple, la dimension émotionnelle de CTS permet la réactivité face au danger (dans la simulation) ce qui donne une empreinte intéressante dans l'apprentissage.

CTS peut être étendu par la conception de modules, appelés codelets, représentant des modules psychologiques jouant un rôle dans la prise de décision de l'agent. CTS, développé par D. Dubois, R. Nkambou et al., est l'agent cognitif hybride implémentant totalement l'architecture du même nom (Dubois et al. 2010).

1.4.6 TouringWorld

TouringWorld est un banc de test[16] écrit en Prolog[17] conçu par I. A. Ferguson et al. pour éprouver et mesurer les performances et les comportements des agents de type TouringMachines dans un environnement simulant le trafic urbain (Ferguson 1992). Cet environnement est dynamique, continu (car en temps-réel) et potentiellement multi-agents, et donc rempli de multiples entités ou objets. Bien entendu, pour chaque agent l'environnement n'est que partiellement observable. Avec TouringWorld, l'étude des comportements se fait de manière empirique car s'il est possible d'étudier les comportements d'agents dans un environnement très limités, il est très difficile de le faire pour un environnement entièrement dynamique. Cela permet de faire ressortir les tendances comportement des agents et de comprendre comment les comportements apparaissent chez les agents TouringMachines par rapport à différents scénarios ou contextes.

---

[16] Un banc de test est un système construit sur mesure pour éprouver un produit ou une technologie afin d'en mesurer les performances, les éventuels défauts, la fiabilité, le respect des spécifications, etc. Ici, le banc de test est un banc d'essai car permettant la mesure des performances en vue d'ajuster les caractéristiques des agents.

[17] Prolog est un célèbre langage informatique de programmation logique (d'où son nom : PROgramation en LOGique) créé par Alain Colmerauer et Philippe Roussel. C'est un langage très utilisé dans le domaine de l'intelligence artificielle, mais également en linguistique pour le traitement des langages naturels.

1.5 Limites et difficultés

Bien entendu, il n'est pas facile de choisir un type d'agent selon les besoins, et chaque type d'agent possède des limites. Généralement on rencontre trois types de difficultés ou limites : le coût en ressources, le choix de l'approche la plus intéressante pour un besoin, et le choix de la théorie à utiliser.

1.5.1 Coût

Le coût est une limite primordiale qu'il faut toujours prendre en compte à différents instants : lors de l'émission du besoin, lors de la conception, ou lors du développement en lui-même. Ce coût se mesure en ressources, à l'instar du coût en informatique, c'est-à-dire en capacité de mémoire, en puissance de calcul, en vitesse de transmission d'informations. Ce coût représente également les besoins en ressources computationnelles.

De nombreux paramètres ont une influence sur le coût d'un agent et ces paramètres dépendent beaucoup de ce que l'on veut que l'agent soit capable de faire. Le principal facteur de coût est le cycle cognitif choisi ou nécessaire. En effet, le cycle cognitif d'un agent est d'une complexité très variable, selon les besoins définis lors de la conception. Par exemple, si un agent doit avoir une réactivité en temps réel ou non, si un agent doit avoir un comportement réflexe ou téléonomique. Toujours selon le type de cycle cognitif choisi, le nombre de capteurs peut également avoir une grande influence sur la capacité des mémoires de travail ou des bases de connaissances.

Au-delà du coût, cela nous amène aux limites que les ressources matérielles (permettant les ressources computationnelles) peuvent offrir. Par exemple, jusqu'à quel point est-il possible d'accélérer le temps de traitement d'un agent au cycle cognitif long pour qu'il puisse avoir une réactivité la plus proche possible du temps réel ?

Quand il faut penser les agents par rapport aux besoins et aux ressources disponibles, il est donc important d'étudier avec soin la complexité des cycles cognitifs, la rapidité des échanges, ainsi que les temps de réaction, pour les ajuster avec soin dans la mesure du possible.

1.5.2 Approche *micro/macro*

L'approche que j'appelle *micro/macro* se réfère à l'échelle que l'on peut utiliser pour identifier l'architecture la plus en adéquation avec l'environnement dans lequel vont évoluer les agents. Le côté *micro* se focalise sur un agent particulier dont l'architecture est relativement complète, ce qui en fait un agent dédié à la résolution en solo d'un problème spécifique. Le côté *macro* se focalise sur une population d'agent dont l'architecture est simple, ce qui permet à cette population d'agents de résoudre un problème comme une seule entité le ferait. Ce côté macro sera vu plus en détail dans le chapitre suivant car cela porte sur les interactions entre les agents d'une même population.

L'idée ici est de savoir, selon les besoins ou les problèmes à résoudre, si un agent (ou un nombre restreint d'agents) complexe avec un cycle cognitif évolué est préférable à un ensemble d'agents plus simple.

1.5.3 Pour quelle théorie ?

Du fait du grand nombre de théories sur la cognition, dont certaines sont citées dans ce mémoire, choisir quel type d'agent utiliser devient vite ardu. En effet, pour construire les agents nécessaires à un problème ou une situation, quel type d'agent sélectionner et pour quelle théorie de la cognition serait-il conforme ? Dans un premier temps, il convient donc de bien définir les capacités pour le ou les types d'agents désirés, en se basant sur les possibilités qu'offre l'une des théories de la cognition existantes.

Dans un second temps, une difficulté vient s'ajouter : celle de créer des agents conformes aux théories de la cognition. Cette difficulté survient lors de la conception même des agents, par exemple à l'étape de la programmation, car selon les caractéristiques de tel ou tel langage de programmation, la réalisation d'algorithmes ou modules conformes à la théorie de la cognition choisie peut être difficile à mettre en œuvre ou peut même imposer des limitations en termes de coût des agents.

CHAPITRE II

SYSTEMES MULTI-AGENTS : DEFINITIONS, ORGANISATIONS, COMMUNICATION, COORDINATION & APPLICATIONS

Après avoir vu dans le chapitre précédent ce qu'est un agent et ce qui s'y rapporte, nous pouvons aborder les ensembles d'agents en tant que système complet : les systèmes multi-agents. Dans un premier temps, en section 2.1, nous allons voir les concepts liés aux systèmes multi-agents, systèmes qui sont une extension de l'utilisation des agents. Ensuite, dans la section 2.2, nous allons nous intéresser aux différents types d'organisation des ces systèmes. Dans la section 2.3, nous allons nous intéresser à la communication chez les agents, qui est une composante fondamentale à tout système multi-agents. Puis, dans la section 2.4, nous allons voir l'aspect coordination qui vient compléter le type d'organisation ainsi que la communication. Enfin, dans la section 2.5, nous allons explorer différentes implémentations de systèmes multi-agents[18], différents langages et plateformes de développement pour les concevoir, ainsi que quelques exemples de systèmes connus.

2.1 Définitions

Dans cette première section nous allons étendre les notions propres aux agents vues dans le chapitre précédent aux systèmes multi-agents. Nous allons donc voir les définitions d'un système multi-agents, notamment selon J. Ferber et M. Wooldridge. Ensuite nous allons définir la notion d'interaction, permettant à un ensemble d'agents d'être regroupés sur la

---

[18] Dans la littérature, l'acronyme SMA (ou MAS en anglais) est couramment utilisé pour désigner les systèmes multi-agents.

forme d'un système complet. Puis nous allons voir la notion d'adaptation qui ouvre, avec l'interaction, les portes de la communication, de l'organisation et de la coordination.

2.1.1 Système Multi-Agents

En utilisant la notion d'agent vue dans le chapitre précédent, nous allons tout d'abord présenter un système multi-agents comme plusieurs agents regroupés ensemble et liés entre eux de manière à satisfaire un ou plusieurs objectifs communs.

Par exemple, pour J. Ferber, un système multi-agents est la réalisation de « modèles électroniques ou informatiques composés d'entités artificielles qui communiquent entre elles et agissent dans un environnement » (Ferber 1995). Il précise que des agents capables de communiquer regroupés sous forme de communauté permettent de nombreuses interactions. De ces interactions apparaissent des structures organisées dans cette communauté d'agents. Ces structures définissent et influencent alors les comportements des agents, et du même coup les comportements du système multi-agents dans son ensemble. D'ailleurs, J. Ferber nous donne lui-même sa définition d'un système multi-agent : « On appelle système multi-agent (ou SMA) un système composés des éléments suivants : 1) Un environnement E, c'est-à-dire un espace disposant généralement d'une métrique ; 2) Un ensemble d'objets O. Ces objets sont situés, c'est-à-dire que, pour tout objet, il est possible, à un moment donné, d'associer une position E. Ces objets sont passifs, c'est-à-dire qu'ils peuvent être perçus, créés, détruits et modifiés par les agents ; 3) Un ensemble A d'agents, qui sont des objets particuliers ($A \subseteq O$), lesquels représentent les entités actives du système ; 4) Un ensemble de relations R qui unissent des objets (et donc des agents) entre eux ; 5) Un ensemble d'opérations Op permettant aux agents de A de percevoir, produire, consommer, transformer et manipuler des objets de O ; 6) Des opérateurs chargés de représenter l'application de ces opérations et la réaction du monde à cette tentative de modification, que l'on appellera les lois des l'univers[19]. »

---

[19] Ici, les lois de l'univers décrivent l'ensemble des règles physiques de l'environnement, à l'instar des lois physiques de notre monde réel.

En plus de cette définition, il met l'accent sur deux points fondamentaux qui permettent de dire qu'un système est un système multi-agents ou non. Le premier point est que le système soit composé de multiples agents autonomes ayant un but ou tout autre système donnant sa raison d'être à l'agent, et que tous les agents fonctionnent en parallèle. Le second point est que les agents disposent d'une composante permettant les interactions avec d'autres agents, comme un protocole de communication ou encore la manipulation de l'environnement.

M. Wooldridge introduit ce qu'est un système multi-agents avec un slogan populaire dans le monde des SMA : « There's no such thing as a single agent system », ce qui traduit nous donne : il n'y a rien de tel qu'un système à agent unique. En effet il introduit d'abord l'intérêt que l'on doit porter au concept d'interaction, concept que nous verrons dans la sous-section suivante en 2.1.2 Interactions. Il introduit ensuite l'idée que l'on doit penser un système multi-agent comme une société d'agents, notamment grâce aux interactions des agents entre eux. Il définit donc ces systèmes comme des sociétés d'agents dont les agents interagissent avec d'autres agents, par la communication, mais aussi par l'intermédiaire de *l'influence* que chaque agent a sur l'environnement, étant donné qu'un agent est capable d'agir, d'utiliser et de transformer son environnement. M. Wooldridge oriente donc sa conception des systèmes multi-agents sur l'interaction des agents entre eux, sur l'influence des uns sur les autres comme on peut le voir plus loin sur la figure 2.1, et plus précisément sur le type même des interactions, point que l'on détaillera également dans la sous-section suivante (Wooldridge 2002).

Le point de vue de J. Ferber sur la conception des systèmes multi-agents est fortement orienté sur les possibilités de comprendre et de pouvoir construire des théories sur l'émergence de constructions structurelles au sein des systèmes multi-agents. Par l'expérimentation, l'observation et l'analyse des résolutions de problèmes de compétition ou de coopération, des modèles mathématiques et informatiques théoriques peuvent ainsi apparaître car la conception des briques fondamentales d'un système multi-agents, à savoir les agents, est de conception humaine. Cela permet de contrôler et de délimiter les caractéristiques et paramètres de bases, et ainsi d'expérimenter les évolutions structurelles qui en découlent.

Il est important de dire que toutes les évolutions d'un système ne peuvent pas être prédites à partir d'une situation initiale. En effet, ces évolutions sont directement dépendantes des interactions et non simplement de l'architecture des agents même si celles-ci ont une influence sur le type d'interactions possibles. Dès lors, les systèmes multi-agents sont soumis aux phénomènes chaotiques tels que décrit la théorie du chaos de J. Gleick. Cela signifie que toute modification de condition ou de variable fait apparaître avec le temps des phénomènes que l'on appelle des effets papillons[20], c'est-à-dire que toute modification initiale provoque des perturbations qui ne sont pas prévues, et qui sont amplifiées avec le temps (Gleick 1989).

Le fait que les systèmes multi-agents soient soumis à l'effet papillon ne signifie pas qu'aucune règle ou aucune organisation ne le régissent. Au contraire, on observe un phénomène appelé auto-organisation[21]. C'est là un grand intérêt des systèmes multi-agents : des structures et organisations peuvent émerger des interactions sans être prédites ou sans être programmées, mais dont l'évolution peut être contrôlée. Pour garantir ces contrôles sur l'évolution d'un système multi-agents, il est très important de bien définir toutes les composantes des agents, que ce soit leur cerveau ou que ce soit leurs possibilités en terme d'interactions.

Un autre aspect des systèmes multi-agents concerne la répartition de l'intelligence, d'après J. Ferber. En effet, les solutions d'un problème peuvent être réparties dans l'environnement, ce qui rend une solution globale caduque. Il existe différentes manières de constater la distribution des problèmes : c'est le cas dans le trafic urbain où chaque véhicule à ses propres buts et sa vision locale ; c'est le cas aussi pour le développement d'un produit industriel où il

---

[20] L'effet papillon est une expression illustrant un phénomène fondamental dans la théorie du chaos, la sensibilité aux conditions initiales. Par exemple, un battement d'aile de papillon peut-il provoquer un ouragan de l'autre côté de la planète ?

[21] Le phénomène d'auto-organisation est une partie de ce que l'on appelle l'émergence. Le principe d'émergence est encore sujet à débat mais est très utilisé et observé dans le cadre de l'intelligence artificielle et des systèmes multi-agents.

est nécessaire de faire appel à divers spécialistes affectés à une partie précise du produit ; c'est le cas pour nombre de problèmes informatiques d'aujourd'hui notamment avec la distribution du calcul et la multiplication des réseaux. De ce fait, les solutions doivent être apportées du point de vue local, et l'intelligence doit être capable de se focaliser sur ces points précis. La répartition de l'intelligence selon ce principe permet d'obtenir une intelligence globale cohérente.

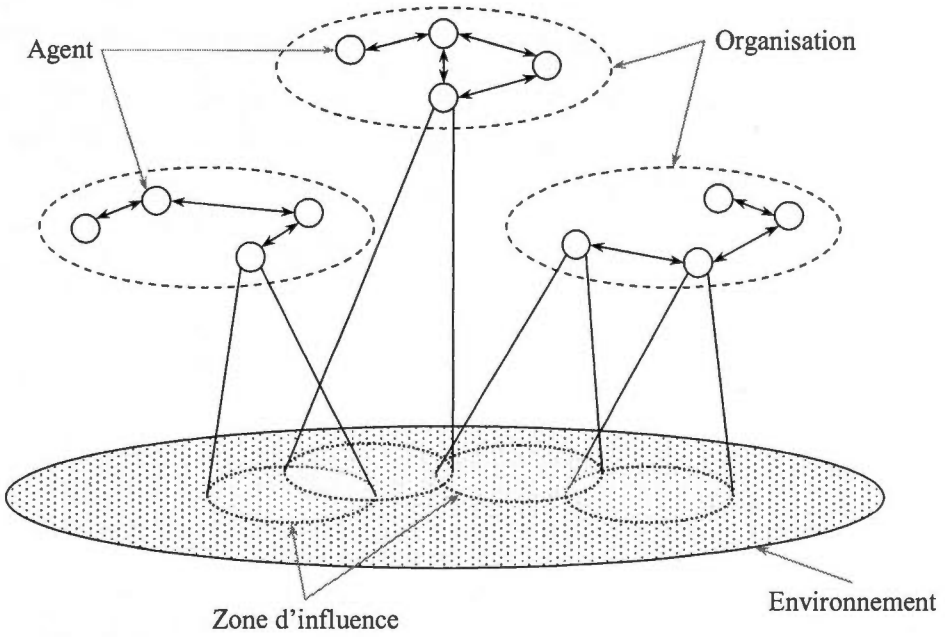


Figure 2.1 Structure type d'un système multi-agents (M. Wooldridge).

Pour les systèmes multi-agents, il existe un lien très fort ente les éléments de l'ensemble A (les agents) et l'environnement E, tels que décrits dans la définition de J. Ferber, notamment du fait que pour chaque agent, les autres agents appartiennent à l'environnement. Ce lien est en fait une dualité[22] car il n'est pas possible de définir ce qu'est un agent sans le définir par

[22] Le terme dualité représente ici le fait que la notion d'agent et la notion d'environnement représentent deux composantes d'un même concept au sein des SMA.

rapport à ce qu'est l'environnement, et il n'est pas possible non plus de définir l'environnement sans le définir par rapport aux agents qu'il contient. Ce qui fait que définir un système multi-agents nécessite de définir la structure de l'environnement ainsi que la structure même des agents. Sur ce point, l'environnement est défini comme un espace métrique car la coordination entre les agents nécessite un tel type d'espace. De là nous pouvons distinguer deux types d'agents différents concernant la dualité agent/environnement :

- Un agent purement communiquant, qui se différencie de la notion d'agent classique telle que vue dans le précédent chapitre du fait que ce type d'agents n'a pas de perception sur d'autres agents et n'est pas dans un environnement métrique. Ce type d'agent est clairement orienté vers les systèmes informatiques, et un agent purement communiquant peut donc se définir comme un agent logiciel.
- Un agent purement situé est de nature contraire à l'agent logiciel car il ne possède qu'une représentation locale de son environnement (et donc faible) et que sa manière de communiquer se fait par l'intermédiaire de l'environnement (avec un langage écrit ou vocal par exemple).

Notons que pour certains, le domaine des systèmes multi-agents et le domaine de l'intelligence artificielle ne sont que des domaines voisins. En effet, l'intelligence artificielle serait liée au fait qu'un individu est intelligent, alors qu'un système multi-agents est une société d'individus qui ne sont pas nécessairement intelligents. Mais en disant cela, il y a une question que l'on peut se poser : en quoi une entité, composée de multiples individus, ne peut-elle pas être considérée comme un individu à part entière ? J'estime pour ma part que ce point, à savoir si les systèmes multi-agents et les intelligences artificielles sont des domaines voisins, dépend du point de vue avec lequel on considère ce qu'est l'intelligence en général, et l'intelligence individuelle ainsi que l'intelligence dite distribuée en particulier.

Nous pouvons donc dire que de manière générale, un système multi-agents est une société composée d'individus appelés agents, que ces agents sont situés dans un environnement et interagissent avec celui-ci pour interagir avec les autres agents. Des phénomènes auto-organisateurs apparaissent au sein de ces systèmes, permettant l'adaptation d'un système multi-agent à son environnement et dépendamment des objectifs de ces systèmes.

2.1.2 Interactions

L'interaction est une notion fondamentale dans les systèmes multi-agents, et cette notion est la cause de bien des problèmes mais également de bien des solutions. Les agents disposent de mécanismes d'interaction pour leur permettre d'accomplir leurs objectifs. Nous allons donc commencer par définir cette notion, et nous détaillerons les différents types d'interactions dans la section 2.4 Coordination.

L'interaction se définit comme étant « (...) une mise en relation dynamique de deux ou plusieurs agents par le biais d'un ensemble d'actions réciproques. Les interactions s'expriment ainsi à partir d'une série d'actions dont les conséquences exercent en retour une *influence* sur le comportement futur des agents » (J. Ferber).

Pour qu'une interaction apparaisse, il faut qu'un évènement provoque ce que l'on appelle une situation d'interaction : « On appellera situation d'interaction un ensemble de comportements résultant du regroupement d'agents qui doivent agir pour satisfaire leurs objectifs en tenant compte des contraintes provenant des ressources plus ou moins limitées dont ils disposent et de leurs compétences individuelles » (J. Ferber). C'est-à-dire une situation où l'interaction est nécessaire pour que chaque agent puisse poursuivre ses buts, comme par exemple une collision ou un accès simultané à la même ressource, ou encore la mise en commun de connaissances (Ferber 1995). Donc on peut dire que pour qu'une interaction puisse exister, il y a besoin : d'agents pouvant agir et communiquer, de situations d'interactions, d'éléments dynamiques provoquant l'interaction (accès à la même ressource par exemple), d'une autonomie chez les agents leur laissant gérer les interactions « librement » selon leurs objectifs.

Un aspect des interactions est ce qu'on appelle l'*utilité*. En effet, M. Wooldridge met en avant ce qui pousse l'agent à interagir : quel intérêt un agent peut avoir à interagir avec d'autres. C'est un aspect important car il amène différentes solutions pour traiter certaines formes d'interactions spécifiques comme la résolution de conflits (Wooldridge 2002).

Notons qu'un agent ne disposant d'aucune interaction avec d'autres agents serait en quelque sorte isolé dans l'environnement, et ne deviendrait qu'un objet pour les autres agents car ne

permettant aucune faculté d'adaptation. C'est l'ensemble des interactions qui permettent, de manière émergente, l'apparition d'organisations qui sont des organisations sociales.

On distingue plusieurs types d'interactions qui sont étudiées dans le domaine. Nous traiterons plus en détail ces types d'interaction, qui sont des formes de coopération, dans la section 2.4 Coordination. Nous verrons ce qu'est la collaboration, l'indépendance, les stratégies de coordination ainsi que les conflits.

Un point important également, auquel nous ne manquerons pas de rappeler l'importance au cours du chapitre, est le phénomène d'émergence caractéristique des systèmes multi-agents, phénomène de construction structurelle dont le moteur principal est composé des actions ou des comportements ainsi que des interactions.

2.1.3 Adaptation

Lors de l'étude des systèmes multi-agents, l'adaptation est un aspect qui est considéré comme important. Cet aspect est particulièrement mis en avant par D. Lestel et il est très apparent lorsque l'on observe les comportements des agents développant une auto-organisation du fait de leurs multiples interactions (Lestel et al. 1994). L'adaptation est alors une conséquence des phénomènes auto-organisateurs dans un tel système.

Selon J. Ferber, l'adaptation peut se voir sous deux aspects : la forme structurelle et la forme comportementale. L'adaptation structurelle concerne des mécanismes collectifs, comme l'auto-organisation ou la reproduction, ce qui amène à considérer ce type d'adaptation comme une forme d'évolution. L'adaptation comportementale concerne l'agent en tant qu'individu et s'apparente à une accumulation d'expérience, ce qui revient à un apprentissage (Ferber 1995). Ces deux aspects ne sont bien sûr pas nécessairement indépendants, et peuvent chacun avoir leur impact sur l'adaptation globale d'un système multi-agents dans son environnement.

2.2 Types d'organisation

De nombreux types d'organisation peuvent apparaître dans les systèmes multi-agents, et on peut en avoir un aperçu lorsque l'on observe certains de ces systèmes que l'on trouve dans la nature, comme les ruches, fourmilières, termitières, certains bancs de poissons, et bien d'autres. Le point commun le plus visible de ces systèmes multi-agents naturels est que leurs agents sont simples mais que la société est complexe. On peut illustrer très succinctement certains de ces systèmes naturels :

- Une fourmilière est l'habitat d'une colonie de fourmis, mais on utilise également ce terme pour désigner une colonie en tant qu'entité à part entière, se développant dans son environnement afin de survivre et se reproduire. La fourmi est ce qu'on appelle un insecte social, c'est-à-dire que sa capacité à communiquer (grâce aux phéromones) est fondamentale. L'organisation au sein d'une fourmilière est telle que tous les travaux sont différenciés et que chaque fourmi a une attribution en fonction de sa caste. Un des problèmes complexes que doit résoudre une fourmilière est la recherche de nourriture, et les très nombreuses interactions des fourmis entre elles permettent la résolution de ce problème de manière locale, sans volonté de la reine.
- Une ruche, ou plus précisément un essaim, est une colonie d'abeilles dont l'organisation, bien que pouvant ressembler à celle d'une fourmilière, est un peu différente. Il existe bien évidemment un système de caste en fonction du sexe, mais deux points diffèrent des fourmis. Premièrement, les interactions entre abeilles sont différentes (il y a les phéromones mais également les danses) et deuxièmement, les rôles sont également déterminés par l'âge d'une abeille. Un problème intéressant est la recherche d'un emplacement intéressant pour la future colonie car chaque abeille dispose du même impact sur la colonie en termes d'interaction, ce qui de fait oblige les abeilles à un consensus pour s'établir.
- Un banc de poissons est un autre modèle organisationnel qui ne comprend aucune structure hiérarchique et où chaque individu est identique (dans le sens où il n'y a pas de rôle spécifique à un individu et où chaque individu peut prendre n'importe quel rôle possible dans le banc), c'est une agrégation d'individus. Dans ce type de structure il est intéressant de noter que le leader n'est que l'individu le plus en avant dans le banc et que chaque individu dispose d'un sens pour s'orienter en fonction des individus voisins. Cela

permet d'éviter la prédation, notamment avec la séparation du groupe lorsqu'un prédateur se jette sur le banc, et sa réunion après le danger.

Ces détails servent à illustrer la diversité des organisations possibles dans un système multi-agents.

Un autre genre d'exemple que l'on peut trouver dans la nature est la différence entre les systèmes multi-agents dont les agents sont des agents tropiques (comportement réflexe et agent de type réactif) et ceux dont les agents sont des agents intentionnels. Si nous comparons l'auto-organisation de l'un avec l'autre, nous pouvons bien sûr voir des différences structurelles dépendantes de l'architecture propre des agents. Par exemple, d'un côté il pourra y avoir un système multi-agents dont les agents sont des molécules, et donc purement réflexes et réactifs, avec des possibilités d'interactions limitées aux seules collisions ; et de l'autre un système multi-agent qui est une ruche ou une fourmilière, dont les agents ont des possibilités d'interactions plus larges, au niveau de la communication, transmission de savoir entre autres. Dans ce genre de modèle organisationnel, nous sommes amenés à tenter de comprendre comment les agents peuvent se regrouper et comment l'attribution de rôles ou la mise à disposition de services se mettent en place.

Suite à ces exemples, il nous faut définir clairement ce qu'est une organisation. Selon E. Morin : « Une organisation peut être définie comme un agencement de relations entre composants ou individus qui produit une unité, ou système, dotée de qualités inconnues au niveau des composants ou individus. L'organisation lie de façon interrelationnelle des éléments ou évènements ou individus divers qui dès lors deviennent les composants d'un tout. Elle assure solidarité et solidité relative, donc assure au système une certaine possibilité de durée en dépit de perturbations aléatoires » (Morin 1977). Nous retrouvons dans cette définition large l'aspect d'un regroupement d'agents interagissant entre eux afin d'assurer une pérennité de leurs objectifs, grâce à la mise en commun des connaissances et des compétences par exemple. Ce qu'ici E. Morin appelle interrelation est une interaction ordonnée, c'est-à-dire qu'au sein de l'organisation, toutes les interactions sont ordonnées, hiérarchisées ou catégorisées afin de maintenir une certaine cohérence organisationnelle. Nous retrouvons ici une dualité, c'est-à-dire que l'organisation est à la fois l'ensemble des mécanismes qui crée celle-ci, mais également le résultat de ces mêmes mécanismes. Et une

organisation étant dépendantes des interactions en son sein, elle ne peut qu'être dynamique dans le cadre des systèmes multi-agents.

Nous rappellerons toutefois que du fait du contrôle que l'on a sur l'architecture des agents, l'apparition et le développement de tel modèle organisationnel et de tel niveau d'organisation peuvent être orientés. Nous allons donc voir dans un premier temps quelques modèles organisationnels, et dans un second temps les niveaux d'organisation qui peuvent apparaître dans les systèmes multi-agents.

2.2.1 Modèles organisationnels

Nous avons vu qu'un système multi-agents est une forme de sociétés d'agents, et nous retrouvons donc principalement deux modèles que nous allons détailler ici, dont l'un est d'ailleurs l'extension de l'autre : le modèle AGR (Agents/Groupe/Rôle) et le modèle AGRS (Agents/Groupe/Rôle/Service).

2.2.1.1 AGR (Agents/Groupe/Rôle)

Le modèle AGR est couramment vu car le plus « naturel » au sens où il permet même d'identifier les modèles organisationnels de nos sociétés humaines à ce type de modèle.

Tout d'abord, ce modèle se découpe en trois pôles :

- Agent : c'est un individu actif qui peut jouer un ou plusieurs rôles dans un ou plusieurs groupes.
- Groupe : c'est un regroupement d'agents ayant des caractéristiques communes et/ou partageant un même ensemble d'activités. Les agents ne peuvent communiquer entre eux qu'au sein de ce groupe (précision : les agents peuvent tout de même interagir avec d'autres agents d'autres groupes).
- Rôle : c'est une fonction limitée au groupe que doit assurer un ou plusieurs agents au sein du même groupe, et tout agent de ce groupe peut s'attribuer ce rôle.

Nous avons donc un modèle de base pour l'organisation d'une société d'agents, comme la figure 2.2 Schéma du modèle AGR ci-après le montre.

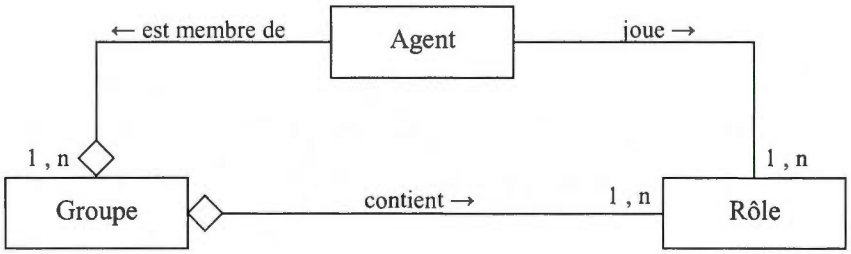


Figure 2.2 Schéma simplifié du modèle AGR (J. Ferber).

2.2.1.2 AGRS (Agents/Groupe/Rôle/Service)

Le modèle AGRS est un modèle visant à étendre le modèle AGR pour répondre à des problèmes d'ouvertures des systèmes multi-agents classiques. En effet, J. Ferber et S. Mansour notent que souvent le développement de systèmes multi-agents utilise des approches soit centrées sur les agents, soit centrées sur le côté social des agents, ce qui tend au développement d'un système fermé et adapté pour une seule application spécifique (Ferber et Mansour 2007).

Ici, la notion de service y est développée pour permettre une clarification au sein d'une organisation : qui peut faire quoi et comment. Dans le modèle AGRS, le service reste abstrait et regroupe l'ensemble des fonctionnalités qu'un agent peut demander ou fournir. Pour ce faire, le schéma d'AGR de la figure 2.2 va être étendu avec une classe « acteur » qui représente un agent qui est en train de jouer un rôle.

Le rôle décrit les services disponibles à travers les compétences de chaque agent, et l'agent voulant utiliser un service s'adresse au rôle qui le met en relation avec un acteur. Un acteur est un agent qui joue un rôle. Chaque agent peut s'inscrire au niveau du rôle, définissant les services et compétences qu'il peut mettre à disposition du groupe, et ce même rôle fera appel à un agent inscrit lorsqu'un autre agent demandera l'accès à un service.

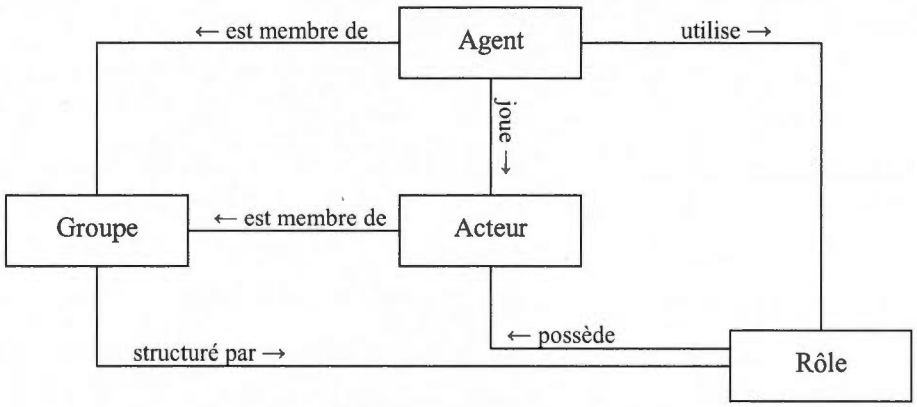


Figure 2.3 Schéma simplifié du modèle AGRS (J. Ferber et S. Mansour).

Ce modèle d'organisation est plus souple et plus ouvert car il permet l'ajout de multiples agents dans les groupes sans affecter l'organisation mise en place. Ce modèle permet une organisation totalement dynamique et permet de réduire la complexité de conception d'un système multi-agents.

2.2.2 Niveaux d'organisation

En s'inspirant de la classification sociologique de G. Gurvitch, J. Ferber distingue dans les systèmes multi-agents trois niveaux d'organisation différents qui peuvent servir de repère dans les systèmes multi-agents (Gurvitch 1963, Ferber 1995) :

- Le niveau *micro-social* : beaucoup d'études ont été faites au niveau micro-social car à ce niveau on s'intéresse particulièrement aux interactions entre des agents et à leurs types.
- Le niveau des *groupes* : ce niveau est porté sur les sous-structures, ou les structures intermédiaires entre le niveau micro-social et le niveau des sociétés globales. C'est à ce niveau que commencent les phénomènes auto-organisateurs, au niveau de petits groupes d'agents, et que l'on peut observer l'attribution de rôles et la spécialisation sur certains types d'activités pour chaque agent. Cela permet d'étudier plus finement la structure interne d'une structure globale.
- Le niveau des *sociétés globales* (aussi appelé populations, qui est un niveau à l'échelle macroscopique) : beaucoup d'études concernant la vie artificielle et l'évolution de populations se font à ce niveau car on s'intéresse à l'évolution d'un système dans sa

structure globale, qui comprend un nombre très important d'agents, générant ainsi énormément d'interactions et donc de phénomènes auto-organisateurs.

Il devient intéressant d'utiliser ces niveaux d'organisation ou l'un d'entre eux et de se baser sur leurs tendances pour concevoir un système multi-agents avec les propriétés recherchées. Selon une conception basée sur les interactions entre agents ou sur une structure globale de société, le résultat et les applications d'un système multi-agents peuvent être bien différents. On remarque notamment qu'il y a une dualité agent/organisation, comme le monde la figure 2.4, du fait des interactions qui font que les agents permettent l'existence des organisations et que les organisations permettent aux agents d'interagir selon leurs objectifs.

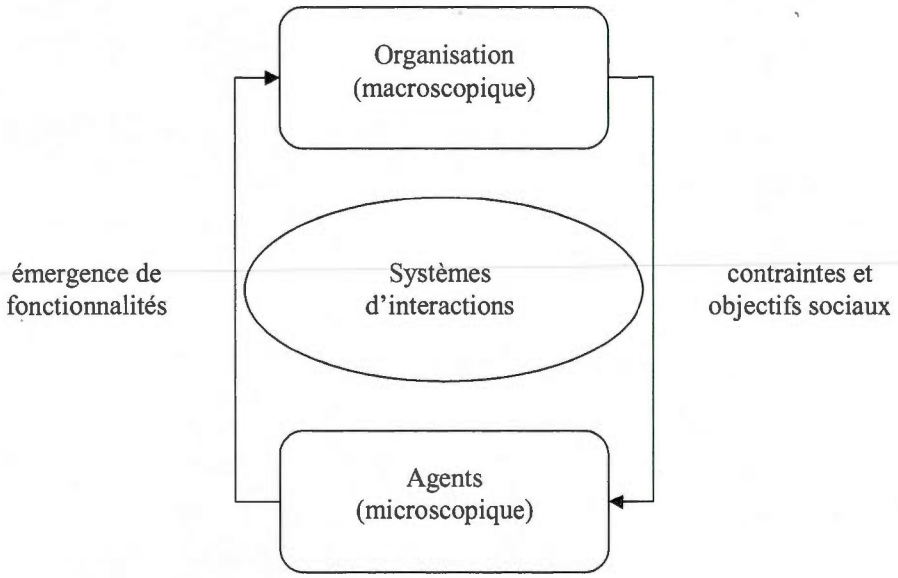


Figure 2.4 Relation microscopique/macroscopique dans les SMA (J. Ferber).

Pour en revenir aux sociétés de type fourmilière ou ruche que nous avons vues dans les sections précédentes, on peut voir grâce aux différents niveaux d'organisation que même si ces sociétés sont dites très complexes, on peut comprendre les phénomènes d'adaptation et d'auto-organisation sans avoir besoin de considérer cette société comme une entité à part entière (même s'il reste possible de les considérer comme des entités pleines). De ce point de vue, il est aisé d'appréhender le phénomène d'émergence d'une organisation grâce aux

multiples interactions entres agents, interactions qui concernent uniquement les individus entre eux.

2.3 La communication et ses enjeux

La communication est un aspect important, même s'il peut être optionnel, dans une grande partie des systèmes multi-agents car cela permet des interactions plus riches entre les agents. La communication est un vecteur d'interaction utilisant un langage permettant la transmission et la réception de représentation car permettant la description de ces représentations. Ces représentations peuvent être des représentations de l'environnement, ou d'un état mental par exemple.

Nous verrons donc les types de langages et leur utilité dans l'aspect communication dans les systèmes multi-agents. Puis nous verrons les définitions et les normes de deux types de langage que sont APL[23] et FIPA-ACL[24].

2.3.1 Types de langages

Il existe de nombreux et différents langages permettant de formaliser et de décrire les échanges et J. Ferber les classe en cinq catégories selon que l'on traite un aspect de la communication de haut niveau (langages permettant l'abstraction) ou de bas niveau (langages permettant la réalisation ou l'implémentation) comme la figure 2.5 ci-après le montre (Ferber 1995).

---

[23] APL est l'acronyme anglais pour Agent Programming Language ou Langage de Programmation Agent en français.

[24] FIPA-ACL est un composé de deux acronymes : FIPA et ACL. FIPA est la Fondation pour les Agent Physiques Intelligents (Foundation for Intelligent Physical Agents) qui est à l'origine d'une norme pour les ACL qui sont des Langages de Communication Agent (Agent Communication Language).

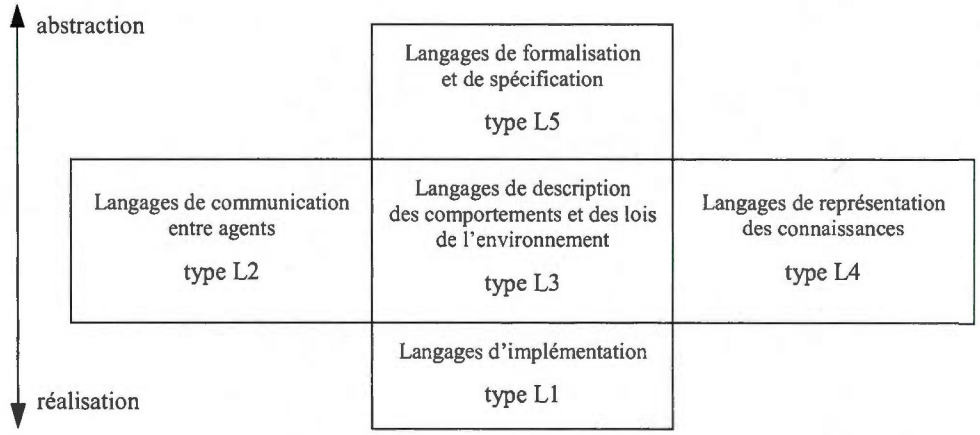


Figure 2.5 Langages et formalismes dans la conception des SMA (J. Ferber).

Le type L1 représente les langages d'implémentation qui sont les langages utilisés pour la programmation d'un système multi-agents. On comprend également dans ces langages les structures que l'on utilise pour implémenter l'environnement et les agents comme les mécanismes de parallélisme, le cerveau des agents, les protocoles informatiques de communications, les environnements de développement. Par exemple des langages tels que C, C++ et Java, mais également LISP ou Prolog.

Le type L2 représente les langages de communication entre agents. Ces langages permettent aux agents d'interagir entre eux au moyen de la transmission d'informations, de connaissances ou de services, ce qui permet à terme la coopération (notion que l'on verra dans la section 2.4 Coordination).

Le type L3 représente les langages de description des comportements et des lois de l'environnement. Ce type de langage est plus abstrait et utilise un certain formalisme pour ne pas gêner l'implémentation de comportements ou l'implémentation des lois physiques de l'environnement. Le fait d'utiliser un formalisme permet de faire abstraction du langage dans lequel l'implémentation se fera afin de garantir, quelle que soit la description des comportements ou des lois, le bon respect du formalisme dans n'importe quel langage d'implémentation.

Le type L4 représente les langages de représentation des connaissances. Ce type de langage est essentiellement utilisé lorsqu'on est amenés à implémenter des agents cognitifs ou hybrides car ces agents doivent être capable de manipuler des représentations qu'ils se font ou se transmettent de leur environnement et de leurs états mentaux. Des langages de type impératif sont généralement utilisés, comme des langages à base de règles par exemple, permettant la manipulation et l'inférence sur ces représentations.

Le type L5 est le plus abstrait et représente les langages de formalisation et de spécification des systèmes multi-agents. Ce type de langage permet de formaliser le système multi-agents que l'on veut développer, et de formaliser toutes les notions dont on aura besoin, comme les interactions, les objectifs, ainsi que les contraintes de développement. Grâce aux langages de spécification, on permet le respect des règles et contraintes du système multi-agents à créer lors de son implémentation avec des langages de type L1, L2, L3, et L4.

2.3.2 APL

Dans la littérature, APL est un terme très souvent utilisé pour désigner les langages de programmation agent. Cette catégorie de langage est dédiée aux langages permettant de programmer des agents dans l'idée de développer des systèmes multi-agents. Certains langages de type APL intègrent d'ailleurs un environnement de développement de systèmes multi-agents, mais axés sur le développement des agents (2APL : en ligne).

Nous verrons quelques uns de ces langages dans la section 2.5 Implémentations et Applications.

2.3.3 FIPA-ACL

Tout d'abord, FIPA est une fondation dont le but est la promotion de technologies des agents et des systèmes multi-agents (FIPA : en ligne). La promotion de ces technologies passe notamment par l'établissement de standards qui servent à spécifier les langages et les contraintes d'interopérabilité avec d'autres langages, services et plateformes.

La spécification, qui sert de norme aujourd'hui, qui nous intéressent est celle concernant les langages de communication agent (ou ACL dans la littérature) dont nous verrons également quelques exemples dans la section 2.5 Implémentations et Applications.

Un ACL prend place au-dessus des protocoles de transfert d'information, et concerne l'aspect social et intentionnel des agents. Lorsqu'un agent a besoin d'interagir avec un autre agent, que ce soit pour de l'aide, de la collaboration ou du conflit, les agents ont besoin de se comprendre entre eux, et l'ACL est là pour définir les règles d'un langage de communication inter-agents : la sémantique de la langue agent, les actions de communication, la structure d'un message et son acheminement.

2.4 Coordination

Dans les systèmes multi-agents, du fait des interactions quasiment omniprésentes entre les agents générant des organisations plus ou moins complexes selon les systèmes, la notion de coordination apparaît comme une nécessité. En effet, lorsqu'un agent interagit pour une raison qui lui est propre, il doit produire plus d'actions que prévu, des actions qui ne sont pas directement source de satisfaction d'un objectif mais sont des étapes intermédiaires nécessaires à l'accomplissement de ce même objectif. Il y a alors un besoin de coordonner ses actions avec ces agents devenus des collaborateurs.

T. W. Malone définit la coordination comme étant « l'ensemble des activités supplémentaires qu'il est nécessaire d'accomplir dans un environnement multi-agents et qu'un seul agent poursuivant les mêmes buts n'accomplirait pas ». La coordination n'implique pas nécessaire de communication, mais comprend la notion d'influence et le fait que les comportements des agents devant se coordonner doivent s'adapter à un contexte particulier, par exemple deux agents qui doivent éviter une collision mais dont les routes se croisent (Malone 1988). Les agents devant se coordonner doivent donc effectuer un traitement additionnel de l'information.

La coordination devient nécessaire lors de différents types d'évènements :

- Le besoin d'une information ou d'un résultat que seul un autre agent peut produire, comme un spécialiste par exemple.

- La limitation des ressources de l'environnement, ce qui peut générer des situations de conflits par exemple.
- L'optimisation des coûts, que parfois liée à une limitation des ressources, qui permet à plusieurs agents décidant de se coordonner d'économiser leurs efforts pour produire le même résultat.
- Les agents qui ont des objectifs différents mais dont les étapes intermédiaires sont dépendantes des objectifs ou d'étapes intermédiaires d'autres agents.

Pour détailler la notion de coordination, nous allons donc voir différentes typologies de coordination, différentes stratégies de coordination, ainsi que la notion de conflits et leur résolution.

2.4.1 Typologies

Nous pouvons distinguer trois grandes typologies de coordination ou d'interaction, dans lesquelles on peut trouver des variantes : les relations d'indépendance, de collaboration, et de compétition.

2.4.1.1 Indépendance

Lorsque dans un système multi-agents, les agents ont chacun des buts compatibles, des ressources suffisantes, et les compétences suffisantes, chacun peut tendre vers la satisfaction de ses objectifs tout en pouvant éviter les formes d'interaction, ce qui fait que les agents sont indépendants entre eux. Cette typologie est la plus simple dans le cas des systèmes multi-agents.

2.4.1.2 Collaboration

La collaboration est un concept mettant en œuvre les différents processus de coopération entre les agents afin de satisfaire des buts, qu'ils soient communs ou non, en mutualisant les ressources et compétences, permettant une plus grande robustesse dans le traitement des objectifs. D'après les travaux de J. M. Hoc, nous pouvons distinguer trois types de coopération (Hoc 1996) :

- La coopération confrontative est une coopération qui implique plusieurs agents de spécialité différente exécutant la même tâche sur les mêmes ressources. C'est une relation

concurrentielle mais du fait des spécialités différentes des agents, et les différents résultats de chaque agent peuvent être fusionnés pour donner le résultat global obtenu de cette coopération.

- La coopération augmentative est une coopération dont les ressources sur lesquelles les agents travaillent vont être réparties entre les différents agents qui sont de même spécialités. Cela va permettre de répartir la tâche à chaque agent et chacun va donner un résultat local. Contrairement à la coopération confrontative, il n'y a pas de fusion car le résultat global est tout simplement l'ensemble des résultats locaux.
- La coopération intégrative correspond à une tâche qui est décomposée en un ensemble de sous-tâches qui vont devoir être traitées par des agents de spécialités différentes, dépendamment de la sous-tâche à traiter. La coordination est de mise et permet le traitement des sous-tâches à la file pour donner le résultat final une fois l'ensemble des sous-tâches accomplies. Une chaîne de montage automobile illustre bien la coopération intégrative.

Il peut être intéressant de s'attarder sur les raisons qui poussent des agents à collaborer. Cela peut être comme lors de la coopération intégrative où c'est uniquement l'intention de l'agent d'aller vers ce type de coopération, cela peut être aussi à cause d'une dépendance où l'agent a besoin d'une compétence d'un autre agent pour satisfaire tout ou partie de son but, cela peut également être dans le cadre de la résolution de conflit comme pour l'accaparement d'une ressource ou dans un cadre de compétition.

2.4.1.3 Compétition

Lorsque les ressources d'un environnement ne sont pas suffisantes pour satisfaire tous les agents ou lorsque des buts des agents sont incompatibles entre eux, des situations de compétition, de même que de conflit comme on le verra dans le point 2.4.3 Conflits, apparaissent. Nous distinguons donc deux genres de compétition, la compétition en tant que telle et les situations encombrement.

Quand les agents doivent se presser pour accomplir une tâche avant que le voisin termine la sienne, il y a un aspect de lutte pour avoir un enjeu particulier. Par exemple dans un jeu ou la

survie est conditionnée par l'accomplissement de tâche avant ses voisins. La compétition peut être vue sous deux aspects :

- L'aspect individuel, qui a pour origine une incompatibilité entre les buts de chaque agent, qui montre la tendance des agents à s'affronter ou négocier pour satisfaire leurs buts. On notera que les ressources ne sont pas importantes dans ce type de compétition. Cet aspect s'apparente fortement à la compétition sportive.
- L'aspect collectif ressemble à l'origine à la compétition individuelle mais lorsque les agents manquent de certaines compétences nécessaires à la satisfaction, ils doivent collaborer avec certains agents qui disposent des compétences manquantes. Des groupes se forment pour assurer un ensemble de compétences utiles à l'aboutissement d'une tâche, puis ces groupes s'affrontent à la manière de la compétition individuelle.

Les situations d'encombrement apparaissent quand les agents doivent se presser pour avoir accès à une ressource limitée avant que l'agent voisin y ait accès pour remplir les mêmes objectifs, il y a gêne car les agents se gênent mutuellement quant à l'accomplissement des objectifs. Il est aisé de faire l'analogie avec une rupture de stock ou d'approvisionnement qui stoppe une chaîne de production. Dans ce cas d'agents informatiques, les ressources peuvent être matérielles comme l'accès au processeur par exemple.

Pour beaucoup de cas de compétition, différentes techniques de coordination sont utilisées afin de les éviter, car pouvant parasiter le système en terme d'interactions et le ralentir en terme d'efficacité.

2.4.2 Stratégies de coordination

Pour ce qui est de la coordination en tant que telle, qui est un concept fortement lié à la planification, on distingue généralement deux stratégies : la planification centralisée et la planification distribuée. Choisir l'une ou l'autre de ces stratégies implique des contraintes futures lors de l'implémentation des agents et de leur structure de communication.

2.4.2.1 Planification centralisée

Cette stratégie de coordination repose sur le principe de la centralisation du pouvoir décisionnel. Une entité a le rôle d'administrateur et gère la planification et les conflits. Cela

permet à un groupe d'avoir un leader naturel qui évitera aux agents de générer beaucoup d'interactions par eux-mêmes en vue de résoudre des situations conflictuelles. On suppose que cette entité administrative connaît les contraintes et les compétences des agents du groupe, et on suppose également qu'elle est capable de découper les tâches. Ce type de planification reste très similaire à un problème de classification classique que l'on peut trouver dans de nombreux systèmes.

2.4.2.2 Planification distribuée

Contrairement à la stratégie de coordination centralisée, la planification distribuée ne dispose pas d'une entité gérant la planification. Dans ce type de configurations, tous les agents font leur propre planification et définissent donc leurs propres plans, aussi appelés des plans partiels car ce sont des sous-tâches à réaliser dans le cadre d'une tâche complète. On suppose que les agents disposent d'une capacité à communiquer conséquente : ils doivent être capables de transmettre leurs buts et intentions ainsi que leurs propres contraintes (par exemple une compétence non disponible ou un accès exclusif à une ressource).

Bien entendu, de cette stratégie de coordination a quelques inconvénients qui résultent du pari de l'émergence. Il se peut qu'un problème ne puisse pas être totalement résolu, notamment du fait de l'auto-planification faite par chaque agent donc la coordination n'est pas aussi rigoureuse que lorsqu'une entité d'administration peut coordonner le tout. Il se peut que la planification devienne laborieuse du fait que les plans partiels de chaque agent peuvent éventuellement être source de conflit. Du même coup, une quantité non négligeable d'interactions, notamment concernant les communications, est générée, ce qui peut ralentir la vitesse de résolution d'un problème. Une manière de soulager ce problème est d'agencer la transmission de plans partiels entre les agents afin qu'ils puissent se prévenir de conflits potentiels avant d'être devant le fait accompli si l'on peut dire ainsi.

2.4.3 Conflits

Les conflits sont un autre type de coordination impliquant d'autres types de comportement. Nous avons vu précédemment que lorsque les objectifs des agents sont incompatibles entre eux, les agents sont dans une situation de compétition. En plus de cet état de fait, si l'on suppose que les ressources sont insuffisantes pour permettre à chaque agent

d'atteindre ses objectifs, alors la compétition tourne en un cas particulier, c'est-à-dire une situation conflictuelle.

Comme pour la compétition, il y a généralement deux types de conflits, dont l'objet est principalement les ressources :

- Les conflits individuels où la volonté de l'agent de s'accaparer les ressources selon ses propres objectifs (possiblement des besoins). Un exemple parlant est la conquête ou la défense d'un territoire ou la recherche d'une position dominante.
- Les conflits collectifs où les compétences des agents ne sont pas forcément complètes et où ils doivent donc collaborer avec certains autres pour s'accaparer les ressources convoitées, en groupe. Les guerres en sont une bonne illustration.

Pour le bon fonctionnement d'un système multi-agents et à moins qu'il ait été conçu pour étudier des situations conflictuelles, il paraît nécessaire que les conflits entre agents ou entre groupes d'agents puissent être résolus. Trois types de solution permettent la résolution d'un conflit : la résolution a priori, l'arbitrage et la négociation. Comme la résolution de conflits fait partie intégrante de l'aspect collaboratif des agents dans les systèmes multi-agents, une partie d'entre elles peuvent émerger des multiples interactions autour du conflit.

2.4.3.1 Résolution a priori

La résolution a priori d'un conflit est l'illustration d'une domination hiérarchique. En effet, dans ce cas de figure, c'est une autorité supérieure qui est chargée de résoudre le conflit, par l'usage de la force ou de cette autorité. L'autorité supérieure peut être une organisation mais aussi un agent (qui disposerait donc d'un pouvoir sur d'autres agents).

2.4.3.2 Arbitrage

L'arbitrage est peut être le type de résolution de conflit le moins évident à appréhender, notamment dans le cadre de la programmation agent. Cela consiste en un agent neutre face à la situation de conflit, qui jouera le rôle de médiateur et aura pour objectif de résoudre les conflits au travers des différents points de vue agent.

2.4.3.3 Négociation

La négociation est peut être la méthode de résolution de conflit la plus parlante. L'objectif de la négociation est de trouver une solution qui convienne à tous les agents qui sont en conflits. On distingue deux types de négociation :

- La compromission : pour cela, les agents en question ont besoin d'interagir de nombreuses fois afin de trouver petit à petit des compromis qui aboutissent à une solution envisageable, c'est-à-dire une solution permettant aux agents de continuer vers leurs objectifs. Ces compromis consistent à vérifier l'importance de chaque contrainte de l'agent et d'abandonner ou mettre de côté les moins importantes. Lorsque les contraintes restantes chez les deux parties sont satisfaisantes, la négociation se termine et le conflit est résolu.
- La remise en question : cela ressemble à la compromission, mais on adapte les buts et non les contraintes. Ici, chaque agent cherche à vérifier ses propres objectifs profonds de manière à échanger des informations et intervenir sur ces buts profonds, et non sur des buts intermédiaires. Cela revient à adapter les objectifs intermédiaires, les modifier ou les redéfinir complètement, de manière à garantir l'objectif profond de chaque agent.

2.5 Implémentations et Applications

Dans cette partie nous allons voir les principaux langages de programmation agent, mais également les plateformes et librairies conçues pour le développement de système multi-agents. Puis nous allons voir des exemples et principes d'application courant.

2.5.1 Langages de programmation agent et orientés agent.

Il n'y a pas vraiment de règles dans les langages informatiques en ce qui concerne l'implémentation des agents ou l'implémentation de systèmes multi-agents. Il y a bien sûr des langages qui sont plus adéquats, mais également des langages qui ont été conçus spécialement pour la conception d'agent, et notre attention va se porter sur ces derniers. De même que des langages ont été créés pour concevoir des agents, des normes ainsi que des formalismes ont été développés pour faciliter la mise en œuvre de la conception d'agents et de systèmes multi-agents.

Parmi les nombreux langages, normes et formalismes qui nous ne verrons pas dans cette section, il y a :

- Le langage GOAL, utilisé pour la création de systèmes multi-agents dont les agents sont de type BDI.
- Le format KIF (Knowledge Interchange Format) pour l'échange de connaissances entre les agents
- Le langage 2APL qui est un composant pouvant s'ajouter à Eclipse[25] et ajoutant une plateforme d'exécution pour ce langage.
- Le langage AgentSpeak orienté pour les agents BDI et basé sur la programmation logique.
- L'interpréteur Jason, en Java, qui étend le langage AgentSpeak (Jason :en ligne).
- Le langage Golog, basé sur Prolog, qui est un langage de haut-niveau pour agent cognitif.

On verra donc les langages Agent0, MetateM et FIPA-ACL, ainsi que la norme KQML et le formalisme BRIC de J. Ferber.

2.5.1.1 Agent0

Agent0 est un langage de programmation multi-agents créé par Y. Shoham qui utilise le paradigme de la programmation orienté agent. Les implémentations dans ce langage s'exécutent sur un système utilisant LISP (Torrance et al. 1991 : en ligne).

Avec Agent0, à l'instar de la programmation orientée objet où l'on commence par définir les objets, on commence par définir les agents qui vont être implémentés, puis chargés de s'exécuter dans l'environnement. Il faut noter que l'environnement est totalement synchrone, c'est-à-dire que chaque action d'un agent est synchronisée avec celles des autres agents. Les agents conçus avec Agent0 sont des agents disposant d'une grande capacité de communication et ayant deux types de représentation : les croyances (connaissance de l'environnement à l'instant) et les engagements (la planification d'exécution). Chaque agent

---

[25] Eclipse est un célèbre environnement de développement intégré (IDE en anglais), principalement utilisé pour le développement en Java.

Agent0 peut agir de six manière différentes : faire, informer, demander, annuler, répéter, si vrai alors faire.

C'est un langage assez ancien, mais qui a posé les bases du paradigme de la programmation orientée agent.

2.5.1.2 MetateM

MetateM est lui aussi un langage de programmation orienté agent, et il est basé sur la logique temporelle[26]. Le développement d'agents se fait avec un interpréteur du langage MetateM qui s'exécute en Java (MetateM : en ligne).

Avec le langage MetateM, les spécifications des agents sont définies, et ce sont ces spécifications qui sont exécutées et qui déterminent le comportement des agents. De ce fait, la logique des agents est toujours valide même lors d'une traduction éventuelle dans un langage de plus bas niveau.

2.5.1.3 FIPA-ACL

FIPA-ACL est un langage de communication agent de type L2 en plus d'être une norme pour les langages de communication agent en général comme nous l'avons vu en 2.3.3 FIPA-ACL. Ce langage est basé sur la théorie des actes du langage de J. Searle et son objectif est de permettre de faire communiquer des agents entre eux et n'utilisant pas nécessairement les mêmes protocoles de communication (Searle 1969). FIPA-ACL permet donc ceci par l'implémentation d'une interface pour chaque protocole, permettant une certaine forme de traduction d'un protocole en un autre et une standardisation des messages échangés entre les agents.

---

[26] La logique temporelle appartient à la logique mathématique et implique que des propositions formelles peuvent vraies ou fausses dépendamment de l'évolution de l'environnement. L'état vrai ou faux d'une proposition peut donc varier avec le temps pour le même environnement.

Ce langage nécessite que soit implémenté un système de gestion des agents, ainsi qu'un service d'annuaire permettant aux agents de savoir quels agents disposent de telles compétences. Les spécificités de ce langage sont l'uniformisation des actions de langage (confirmer, demander, informer, annuler, proposer, etc.), la structuration des messages (syntaxe, protocole, langue parlée, etc.) et la gestion du transport des messages (expéditeur, destinataire, taille du message, date, etc.).

Il faut noter que les spécifications de la FIPA concernant FIPA-ACL sont encore soumises à évolution et sont disponibles sur leur site internet (FIPA : en ligne).

2.5.1.4 KQML

KQML (Knowledge Query and Manipulation Language en anglais) est plus ancien que FIPA-ACL et est également un langage de communication agent de type L2. Comme pour FIPA-ACL, il est basé sur la même théorie des actes du langage. Mais les spécifications de KQML sont relativement légères, comme l'ont montré P. R. Cohen et H. J. Levesque, et sont donc sources de quelques défauts. Ce langage permet toutefois de standardiser les messages échangés entre les agents (Cohen et al. 1995).

2.5.1.5 BRIC

BRIC est un formalisme créé par J. Ferber mais il le définit : « BRIC (Block-like Representation of Interactive Components) est un langage de haut niveau permettant de concevoir et de réaliser des systèmes multi-agents à partir d'une approche modulaire (Ferber 1995). Un système BRIC comprend un ensemble de composants reliés entre eux par des liens de communication. » Le formalisme BRIC est donc une approche « componentielle » dans lequel sont utilisés les langages de type L3.

Un composant BRIC est alors défini comme une structure qui a des entrées et des sorties, et donc l'intérieur contient d'autres composants (ces composants classiques sont une instance de classe, comme pour les objets). Pour les composants BRIC qui sont en fait composés de plusieurs composants BRIC, on dit que ce sont des composants structurés et le comportement

est alors spécifié. Le comportement peut aussi être spécifié avec un réseau de Petri[27], et ces composants sont qualifiés d'élémentaires.

Le formalisme décrit également les messages qui sont des liens de communications, et les conventions et équivalences, notamment pour la traduction en réseaux de Petri.

2.5.2 Librairies et plateformes de développement

Pour développer des systèmes multi-agents ou s'essayer à leur conception et implémentation, il existe différentes librairies et plateformes de développement. Nous ne les verrons pas toutes mais seulement les plus célèbres et utilisées, à savoir JADE et son extension JADEX, MADKIT et Agent Builder.

2.5.2.1 JADE et JADEX

JADE (Java Agent DEvelopment) est une plateforme de développement agent et systèmes multi-agents développée en Java. Cette plateforme est également un logiciel libre. Il est important de noter que JADE supporte la norme FIPA-ACL (JADE : en ligne). Le débogage ainsi que le déploiement des systèmes développés sont totalement supportés par JADE. La plateforme est composée de trois éléments nécessaires pour produire un système multi-agents : un service d'annuaire pour les agents, un service gérant la communication entre les agents, et un service de gestion des agents.

JADEX est une extension de JADE, mais est désormais totalement indépendante, et est développée en Java également (JADEX : en ligne). Cette plateforme développée par l'université de Hambourg est capable de faire tout ce que JADE fait au niveau du déploiement, des tests et de l'implémentation d'agents et de systèmes multi-agent, mais apporte une dimension plus modulaire dans la conception de systèmes notamment avec les modules Jadex Agents, Jadex Processes, Jadex Rules et Jadex XML. Il est important de noter que JADEX permet l'implémentation d'agent BDI.

---

[27] Les réseaux de Petri manipulent des variables discrètes, et est un modèle mathématique qui est utilisé pour la représentation de systèmes informatiques ou industriels.

2.5.2.2 MaDKit

MaDKit (Multi-Agent Development KIT) est une plateforme de développement multi-agents évolutive et modulable développée par O. Gutknecht et J. Ferber. La plateforme est également un logiciel libre. MaDKit est très souple et permet la conception de systèmes multi-agents en laissant une grande liberté dans la structure même des agents (permet facilement la création d'agents cognitifs et réactifs) ainsi que dans les protocoles de communication (Ferber et al. 2000, MaDKit : en ligne).

La plateforme permet un grand parallélisme, c'est-à-dire qu'elle permet l'action en parallèle d'un très grand nombre d'agents, comme dans le cas d'un système multi-agents dont les agents sont entièrement réactifs par exemple. Cela permet de créer simplement toutes sortes de simulation tout en gérant simplement les caractéristiques des environnements.

Cette plateforme est clairement orientée sur l'organisation des systèmes multi-agents et intègre nativement le modèle AGR ainsi qu'AGRS dans ses dernières versions.

2.5.2.3 Agent Builder

Agent Builder est un environnement de développement complet de systèmes multi-agents, implémenté en Java et s'exécutant dessus. Il se base sur deux composantes : la boîte à outil pour concevoir les agents, et l'environnement d'exécution. L'environnement de développement inclut la gestion du développement de logiciels basés sur des agents, l'analyse des domaines d'activités des agents, la spécification des comportements agent, la conception des réseaux de communications inter-agents, et bien sûr des outils pour tester et déboguer les agents logiciels. Les agents conçus avec Agent Builder utilisent KQML pour les communications et sont de type BDI implémenté par le langage Agent0.

Il est intéressant de noter qu'Agent Builder intègre une partie dédiée aux ontologies[28], que peuvent utiliser les agents. Agent Builder est donc principalement conçu pour développer des systèmes multi-agents purement logiciels, utilisant spécifiquement des agents intelligents.

2.5.3 Domaines d'applications

Le domaine systèmes multi-agents est très large et ouvert. Selon J. Ferber, tenter de détailler chaque domaine d'application serait compliqué du fait de la très grande variété de domaines possibles. Il a donc distingué cinq grandes familles de domaines d'application des systèmes multi-agents :

- La résolution de problèmes au sens large dans laquelle on peut distinguer plusieurs types de résolution. Tout d'abord il y a la résolution distribuée de problèmes qui consiste à résoudre un unique problème en faisant appel à plusieurs spécialistes ayant pour objectif la résolution de cet unique problème. Il y a également la résolution de problèmes distribués qui consiste en la résolution distribuée comme précédemment mais d'un problème qui n'est pas centralisé, comme par exemple les systèmes de gestion d'énergie, de communication ou de surveillance. Dans le dernier cas on trouve la résolution par coordination, qui est une résolution de problèmes tout à fait classique où le problème n'est pas distribué et où il n'y a pas de compétences d'expertise requises.
- La simulation multi-agents qui étend le champ de recherche des simulations en général, qui est très développé en informatique. La simulation permet l'étude de modèles théoriques s'appliquant à des domaines concrets et la simulation multi-agents en particulier permet d'étudier des problèmes de populations, de phénomènes chimiques et physiques comme la météo, ou encore des problèmes liés aux sciences sociales. Ce type de simulation permet de prévoir l'évolution de tels systèmes dans des cas concrets et possiblement d'en comprendre les mécanismes.

---

[28] Le domaine des ontologies est particulièrement développé, mais si on devait donner une définition simple en informatique, une ontologie serait la représentation d'un champ d'informations par les termes et concepts liés.

- L'élaboration de mondes virtuels qui permettent de manipuler un aspect des systèmes multi-agents : l'environnement. En effet, nous avons vu que l'environnement est un facteur fondamental dans les systèmes multi-agents, et l'élaboration de ceux-ci ainsi que leur manipulation par ajustement par exemple, permet d'exercer une influence et permet également des analyses plus poussées concernant les interactions des agents avec leur environnement. C'est un aspect qui est également utilisé dans le cadre des simulations multi-agents.
- La robotique distribuée qui est un domaine d'application concret, et non une simulation, dont le but est la résolution d'un problème avec des agents robotiques réels dans un environnement réel. J. Ferber distingue deux types de robotique distribuée. Le premier type est la robotique dite cellulaire dans laquelle un robot est le système multi-agents et dans lequel chacun de ses composants est un agent. Cela permet de développer des robots permettant l'économie de calculs et de mouvements pour une action. Le second est la robotique dite mobile où chaque robot est un agent du système et où la coordination de ces robots permet la résolution de tâches telles que l'exploration en territoire inconnu, la surveillance, où la gestion des décollages et atterrissages dans un aéroport.
- La kénétique, qui est une tentative de J. Ferber de définir une science et une technique permettant l'étude et la conception de systèmes artificiels basés sur des organisations d'agents pouvant agir, collaborer, communiquer et planifier. Il y a un aspect purement informatique dans la kénétique où il est proposé de concevoir des logiciels encore plus modulaires qu'aujourd'hui, en se basant sur les notions d'agents et d'interactions. Un exemple serait un logiciel, adapté à l'utilisateur, de gestion de vie quotidienne.

CHAPITRE III

QUELQUES RECOMMANDATIONS EN MATIERE D'IMPLEMENTATION DE SYSTEMES MULTI-AGENTS

Dans les chapitres précédents, nous avons vu ce que sont les agents et quels sont leurs types et architectures, nous avons également vu ce que sont les systèmes multi-agents et quels sont leurs architectures et modèles organisationnels. Dans ce chapitre nous allons donc explorer les recommandations pour la conception et l'implémentation de systèmes multi-agents.

Pour ce faire nous allons donc voir en premier lieu dans quels types de cas il est intéressant d'utiliser les systèmes multi-agents. Ensuite nous allons voir comment choisir un type d'agent ainsi qu'une architecture d'agent, pièce maîtresse de tout système multi-agent. Puis nous allons voir comment choisir un modèle de système multi-agents ainsi que le niveau organisationnel de celui-ci en fonction des besoins. Enfin vient le choix d'un environnement de développement à proprement parler, dans lequel nous allons voir quels sont les choix possibles de langages d'implémentation dont nous pouvons avoir besoin.

Des questions types, avec une numérotation ordonnée, seront proposées tout au long de ce chapitre afin de guider le processus de réflexion concernant l'intérêt de concevoir ou non un système multi-agents.

3.1 Pourquoi concevoir et implémenter un système multi-agents

Avant d'envisager la conception d'un système multi-agents, il convient de se demander si un tel système est nécessaire pour résoudre la ou les problématiques identifiées. Il faut aussi se demander si le besoin d'un système purement logiciel ou d'un système plus robotisé

pouvant interagir dans un environnement réel est nécessaire. Il convient donc d'exprimer clairement la problématique générale ainsi que les besoins que l'on a par rapport à cette problématique.

Il existe bon nombre de problématiques pouvant être résolues de manière classique par l'ingénierie logicielle. Des solutions existant déjà, est-il nécessaire de développer un système multi-agents pour offrir de nouvelles solutions ? Cela dépend des besoins que nous devons définir, car si une solution existe et que nous avons encore des besoins, c'est qu'elle ne nous convient pas. Par exemple une solution informatique classique ne donne pas satisfaction sur le temps d'exécution, ou alors est satisfaisante au niveau du temps mais pas au niveau de la productivité par rapport à ce qu'il est humainement possible de faire. Dans certains cas, une étude plus poussée de ces besoins peut mettre en évidence la nécessité de trouver une autre solution par un système multi-agents.

Pour bon nombre de problèmes ne pouvant pas être résolus de manière conventionnelle, c'est-à-dire par des algorithmes ou parfois même par des systèmes experts, ou de façon raisonnable, c'est-à-dire avec un rapport de temps ou de productivité intéressant, la nécessité de développer un système multi-agents peut alors facilement devenir concevable.

Attention toutefois : un système multi-agents n'offre aucune garantie de résolution d'un problème actuellement impossible à résoudre de manière conventionnelle. Mais il permet de changer de paradigme dans la résolution de ce problème pour espérer offrir une résolution convenable en termes d'efficacité, de temps ou de coût.

De manière générale, il y a bien des situations où les systèmes multi-agents sont la première solution suggérée, comme par exemple les simulations de particules, les simulations de comportements sociaux, ou tout autre système dont la résolution passe uniquement par de multiples entités logicielles ou matérielles interagissant entre elles avec un objectif commun.

Il y a donc deux questions types, posées ci-après, qui peuvent permettre de décider de la nécessité de concevoir un système multi-agents.

Question n°1 : Le problème nécessite-t-il l'utilisation de plusieurs entités agissant dans un cadre défini ? Il est clair qu'un système multi-agents est plus adapté pour ce besoin qu'une solution plus classique de type itérative ou procédurale.

Question n°2 : Le problème peut-il être décomposé en plusieurs sous-problèmes (différents ou similaires) ? La question se porte sur la distribution de la résolution du problème. Si le problème en question ne nécessite pas obligatoirement plusieurs entités pour sa résolution, il faut étudier la faisabilité d'un autre type de solution comme une solution itérative pouvant être parallélisée et voir si la complexité ou le coût en temps ou ressources rend la solution d'un système multi-agents intéressante. Si la réponse à la question n°1 est affirmative, la nécessité d'un système multi-agents ne peut qu'être appuyée.

3.2 Choix d'un type d'agent et d'une architecture agent

Après avoir décidé de la nécessité de développer un système multi-agents, il convient d'affiner les besoins en commençant par le niveau local de problèmes que l'on doit résoudre. Il ne faut pas non plus oublier dans quel type d'environnement les agents, et donc le système multi-agents, vont évoluer. Pour cerner le type d'environnement et donc ses propriétés, on se référera en détail au point 1.1.3.2 Propriétés des environnements du premier chapitre. La complexité de l'environnement que l'on va devoir définir a une grande influence sur le type d'agent nécessaire au système que l'on veut concevoir.

Affiner les besoins que l'on a en terme de solution globale revient à bien définir le type d'agent dont on va avoir besoin et les capacités dont il doit être doté tant au niveau de ses actions que de son cycle cognitif. Il ne faut pas oublier bien sûr de bien définir les tâches que chaque agent va devoir accomplir, découlant du problème global à résoudre, car ce seront leurs objectifs et leur raison d'être.

En reprenant la classification des agents établie au point 1.2.3 Classification des agents, dans le premier chapitre, et en les confrontant à certaines propriétés d'un environnement, on obtient le tableau 3.1 ci-après. C'est-à-dire qu'en sachant dans quel environnement les agents vont évoluer, on peut orienter le choix du type d'agent sur lequel se concentrer. La propriété mono-agent/multi-agents a volontairement été mise de côté du fait de notre objectif de

concevoir un système multi-agents. Les propriétés discret/continu, déterministe/stochastique et statique/dynamique ne sont pas représentées car elles influent sur le comportement des agents mais pas sur notre choix de type d'agent. En effet, ces trois dernières propriétés sont à prendre en compte lors de la conception du cycle cognitif des agents.

Question n°3 : Quelles sont les caractéristiques de l'environnement du problème (le cadre) ? Et quel est donc le type d'agent le plus adéquat par rapport aux caractéristiques de cet environnement ? Le tableau 3.1 suivant illustre les choix possibles à cette question selon les caractéristiques définies.

Tableau 3.1 Types d'agent selon certaines caractéristiques environnementales.

	Observable	**Episodique**
Tropique	Partiellement	Episodique
Pulsionnel	Partiellement	Séquentiel
Module	Entièrement et partiellement	Episodique
Intentionnel	Entièrement et partiellement	Séquentiel

Il ne faudra pas oublier, selon l'environnement de développement que l'on utilise, de définir des mesures de performances pour bien équilibrer les agents selon leur environnement et selon nos besoins.

Il faut également penser, toujours en se basant sur la classification des agents vue dans le premier chapitre, si l'on a besoin d'agents réactifs ou d'agents cognitifs, mais également si l'on a besoin que nos agents aient un comportement réflexe ou téléonomique. Cela va permettre de définir le cycle cognitif que l'on veut pour concevoir les agents, et donc de choisir l'architecture idéale des agents pour la solution.

Question n°4 : Quelle architecture agent est appropriée pour que les agents soient efficaces ? Le type d'agent dépend en grande partie de son cycle cognitif, et donc de son

architecture. Il est donc important de définir les capacités voulues chez les agents et par conséquent le cycle cognitif, qui est le cœur du raisonnement d'un agent.

Nous avons vu dans la section 1.3 des architectures plutôt développées comme BDI, ACT-R, IDA et CTS qui sont des architectures pour développer des agents de type cognitif ou hybride ainsi que l'architecture en couche TouringMachine qui est plutôt orientée pour les agents réactifs. Plus un agent se rapproche d'un comportement réflexe et est de type réactif et plus son architecture sera simple. Par exemple, des agents représentant des molécules dans une simulation ont une architecture très simple, que l'on pourrait désigner comme un ensemble restreint de règles immuables. A l'inverse, des agents représentant des individus autonomes, et doués d'une intelligence propre, les architectures sont beaucoup plus complexes, à l'image des architectures de type BDI ou encore CTS. Il est à noter que l'architecture CTS est tout à fait adaptée pour les systèmes destinés au tutorat ou plus largement aux interfaces homme-machine.

3.3 Choix d'un modèle et du niveau organisationnel

Après avoir sélectionné le type d'agents ainsi que le modèle d'architecture nécessaires aux agents du système que l'on veut concevoir, il faut passer à l'aspect du système multi-agents en tant que tel. Le domaine d'application dans lequel on veut mettre en œuvre un tel système est important, comme nous l'avons succinctement vu dans le point 2.5.3 Domaines d'applications du deuxième chapitre. Si le domaine d'application concerne la résolution de problèmes distribués, ou la résolution distribuée de problèmes, ou la simulation, ou la génération d'environnements, ou encore la robotique distribuée, les types d'organisation ainsi que les niveaux d'organisation au sein des systèmes multi-agents vont varier. Les stratégies de coordination sont également un point à prendre en compte puisque cela influe sur le type d'organisation d'un système.

Question n°5 : Ai-je besoin d'un modèle d'organisation et si oui, lequel employer pour que la population d'agents soit la plus efficace possible sur mon problème ? Concernant le type d'organisation, l'aspect est lié aux possibilités d'interaction des agents entre eux et avec leur environnement. Souvent le type d'organisation est illustré par des organisations naturelles, comme les colonies, les essaims ou encore les bancs, comme vus dans la section 2.2 du

deuxième chapitre. De ces types d'organisation ressortent différents modèles dont les plus couramment utilisés sont les modèles AGR ainsi que de sa version étendue AGRS. Ces modèles permettent de prioriser les interactions et donc d'orienter l'émergence des organisations internes au système. Les modèles AGRS, plus récent et plus complet que son grand frère AGR, permet une plus grande souplesse pour les systèmes multi-agents dont les groupes organisationnels internes voient leur nombre de membres fluctuer. En effet, il peut arriver que l'on conçoive des systèmes multi-agents dont les agents peuvent être créés ou détruits de manière dynamique, que ce soit mu par les objectifs du système ou bien par imitation de la vie dans le cadre de simulation de populations par exemple. Mais il peut également ne pas avoir besoin d'un modèle, par exemple dans un problème de simulation de particules où les notions de groupe et de rôle n'ont pas lieu d'être.

Question n°6 : Ai-je besoin d'utiliser au moins un niveau d'organisation et si oui, le ou lesquels ? Un autre aspect à considérer concerne les niveaux d'organisation du système multi-agent, que l'on peut revoir de manière détaillée dans le point 2.2.2 du deuxième chapitre. Un système multi-agents peut être conçu pour s'exécuter à un niveau d'organisation spécifique, comme par exemple le niveau *micro-social* ou à l'opposé avec le niveau des *sociétés globales*. Il peut également être conçu pour prendre en compte plusieurs niveaux d'organisations, selon que l'on se focalise un niveau d'organisation particulier du système lors de son développement.

En effet, les niveaux d'organisation dit macroscopiques ont une influence sur les niveaux d'organisation microscopique et vice-versa, le tout grâce aux interactions possibles par les agents.

Par ailleurs, il est intéressant de noter qu'il est possible de prendre les questions de modèles d'organisation (question n°5) et de niveaux organisationnels (question n°6) de manière indépendante. L'une peut être affirmative sans que l'autre le soit.

On arrive donc à l'enjeu des interactions, et donc des communications qui sont un aspect des interactions entre agents. Le langage de communication qui va être utilisé par les agents est très important car c'est lui qui conditionne le degré de liberté des communications inter-agents. Nous avons vu que KQML est plutôt ancien est possède quelques défauts, et

l'utilisation de FIPA-ACL convient mieux car plus étendu, mais également plus normé, ce qui facilite son implémentation dans les agents.

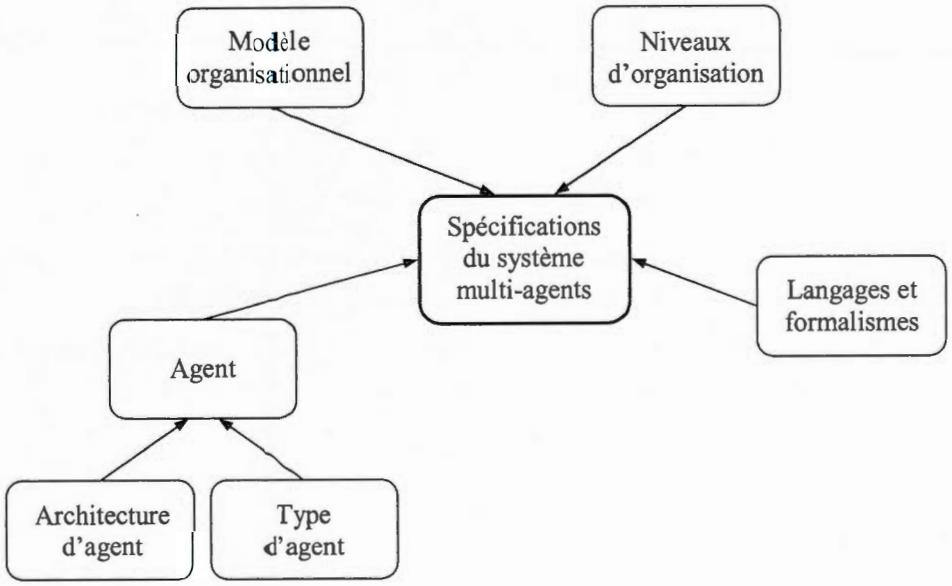


Figure 3.1 Choix influents sur les spécifications d'un système multi-agents.

Les choix d'un modèle ainsi que d'un ou plusieurs niveaux organisationnels permettent, avec les spécifications *agent* définies préalablement, de définir les spécifications du système multi-agents que l'on veut mettre en œuvre, comme l'illustre la figure 3.1.

3.4 Choix d'un environnement de développement

Le choix d'un environnement de développement à proprement parler pour l'implémentation d'un système multi-agents survient, à mon sens, une fois que toutes les spécifications de notre système multi-agents sont définies. On retrouve une certaine méthodologie classique dans la programmation de logiciels avec les spécifications fonctionnelles définies avant le choix des technologies de mise en œuvre et de leurs éventuelles contraintes.

J'attire l'attention sur le fait que le développement d'un agent ou d'un système multi-agents n'est pas uniquement le fait de la programmation. Il est tout à fait possible de développer un

système multi-agents dont les agents sont des robots physiques n'ayant pas ou peu de programmation logicielle interne. Mais dans cette partie nous nous focaliserons sur les environnements de développement logiciel.

Question n°7 : Quels sont les langages de programmation disponibles (sur le matériel utilisé) me permettant de respecter les spécifications de mon système multi-agents ?

Nous avons vu certaines plateformes de développement très connues, mais il en existe de nombreuses, et il convient de bien choisir son environnement de développement multi-agents en fonction des possibilités de ces environnement en terme d'architecture agent et de modèle organisationnel multi-agents. Il convient de noter qu'il n'existe tout de même aucune règle concernant les langages d'implémentation tant que ceux-ci permettent d'implémenter les spécifications définies pour notre système multi-agents.

Question n°8 : Est-il utile d'utiliser un environnement de développement complet spécifique à une architecture d'agent ? Tout cela dépend de la question n°7 : rien n'interdit d'implémenter une architecture dans un langage dans lequel cette architecture n'a jamais été implémentée. Mais est-ce judicieux si le langage suggéré par l'outil est supporté par le matériel sur lequel le système multi-agents doit être mis en place ? Point intéressant, beaucoup d'outils de développement agent et multi-agents sont conçus en Java, ce qui permet une grande portabilité des systèmes multi-agents que l'on peut créer et ce, nativement. Parmi eux on trouve : JACK, JADE, JADEX, JAgent, Jason, MaDKit.

Les agents BDI étant courants car performants, de nombreuses plateformes permettent de les implémenter, comme JADEX et Agent Builder. Mais concernant ces deux plateformes, le langage de communication agent (ACL) diffère : JADEX utilise FIPA-ACL et sa norme plus complète que KQML, tandis que Agent Builder utilise KQML. JADEX permet en plus de concevoir des systèmes multi-agents disposant de stratégies de communications plus variées grâce au service d'annuaire incorporé et au service de gestion des agents.

Ensuite il convient de choisir les langages que l'on va utiliser lors de l'implémentation en fonction de leur type, selon leur classification que nous avons vue au point 2.3.1 dans le chapitre précédent.

3.4.1 Choix des langages d'implémentation

Là encore, il n'existe aucune règle spécifique concernant quel langage d'implémentation est approprié ou non pour tel type de langage, mais J. Ferber en a distingué certains qui sont naturellement plus adaptés à certains types de langage.

On va se focaliser sur les types L1, L2, et L4 car les types L3 et L5 sont plus abstraits et concernent les formalismes que l'on utilise lors de la conception d'un système multi-agent et qui permettent l'utilisation des trois types précédemment cités sans problème d'implémentation particulier (la figure 2.5 rappelle ce que sont dans notre cas les types de langage).

Le langage de type L4, chargé de représenter les connaissances, est souvent implémenté par des langages impératifs permettant des bases de règles, mais aussi par XML dans certains, comme JADEX.

Le langage de type L2 concerne les ACL (langages de communication agent). Nous en avons vu les deux principaux, à savoir KQML ainsi que FIPA-ACL et sa norme. Bien entendu cette liste n'est pas limitative.

Le langage de type L1 concerne l'aspect programmation pure au niveau des protocoles, des mécanismes de gestion du système multi-agents ainsi que des environnements dans lesquels ces systèmes vont évoluer. Ces langages sont les plus courants car ce sont ceux qui vont permettre l'exécution en tant que telle des systèmes multi-agents. On peut citer C, C++, Java ainsi que LISP et Prolog.

Question n°9 : Ai-je besoin, selon l'architecture des agents, d'utiliser un langage de communication ou de représentation ? Et si oui, quels sont les langages de communication ou de représentation disponibles me permettant de respecter les spécifications des agents dans mon système multi-agents ? Cette question est relativement similaire à la question n°7 mais ne concerne pas la programmation des agents à proprement parler. Ici l'accent est mis sur les langages de type L2 et L4, qui sont d'ailleurs complémentaires lorsque la communication entre des agents implique un partage ou une mise en commun des connaissances, qui sont en quelque sorte les langues parlées par les agents.

Chaque langage de programmation, de communication ou de représentation a ses spécificités, ses avantages et inconvénients, que ce soit en termes de portabilité, d'environnement d'exécution, de compilation ou de souplesse de conception. Il convient à chacun de définir quels langages utiliser pour quelles parties du système multi-agents à mettre en œuvre, comme on peut les découper avec les types de langage L1 → L5. Il convient également de bien définir quel l'environnement de développement utiliser, qui peut accepter ou non tels langages spécifiques, et qui dispose d'outil de tests et de débogage des agents.

CONCLUSION

Le domaine de l'intelligence artificielle, et notamment de l'intelligence artificielle dite distribuée, est particulièrement vaste, et beaucoup d'aspects sont flous, sujets à débat, et encore en cours d'étude. L'objectif du présent mémoire est donc de rendre abordable un des aspects que l'on appelle les systèmes multi-agents, et ainsi démystifier la conception de tels systèmes.

Une grande partie introductive concernant les agents, brique fondamentale pour l'élaboration de systèmes multi-agents, a été nécessaire afin de pouvoir monter d'un niveau d'abstraction et d'appréhender ce que sont les systèmes multi-agents. Certains domaines de recherche et développement pour des cas industriels ou sociaux ont beaucoup poussé et motivé le développement des systèmes multi-agents ainsi que des théories qui les entourent. Nous avons donc pu faire le tour des concepts inhérents aux systèmes multi-agents ainsi que des technologies actuelles les concernant.

Ensuite, quelques recommandations en ce qui concerne les choix de conception de systèmes multi-agents ont été proposées. Bien qu'ayant eu la volonté d'aller plus loin sur ces points mais ayant été limité dans le temps, ces propositions ont pour but d'aiguiller le lecteur sur ce qu'il lui est possible d'étudier ou de faire s'il désire approfondir ce sujet.

Beaucoup d'applications concrètes aux systèmes multi-agents restent à découvrir et à mettre en œuvre dans de multiples domaines que nous ne pourrons pas tous les citer ici, mais dont j'esquisserai certaines pistes. L'élaboration de nouveaux modèles organisationnels, pouvant être inspiré par une organisation naturelle non encore découverte, pourrait par exemple apporter une nouvelle façon de voir la résolution d'un ensemble de problèmes difficiles à résoudre à ce jour. Il est également possible d'imaginer à court terme des mises en œuvres concrètes dans le domaine de l'exploration spatiale comme la régulation des systèmes

de vie à l'intérieur d'une station spatiale, ou bien l'automatisation des arrimages de véhicules spatiaux entre eux, ou encore l'exploration terrestre d'autres planètes. La conception de systèmes multi-agents pour l'exploration minière peut être envisagée pour la détection automatisée de ressources spécifiques. Le domaine de la médecine permet d'imaginer de nombreuses futures applications comme par exemple une population de nano-robots chargés d'administrer in vivo un médicament sur une tumeur ou une région précise de l'organisme, ou encore la simulation et une meilleure compréhension de la propagation d'une infection virale ou bactérienne chez un individus ou une chez une espèce. L'étude de la météo ou du climat est également un domaine possiblement perfectible par l'apport des systèmes multi-agents, de même que le tutorat pour le développement par exemple de plusieurs tuteurs virtuels étant en relation pour permettre le suivi adapté d'un apprenant dans plusieurs domaines de compétences.

J'espère, à travers ce mémoire, avoir pu aider à la compréhension de ce type de système et donner l'envie d'aller plus loin dans ce domaine qui a beaucoup d'avenir, notamment grâce au développement en parallèle dans le monde des technologies robotiques et cybernétiques. Je pense que c'est à chacun d'apporter une nouvelle pierre à l'édifice qu'est l'intelligence artificielle en général et les systèmes multi-agents en particulier, selon ses compétences propres.

BIBLIOGRAPHIE

2APL [en ligne] http://apapl.sourceforge.net/ (dernière consultation, juillet 2012).

ACT-R [en ligne] http://act-r.psy.cmu.edu/ (dernière consultation, juillet 2012).

Anderson, J. R. et Bower, G. H. 1973. *Human associative memory*. Washington : Winston.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C. et Qin, Y. 2004. *An Integrated Theory of the Mind.* [en ligne] http://act-r.psy.cmu.edu/papers/403/IntegratedTheory.pdf, (dernière consultation, juillet 2012)

Baars, B. J. 1997. *In the Theater of Consciousness. The Workspace of the Mind.* New York, Oxford University Press.

Bergeret, M. 2007. *Analyse d'architectures d'agent hybride pour la modélisation de comportement d'avatars*, à l'Université de Pau, France, p. 10-12.

Boissier, O. 2001. [en ligne] http://www.emse.fr/~boissier/enseignement/sma01/pdf/environnement.pdf (dernière consultation, juillet 2012).

Bothell, D. *ACT-R 6.0 Reference Manual*, [en ligne] http://act-r.psy.cmu.edu/actr6/reference-manual.pdf (dernière consultation, juillet 2012).

Bratman, M. 1987. *Intention, Plans, and Practical Reason*. Harvard University Press.

Broersen, J., Dastani, M., Huang, Z., Hulstijn, J. et Torre, L. 2001. *An alternative classification of agent types based on BOID conflict resolution*, Asterdam : Utrecht University.

Brooks, R. A. 1991. *Intelligence without reason*, à Proceedings of 12th International Joint Conference on Artificial Intelligence, p.569-595.

Brooks, R. A. 1991. *Intelligence without representation*, Artificial Intelligence Journal n°47, p.139-159.

Brooks, R. A. 1986. *A robust layered control system for a mobile robot*, IEEE Journal of Robotics and Automation.

Cohen, P. R., et Levesque, H. J. 1995. « Communicative Actions for Artificial Intelligence ». First International Conference on Multi-Agents Systems (ICMAS'95), San Francisco : The MIT Press.

Dubois, D., Nkambou, R., Quintal, J.-F. et Savard, F. 2010. « Decision-Making in Cognitive Tutoring Systems », *Advances in intelligent tutoring systems*, Spinger.

Ferber, J. 1995. *Les systèmes multi-agents, vers une intelligence collective*, France : InterEditions.

Ferber, J. et Gutknecht, O. 2000. *MadKit: a generic multi-agent platform*, article présenté à Proceedings of the 4th International Conference on Autonomous Agents, Barcelone, Espagne.

Ferber, J. et Mansour. S. 2007 *AGRS : un modèle organisationnel pour les systèmes multi-agents ouverts*. LIRMM.

Ferguson, I. A. 1992. *Touring Machines: Autonomous Agents with Attitudes*, IEEE Computeur Society Press vol.25, p.51-55.

Ferguson, I. A. 1992. *Touring Machines: an architecture for dynamic, rational, mobile agents*, Tech. Rep. n°73, University of Cambridge Computer Laboratory.

Ferguson, I. A. 1994. *Autonomous Agent Control: a Case for Integrating Models and Behaviors*, AAAI Technical Report FS-94-03.

FIPA [en ligne] http://www.fipa.org/ (dernière consultation, juillet 2012).

Franklin, S., Kelemen, A. et McCauley, L. 1998. *IDA: A cognitive Agent Architecture*, IEEE Conference on Systems, Man and Cybernetics, p. 2646-2651, IEEE Press.

Georgeff, M. P. et Rao, A. S. 1991. *Modeling Rational Agents within a BDI Architecture*, San Mateo (USA) : Morgan Kaufmann publishers Inc.

Georgeff, M. P. et Rao, A. S. 1995. *BDI Agents: From Theory to Practice*. Tech. Rep. n°56, Australian Artificial Intelligence Institute, Melbourne, Australia.

Gleick, J. 1989. *La Théorie du chaos*. Paris : Flammarion.

Gurvitch, G. 1963. *La vocation actuelle de la sociologie*, Paris : PUF.

Hoc, J. M. 1996. *Supervision et contrôle de processus : la cognition en situation dynamique*, Grenoble : PUG.

jACT-R [en ligne] http://jactr.org/ (dernière consultation, juillet 2012).

JADE [en ligne] http://jade.tilab.com/ (dernière consultation, juillet 2012).

JADEX [en ligne] http://jadex.informatik.uni-hamburg.de/xwiki/bin/view/About/Overview (dernière consultation, juillet 2012).

Jason [en ligne] http://jason.sourceforge.net/ (dernière consultation, juillet 2012).

Jennings, N. R., et Wooldridge, M. 1995. « Intelligent Agent: Theory and practice », *The Knowledge Engineering Review 10*.

Lestel, D., Drogoul, A., Grison, B. 1994. *Les agents réactifs et le vivant dans une perspective d'évolution coopérative*. Intellectica n°19, p.73-90.

MaDKit, Ferber, J. et Mansour, S. [en ligne] http://www.madkit.org/ (dernière consultation, juillet 2012).

Magnin, L. 1996. *Modélisation et simulation de l'environnement dans les systèmes multi-agents*, thèse.

Malone, T. W. 1988. *What is coordination theory*, à National Science Foundation Coordination Theory Workshop, MIT.

MetateM, Hepple, A. [en ligne] http://www.csc.liv.ac.uk/~anthony/metatem.html (dernière consultation, juillet 2012).

Morin, E. 1977. *La Méthode : la Nature de la Nature*, Paris : Le Seuil.

Müller, J. P. et Pischel, M. 1993. *The Agent Architecture InteRRaP: Concept and Application*, Deutsches Forschungszentrum für Künstliche Intelligenz.

Nkambou, R., Belghith, K., Kabanza, F. et Khan, M. 2005. *Supporting Training on Canadarm Simulator using a Flexible Path Planner*, à Proceedings of the 12th International Conference on Artificial Intelligence in Education (AIED), Asterdam : IOS Press.

Quinqueton, J. 2004 *Organisation des systèmes multi-agents*, présentation au LIRMM, Montpellier, France.

Richard, N. 2004. *Une approche située, coopérative et décentralisée pour l'interprétation d'images cérébrales par RMN*, thèse, p.73-74.

Rolls, E. T. 1999. *The brain and emotion.*, Oxford University Press.

Russell, S., et Norvig, P. 2006. *Intelligence artificielle*, 2ème édition, France : Pearson.

Searle, J. 1969. *Speech Acts*, Cambridge University Press. *Les actes de langage*, France : Hermann 1972.

Torrance, M. et Viola, P. 1991. *The AGENT0 Manual*, [en ligne] ftp://db.stanford.edu/pub/cstr/reports/cs/tr/91/1389/CS-TR-91-1389.pdf dernière consultation, juillet 2012).

Weiss, G. 1999. *Multiagent Systems, a modern approach to distributed*, Cambridge, The MIT Press.

Wooldridge, M. 1999. « Intelligent Agent », *Multiagent Systems* de Weiss G., Cambridge, The MIT Press.

Wooldridge, M. 2002. *An Introduction to MultiAgent Systems*, Wiley and Sons.