Proposition d'un modèle d'agents hybrides basé sur la motivation naturelle

Fenintsoa Andriamasinoro

Institut de Recherche en Mathématiques et Informatique Appliquées

Université de La Réunion

Université de l'Océan Indien
Commission de l'Océan Indien
Union Européenne

Agence Universitaire de la Francophonie

Thèse de Doctorat

intitulée :

Proposition d'un modèle d'agents hybrides basé sur la motivation naturelle

Présentée par

Fenintsoa ANDRIAMASINORO

pour obtenir le Grade de Docteur de l'Université de La Réunion

Spécialité : Informatique

Soutenue publiquement le 28 Août 2003

Membres du jury :

Président :	Joël Quinqueton	Directeur de Recherche
Rapporteurs :	Bernard Moulin Vincent Chevrier	Professeur, Université Laval, Canada Québec Habilitation à Diriger des Recherches, Université H. Poincaré Nancy
Examinateur :	Rémy Courdier	Maître de Conférences, Université de la Réunion
Directeur de Thèse :	Pierre Marcenac	Professeur, Université de La Réunion

A ma famille,

mon soutien moral sans faille, malgré la distance✍

Résumé

La thématique principale des recherches présentées dans le présent travail concerne la modélisation de comportement d'agents informatiques hybrides au sein de Systèmes Multi-Agents, i.e. d'agents qui disposent à la fois de comportements réactifs et de comportements cognitifs. S'inspirant, à la base, d'une réflexion principalement psychophysiologique du vivant, ce travail s'insère dans la problématique plus vaste de l'Intelligence Artificielle Distribuée et de la Vie Artificielle.

Partant du principe qu'un être humain, même s'il est cognitif, est aussi une espèce issue de la *nature*, le présent travail a pour objectif de mettre en place une architecture hybride générique d'agents artificiels dont la base du comportement est la motivation naturelle existant dans ces agents. Notre étude se particularise par l'intégration des concepts d'instinct, pulsion, faim, soif, etc. dans les agents cognitifs ou hybrides alors que dans la plupart des cas, ces notions ont toujours été l'apanage des modèles d'agents réactifs, particulièrement ceux basés sur les animats. A terme, le raisonnement et les attitudes mentales ne seront donc plus les seuls facteurs à prendre en compte pour comprendre le comportement de ces agents cognitifs ou hybrides. Ainsi :

- au niveau de l'architecture, nous obtenons un modèle générique basé uniformément sur le concept de motivation naturelle et ce, quel que soit le type de l'agent étudié (réactif/cognitif/hybride).
- au niveau du comportement, nous proposons de fournir un cadre hybride permettant à l'agent de gérer, à un niveau générique, la sélection d'actions et plus précisément pour le présent contexte, la sélection de motivations. Le terme " générique " signifie que l'ensemble des critères fournis dans le mécanisme de sélection ne dépend pas de l'application et permettra à l'utilisateur du système de ne se préoccuper qu'au minimum des stratégies

de sélection. Pour ce faire, il appartient au modèle de fournir les règles de positionnement pour chaque motivation.

Sur le plan de la conception, nous adoptons un modèle issu d'un travail déjà existant dans le monde de la psychologie. Il s'agit de la pyramide du psychologue américain Abraham Maslow, portant sur les cinq besoins abstraits des humains[1].

Notre cadre d'application concerne en premier lieu un cas d'école axé sur des agents fourrageurs qui se déplacent pour transporter des objets. Puis, dans un cadre plus réel, l'étude porte sur la simulation du comportement des paysans à Mangatany, une région au centre de Madagascar.

[1] Cette pyramide existe depuis des décennies, Abraham Maslow, son auteur étant décédé en 1970. Cependant, le modèle qu'il a créé est encore aujourd'hui enseigné dans le monde, notamment dans les filières de la psychologie et dans les écoles de gestion et de commerce. C'est dans ce dernier domaine que le psychologue a souvent appliqué ses recherches.

Remerciements

Je te rends grâces Mon Dieu
de m'avoir soutenu tout au long de cette thèse difficile,
et de m'avoir dirigé vers les personnes ci-dessous pour m'aider à la mener à bonne fin.

Mes remerciements iront tout d'abord à Pierre Gigord, ancien Directeur de l'IREMIA, à qui je suis redevable de bien des choses. Il m'a non seulement accueilli dans son laboratoire mais il a par la suite beaucoup investi (c'est bien le mot) pour que j'obtienne un minimum de financement pour ce travail. J'espère avoir mérité sa confiance et je souhaite que cet ouvrage comblera une partie de mon immense dette à son égard.

Mohamed Rochdi, son successeur, a pris la relève en soutenant au maximum toute démarche réalisée en vue de la continuité de mon financement. De plus, il a énormément contribué à la préparation des projets que j'ai envisagés pour l'après thèse. Qu'il reçoive au rappel de ces quelques faits non exhaustifs l'expression de ma profonde reconnaissance.

J'aimerais exprimer toute ma gratitude à Pierre Marcenac et Rémy Courdier qui m'ont fait confiance en acceptant d'être les encadrants de ce travail. Cette thèse n'aurait sans doute pu voir le jour sans leur soutien constant, leur patience (souvent mise à rude épreuve) et, bien sûr, les idées sur lesquelles nous nous sommes débattus pendant quatre ans.

Merci à Bernard Moulin de m'avoir fait l'honneur de rapporter cette thèse. Bien plus qu'un simple rapporteur, il a suivi l'évolution de cette recherche alors que celle-ci n'était encore que dans une phase intermédiaire. Il est certain que les critiques constructives qu'il a émises durant les moments de discussion que nous avons eus ont beaucoup contribué à l'aboutissement de ce travail.

J'adresse également ma reconnaissance à Vincent Chevrier de m'avoir consacré une partie de son précieux temps pour effectuer un rapport sur cette thèse. Je sais combien il avait un programme bien chargé. Néanmoins, il a accepté d'être mon rapporteur et cela m'a profondément touché.

Joël Quinqueton a aussi accepté de faire partie de mon jury. Connaissant ses qualités, je suis particulièrement honoré qu'il figure parmi les personnes qui valident ce travail.

Le *nous* employé tout au long de la thèse n'est pas qu'académique. Il englobe dans son anonymat les personnes avec lesquelles j'ai pu travailler pour réaliser les travaux présentés. Il serait sans doute trop long d'en faire la liste exhaustive, mais je tiens à remercier particulièrement :

- Jean Pierre Müller et Jean Pierre Briot à qui j'ai eu la chance de présenter mon approche de modélisation pour bénéficier de leurs précieux conseils. Leurs remarques m'ont fait comprendre bien d'éléments techniques que je n'ai pu découvrir par moi-même durant cette thèse.
- Stéphane Calderoni qui a bien voulu me fournir sa plate-forme agent et ce, jusqu'à son code source, ce qui m'a considérablement permis d'obtenir un réel support technique. Sans son aide, l'implémentation du modèle que j'ai proposé dans ce travail ne serait certainement aujourd'hui qu'à un état encore embryonnaire.
- Fanilo Harivelo, mon aide-développeur qui, après sa journée quotidienne de travail, se met encore à veiller jusqu'à deux heures du matin pour que l'interface graphique de mon simulateur agent fonctionne. J'ai pu apprécier à quel point il avait le souci du travail bien fait et je peux dire que si j'ai pu aujourd'hui expérimenter mon modèle, c'est en grande partie grâce à lui.
- François Guerrin qui s'est dévoué à la relecture de cette thèse, en y apportant en sus ses précieuses connaissances de la modélisation en général, et du monde des agents en particulier. Malgré beaucoup de travail en attente, étant de retour de vacances, il n'a ainsi pas hésité à accorder son attention au mien. Je reconnais ici, à la fois le technicien et l'ami.

Sur le plan financier, la présente recherche a été *partiellement* supportée par l'Université de l'Océan Indien (UOI), une institution instaurée dans le cadre de la coopération " Commission de l'Océan Indien / Union Européenne ". Je remercie Mme Masséande Allaoui, Directrice de l'UOI, et toute son équipe, ainsi que l'Ecole Nationale d'Informatique

de Fianarantsoa (Madagascar) qui a aussi contribué administrativement à l'obtention de cette bourse. Même avec les conditions logistiques et les relations humaines que j'ai pu avoir, je n'aurai sans doute jamais pu entamer mes recherches sans ce financement, si partiel soit-il.

Merci également à l'Association Universitaire Francophone (AUF) qui a financé ma mission de Recherche à Madagascar en 2002, ainsi qu'aux membres de l'équipe malgache avec laquelle j'ai collaboré : Simone Ratsivalaka, Fara Lala Razafy, Léa Raharijaona, Solofo Rakotondraompiana et Lova Tahina Randrianarison.

Avant de terminer, je tiens à faire savoir aux personnes ci-dessous combien précieuse a été leur présence et à quel point je leur sais gré de ne pas s'être trop plaint de l'être asocial que je suis devenu lors de cette période de thèse. Je pense à Stéphanie Ratianarimanana (dit " Joja "), à Marie-Andrée Mariapoullée, aux époux Samuel et Mélanie Andrieux, ainsi qu'à Michel Bonnabeau et son épouse. Bien plus que des amis, ils ont été ma famille dans ce pays loin du mien.

Enfin, je tiens à remercier toute ma famille pour l'affection et pour la patience qu'elle m'a accordées depuis le début de mes études, et en particulier mes parents, mon frère et mes sœurs, qui m'ont toujours soutenu. Que ce document vous soit dédié pour l'effort qui ne fut pas le mien, mais le nôtre.

Table des matières

Introduction générale

Préambule : l'homme et l'animal

Depuis fort longtemps, comprendre le vivant a toujours été l'axe central des chercheurs dans le domaine de la psychologie et de l'éthologie. Et aussi bien dans la psychologie humaine/animale (Buytendijk 1952) que dans l'éthologie animale/humaine (Feyreisen et De Lannoy 1987, Renck et Servais 2002), un des paramètres d'études concerne le comportement.

Les scientifiques s'accordent à dire que dans sa conception, l'homme est issu de la biologie animale[1]. La partie " reptilienne " de son cerveau et de sa biologie fonctionne sur cette mémoire de l'animal. En outre, l'animal inconscient en l'homme est garant de la survie de son espèce. Le point *commun* entre l'homme et l'animal est donc cette présence d'un ensemble de besoins *naturels* liés à sa survie[2] (faim, soif, sexualité, défense contre des attaques extérieures, ...) dont la satisfaction constitue à la base leur principale *motivation* dans leur comportement.

Nous reconnaissons certes, que si ces besoins naturels sont communs aux hommes et aux animaux, la manière de les satisfaire n'est pas la même dans les deux cas : comme l'animal, l'homme doit manger et se reproduire, mais il ne mange ni se reproduit tout à fait comme l'animal. L'homme n'accepte de satisfaire ses besoins physiques que si certaines conditions d'un autre ordre (esthétiques, morales) comme le savoir-vivre, la politesse, etc. sont remplies (Picard 1998). Prenons un exemple simple donné par Andriamasinoro

---

[1] Interview de Jean-Philippe Brebion, un kinésithérapeute de formation qui a orienté son travail sur la compréhension de l'équilibre énergétique du corps humain.

[2] En fonction de sa structure génétique, chaque animal a sans doute ses propres manières de s'organiser et de gérer sa survie. Mais quelles que soient ces manières, les motivations à la base sont les mêmes et c'est ce qui nous intéresse ici.

et Courdier (2002b) : lorsqu'un animal affamé trouve de la nourriture, il la consomme immédiatement[3]. Cette action de l'animal est dite de *bas niveau*, c'est-à-dire liée à la satisfaction immédiate des instincts initiaux. Par contre, l'homme, malgré une faim aussi intense que celle de l'animal, se doit (dans la mesure du possible) de tenir compte de certaines règles ou normes sociales " préliminaires " telles que l'achat de la nourriture en question pour en acquérir les droits de propriétés, l'utilisation, le cas échéant, d'une serviette de table avant de consommer, etc. Néanmoins, nous pouvons ainsi résumer que ce qui motive l'homme et l'animal à la base, c'est toujours la satisfaction de leurs besoins *naturels*. Même si leur moyen de les satisfaire est différent, cette *source initiale commune de comportement* existe. C'est cette source commune basée sur le naturel que nous appelons " motivation naturelle " et *la motivation naturelle d'une entité vivante, c'est la satisfaction de ses besoins naturels.*

Si tel est le cas dans le monde du vivant, qu'en est-il de la modélisation agent du contexte ?

Modélisation agent et problématique

Dans de nombreux travaux réalisés dans la discipline Systèmes Multi-Agents (désormais abrégé en SMA dans ce document), l'étude du comportement, et en particulier celui des vivants a toujours pris une place prépondérante. A cet effet, cette discipline s'est toujours rapprochée de l'éthologie ou de la psychologie. L'existence de bon nombre de modèles éthologiques (Bonabeau et Theraulaz 1994, Drogoul et Ferber 1994, Quinqueton et Hamadi 1999, etc.) et de modèles en relation avec la connaissance humaine (Rao et Georgeff 1992, Erceau et al. 1994, Lhuillier 1998, Panzarasa et al. 1999, etc.) en est une preuve irréfutable.

Néanmoins, nous pensons qu'à l'instar de cette existence d'un ensemble commun de motivations naturelles chez l'homme et l'animal, toutes ces catégories de modèles d'agents mimant le vivant, que ces agents soient rationnels ou non, devraient aussi pouvoir converger vers une structure commune basée sur ce concept de motivation naturelle[4]. L'idée

[3] Le terme " immédiatement " ne signifie pas ici " dans l'instant qui suit ", mais plutôt " sans ces conditions d'un autre ordre " mentionnées plus haut.

[4] Comme le disait le psychologue Daco (1965), l'homme (souvent modélisé ici comme agent cognitif) qui ne vit que par sa raison n'est qu'une moitié d'homme. Cette affirmation visait à mettre davantage en évidence la partie instinctive de l'homme même si celui-ci est aussi rationnel.

générale serait alors de mettre en place un modèle générique d'agents simulant ce fonctionnement du vivant, agents connus sous le terme *d'agents artificiels*[5] et possédant ce composant commun de base.

La présente thèse constitue un effort vers cet objectif. Le cadre général de notre travail concerne les agents *hybrides* (Wooldridge et Jennings 1995, Wooldridge 2002) c'est-à-dire, des agents dotés de capacités substantielles de raisonnement et de capacités réactives. Cependant, au vu de la définition de l'hybridisme ci-dessus, notre recherche aurait alors tendance à se restreindre uniquement aux agents ayant *toujours à la fois* ces deux capacités. Aussi, pour avoir une vue plus large de nos ambitions, il sera aussi intéressant de couvrir également les agents cognitifs et les agents réactifs *individuellement*. En somme, le terme " hybride " est ici générique pour pouvoir qualifier, soit un agent réactif, soit un agent cognitif, soit un agent qui est à la fois réactif et cognitif. La page 11 de ce document situe plus précisément l'utilisation de ces termes par rapport à cette thèse.

La problématique générale de cette thèse se résume alors à la question suivante : " comment devra-t-on mettre en œuvre un modèle générique de motivations exploitable génériquement à un niveau hybride ? ". Pour pouvoir répondre à cette question, nous devrons en premier lieu analyser la manière dont l'aspect générique de *l'architecture* de notre agent sera mis en œuvre. La contrainte inhérente à ce point est que le modèle doit toujours considérer le concept de motivations naturelles tout en pouvant s'adapter, au niveau applicatif, à des agents réactifs ou à des agents cognitifs. En outre, comme nous sommes dans le milieu hybride, il faudra également considérer la gestion d'une problématique propre à cette notion : le basculement entre le comportement réactif et cognitif de l'agent.

Au-delà de cet aspect " architecture ", il y a l'aspect " comportement ". Elle concerne particulièrement la recherche des critères à utiliser lors du processus de sélection des motivations introduites dans l'application. Cependant, pour qu'ils puissent s'appliquer à tous les types d'agents, ces critères, de même que le moteur de sélection associé, doivent être *génériques*. Ce cadre générique du processus de sélection doit pouvoir montrer que dans le cadre applicatif, les travaux des utilisateurs du système lors de l'attribution de règles de comportement devraient être réduits au minimum, tout en ayant des résultats qui respectent le comportement observé dans la réalité. La difficulté fréquente inhérente

[5] Ce terme fait référence à l'Intelligence Artificielle et la Vie artificielle (Drogoul 2000).

à ce type de problème se situe notamment au moment où l'agent se trouve face à de nombreuses motivations. Il faut alors que notre modèle arrive à offrir un ensemble de règles génériques qui permettront à l'agent de prioriser une motivation plutôt qu'une autre. La motivation à traiter en priorité varie en fonction du temps. Une motivation peut être importante à l'instant `t-2`, puis ne plus l'être à l'instant `t-1`, et l'être de nouveau à l'instant `t`, etc. De ce fait, il faut introduire un processus de contrôle répétitif qui va déterminer en permanence la prochaine motivation considérée comme la plus importante.

Organisation de la thèse

Dans cette rédaction qui nécessite une importante reconstruction du cheminement intellectuel effectué durant nos recherches, nous nous efforçons de préserver la forte interaction qui a existé depuis le début entre les deux grands axes exposés qui ont fait naître l'hybridisme : le réactif et le cognitif. Ce document de thèse est ainsi rédigé de façon à montrer au lecteur la progression des idées proposées dans la résolution de la problématique précédemment énoncée. Dans cet objectif, nous avons organisé notre rédaction en 7 Chapitres.

Le chapitre I est un chapitre d'introduction. Il englobe tout ce qui est initialement indispensable de savoir : le contexte général de notre travail, la méthodologie théorique et expérimentale adoptées, ainsi que la définition des concepts informatiques manipulés par la suite. Sa lecture constitue à ce titre un préalable à la compréhension générale de la thèse.

A partir du chapitre II, nous entrons dans les détails du sujet. Ce chapitre présente particulièrement l'état de l'art concernant la modélisation de la motivation dans les modèles de comportement actuels. En respectant la méthodologie énoncée dans le chapitre I, nous parcourons ainsi successivement les modèles d'agents réactifs, d'agents cognitifs et d'agents hybrides. Ce parcours se termine à chaque fois par des analyses concrètes de chaque type de modèle.

À l'issue des critiques relevées de l'analyse du Chapitre II à propos des motivations, nous introduisons au Chapitre III le modèle Maslow que nous proposons dans cette thèse. Le modèle initial adopté pour l'élaboration de notre proposition est la pyramide

des besoins[6] conçue par le psychologue américain Abraham Maslow. Nous montrons dans ce chapitre comment ce modèle issu du monde psychologique est étendu et adapté en modèle agent utilisable dans un contexte hybride.

Le chapitre IV a pour but principal de formaliser tous les concepts manipulés par le modèle Maslow. Ces concepts sont ensuite repris et utilisés dans le chapitre V dont l'objectif est précisément de montrer le fonctionnement et les particularités du modèle générique hybride conçu. Le modèle, rappelons-le, est basé sur le concept de motivation naturelle.

Dans le chapitre VI, nous expérimentons notre modèle au travers de deux cadres d'applications, à savoir celui de robots fourrageurs (cas d'école) et de paysans qui cultivent la rizière (cas réel).

Enfin, avant de conclure la thèse, nous discutons, au chapitre VII, de la contribution de ce travail de thèse dans le domaine des SMA.

[6] La question que l'on pourrait se poser ici est : pourquoi traiter une pyramide des *besoins* alors que nous parlons de *motivation* ? En fait, bien que le terme de " motivation " ait été choisi comme principal sujet de travail dans la mesure où il est directement lié à la notion de comportement, ces deux termes peuvent être alternativement employés en fonction des explications.

Chapitre I

Cadre des travaux et méthodologie

Le présent chapitre a pour but de situer le cadre dans lequel cette thèse est menée. Tout d'abord, la section I.1 rappelle brièvement le contexte général du travail. La section I.2 décrit par la suite les précisions initiales à apporter (positionnement de recherche, terminologie adoptée, ...) par rapport au sujet à traiter. Enfin, après avoir donné un aperçu de notre démarche bibliographique (Section I.3), nous présentons le domaine d'expérimentation de cette thèse (Section I.4).

I. 1 Historique des travaux

Le cadre général de ce travail est celui du projet MAS[2] de l'IREMIA[(1)] à l'Université de La Réunion, dont l'objectif est de proposer des outils pour la modélisation distribuée de systèmes. De par l'environnement scientifique local, la problématique de l'équipe s'est initialement inscrite dans la prévision de catastrophes naturelles, comme par exemple celles liées au Volcan de la Fournaise (Lahaie et al. 1996), etc.

[1] Mas[2] est l'acronyme de Multi-Agent System Modelling And Simulation.

I. 1. 1 Projet initial : Geamas / Biomas

La mise en œuvre d'applications aussi utilisables par des experts quel que soit leur environnement informatique nous a guidés vers un développement en Java d'une plate-forme agent appelée *Geamas*[2] (Marcenac et Giroux 1998). Le projet d'application sur lequel nous nous sommes initialement investis avec Geamas est un problème concernant la gestion de la biomasse agricole à l'Ile de La Réunion où l'on observe, dans certaines zones d'élevage intensif, une concentration d'exploitations fortement excédentaires ou déficitaires en matière organique. Nous avons réalisé un prototype appelé Biomas (Guerrin et al. 1998), (Andriamasinoro 1999), (Courdier, Guerrin, Andriamasinoro et Paillat 2002). Biomas est un modèle visant à représenter les pratiques de gestion des effluents d'élevage à l'échelle de territoires, afin de simuler les transferts possibles de matières entre exploitations, et tester ainsi les alternatives d'organisation des exploitants (éleveurs, cultivateurs, transporteurs). Ces travaux ont été menés avec l'équipe GDOR (Gestion des Déchets ORganiques) du CIRAD de La Réunion.

I. 1. 2 De Geamas à ADK

La plate-forme Geamas renferme cependant certaines limites, notamment au niveau du mécanisme de la sélection d'actions. Tout d'abord, un agent Geamas a été théoriquement conçu pour être hybride (Marcenac 1997), c'est-à-dire, rappelons-le, doté de capacités substantielles de raisonnement et de capacités réactives. Cependant, au niveau du noyau, il n'existe pas de règles génériques mettant en œuvre les motivations réelles des agents à effectuer des actions. Le modèle actuel n'est constitué que d'un squelette à partir duquel il appartiendra à la partie applicative de devoir apporter la majeure partie des règles de comportement. Nous avons noté qu'il n'y a pas non plus de critères de sélection des messages ni de l'évaluation de leurs priorités (problème de sélection d'actions) c'est-à-dire que chaque information reçue est traitée séquentiellement jusqu'à son terme avant que l'information suivante soit prise en compte (Andriamasinoro et Courdier 2000). Même si deux événements ont lieu en même temps, il n'existe, au niveau du noyau, aucun indice générique pouvant guider à la prise de décision quant à la priorité de l'un ou l'autre de ces événements.

---

[2] Acronyme de GEneral Architecture for Multi-Agent System

Sur un autre plan, à savoir la modélisation de l'environnement, Geamas n'était prévu ni pour prendre en compte ni pour simuler des environnements artificiels issus du monde réel (Soulié 2001). En effet, lors des simulations précédentes construites sur la base de Geamas, aucun besoin de représentation de l'environnement ne s'était fait sentir. En fait, Geamas avait été défini comme un SMA purement communiquant et non comme un SMA situé. Ce qui pourrait présenter une certaine lacune, notamment dans la simulation des attractions/répulsions associées au comportement naturel que nous comptons mettre en œuvre dans ce travail.

De ce fait, pour ce travail, nous utilisons une autre plate-forme de développement à savoir ADK[3] (pour " Agent Developer Kit ") récemment réalisée par Calderoni (2002). Le critère du choix d'ADK s'est porté sur la gestion de ce problème d'environnement qui joue un rôle fondamental, notamment chez les agents réactifs. En effet, ADK est une plate-forme dont l'environnement physique permet la diffusion de signaux, aussi bien par les agents que par les objets. De plus, comme nous savons que développer une plate-forme intégrant des concepts propres aux réactifs demande des algorithmes mathématiques complexes (ex : le calcul de la navigation des robots ou encore celui de la propagation de signaux dans l'environnement), il nous semble astucieux d'utiliser ADK pour gérer cette partie puis de ne développer à notre niveau, que la partie informatique du comportement.

Précisons qu'en terme de mécanisme de sélection d'actions ou de motivations, ADK possède les mêmes limites que Geamas. Et comme notre objectif dans cette thèse est de contribuer à l'amélioration de ce mécanisme chez les agents, nous aurions pu, si l'on se référait uniquement à ce critère, choisir l'une ou l'autre de ces deux plates-formes pour expérimenter notre travail. Le choix d'ADK a donc été favorisé par l'existence de ces éléments plus riches au sein de l'environnement. En conséquence, notre contribution dans cette thèse concernera le mécanisme de sélection d'actions et de motivations de ADK

[3] Les spécifications techniques de ADK sont développées à la section V. 4. 1, au moment où nous l'utilisons réellement dans notre travail.

I. 2 Positionnement conceptuel de départ

Après l'énoncé du cadre général de notre travail, nous passons progressivement à partir de cette section à un aspect plus technique sur la présentation de cette thèse.

I. 2. 1 Positionnement thématique en SMA

Notre présente étude concerne les agents individuels, que ces derniers soient seuls ou intégrés dans un groupe. En voici les précisions supplémentaires :

- lorsque nous touchons le domaine du comportement de groupe, nous nous intéressons plutôt à la manière dont ces agents individuels se comportent socialement dans ce groupe. Nous ne sommes donc pas dans un contexte de collaboration à proprement parler.
- lorsque nous parlons d'objectif de groupe, nous parlons de la situation individuelle de l'agent par rapport à ce groupe.
- lorsque nous parlons d'agents, il s'agit soit de l'homme, soit des animaux. Cependant, il faut bien préciser que nous n'entrerons pas dans l'étude d'une espèce particulière d'animal. Si tel est le cas, c'est uniquement pour illustrer un certain comportement qu'il est possible de trouver chez les humains.

Le travail se situe à un niveau *microscopique* de la conception agent, c'est-à-dire centré sur l'agent, contrairement au niveau macroscopique, centré cette fois-ci sur les interactions. Il est néanmoins évident que le second niveau ne peut être ignoré car comme le soulignait Erceau et al. (1994), la théorie SMA privilégie les aspects interactionnels, contrairement à l'Intelligence Artificielle (IA) dite " classique " qui, elle, privilégie la modélisation du mode de raisonnement.

Du point de vue architecture, nous nous intéressons à l'étude des agents *situés* c'est-à-dire des agents qui possèdent une position physique dans l'environnement (Drogoul 2000). Le fait que notre agent soit situé est important pour notre démarche. En effet, les études sur le comportement animal ont montré le rôle essentiel de l'environnement pour la communication (Bonabeau et Theraulaz 1994, Quinqueton 1999). Lorsque l'on parle de communication, il ne s'agit pas seulement de communication entre agents via l'environnement comme dans le cas de la proie et du prédateur (De Jong 1997). Il peut

aussi s'agir de celle d'un agent avec un objet de cet environnement tel qu'un obstacle ou une nourriture. Dans notre travail, nous suivons également cette même lignée.

I. 2. 2 Terminologie agent

Le réactif et le cognitif

La distinction agent *cognitif/réactif* existe mais semble parfois trop simpliste (Ferber 1997) d'autant qu'elle diffère selon le point de vue adopté et la discipline qui l'établit (Bonabeau et Theraulaz 1994). Un agent qui adapte son *plan* en temps réel par rapport à un changement dynamique de l'environnement est par exemple un agent réactif (Mavromichalis et Vouros, 2000) alors que la notion de " plan ", généralement utilisée dans un cadre cognitif, y est bien mentionnée. Aussi, selon Ferber, l'étude devra plutôt s'effectuer par la recherche de la relation entre l'agent et son environnement en essayant de savoir si la représentation du monde par l'agent se situe à un niveau sub-symbolique, c'est-à-dire intégré dans ses capacités sensori-motrices (dans ce cas, l'agent est réactif) ou, au contraire, si l'agent dispose d'une représentation symbolique et explicite de ce monde à partir de laquelle il peut raisonner (agent cognitif).

Dans le cadre de la Vie Artificielle, on distingue la *tendance* suivante : les animaux sont le plus souvent modélisés comme des agents réactifs, vu leur comportement à base d'instinct. Pour preuve, les modèles réactifs (Drogoul et Ferber 1994, Gershenson et al. 2000) mettent davantage en évidence cette notion d'instinct (fuite face à un prédateur, faim, ...). Par contre, les humains sont davantage considérés comme des agents cognitifs étant donné l'étude plus poussée de leur capacité à raisonner que de leur partie instinctive. Les modèles cognitifs (ex : Panzarasa et al. 1999) portent plus sur la modélisation des connaissances.

Pour la thèse, nous combinons les spécifications énoncées par les deux paragraphes ci-dessus. Ainsi, nous mettons dans la catégorie " réactif " les modèles simulant des animaux et disposant de la représentation du monde à partir de ses capacités sensori-motrices, en sachant qu'ils peuvent *éventuellement* disposer de capacité cognitive[4]. La symétrie

[4] Cette éventuelle partie cognitive est introduite uniquement par respect de la limite " floue " qui existe entre le réactif et le cognitif. Cela ne signifie pas qu'elle est considérée lors de l'analyse du modèle en question. Cette présence simultanée du réactif et du cognitif est étudiée dans le cadre des agents hybrides (Section II.4).

de cette catégorisation est valable pour les agents cognitifs (qui sont donc, pour notre étude, les humains).

Comme l'hybridisme est la " somme " du réactif et du cognitif, un agent hybride dispose ainsi des deux capacités de représentation ci-dessus. Cependant, dans la prise en compte du réactif dans l'hybridisme, nous allons également prendre en compte des cas de réactivité autre que ceux des animats.

Le social

Ce terme peut être interprété de différentes manières. Il est par exemple utilisé dans le domaine animal : une espèce est dite *sociale* si la rencontre entre plusieurs individus dans un espace donné est un événement régulier et non fortuit, c'est à dire si ces individus vivent durablement en couple, en famille, en groupes structurés. Ces groupes se caractérisent par l'existence d'une hiérarchie sociale entre leurs membres (Immelmann 1990). Le social est ici utilisé dans un cadre éthologique et les agents sociaux sont ceux qui sont issus de telles études, comme par exemple les fourmis artificielles (Drogoul 1993).

Une autre définition d'un agent social est donnée par Moulin et Chaib-Draa (1996). Elle concerne les agents *intentionnels*, c'est-à-dire des agents qui sont capables de générer des plans, de raisonner en fonction de ces intentions et de ses croyances courantes, etc. Un agent social est un agent intentionnel qui possède en outre une représentation explicite des autres agents. Ainsi, il doit pouvoir raisonner et prendre ses décisions en tenant également compte de cette connaissance des autres agents. Nous sommes ici dans un cadre totalement cognitif.

Le social que nous étudions présentement tend plus vers la première définition que vers la seconde. Le besoin social correspond ici au besoin naturel psychologique d'un agent social de s'intégrer dans une société. En fait, que celui-ci dispose ou non d'une représentation des autres agents, son besoin premier consiste d'abord à satisfaire cette intégration. Cependant, la deuxième définition peut constituer un moyen (mais non l'unique) pour arriver à la satisfaction des besoins spécifiés dans la première. Nous disons bien " un des moyens " car il existe des agents réactifs qui arrivent à s'intégrer dans un groupe

sans pour autant avoir une représentation des autres agents ni une anticipation sur leur comportement.

En somme, le fait que le social corresponde à la seconde définition sous-entend implicitement que la première définition est déjà vérifiée.

I. 2. 3 Autre terminologie

Définissons à présent trois notions très liées entre elles et qui sont introduites dans notre conception : *le naturel*, *l'instinct* et *l'homéostasie*. Si l'analyse du terme " naturel " s'avère logique pour la présente thèse, il faut par contre se poser une limite de recherche pour " instinct " et pour " homéostasie ", c'est-à-dire qu'il faut rester uniquement sur l'acquisition des connaissances indispensables pour le travail.

Le naturel

Le domaine de définition du " naturel " est très vaste car ce terme est utilisé dans bon nombre de contextes[5].

Dans le monde physique, on utilise par exemple ce terme pour désigner tout objet ou tout événement n'étant pas le produit du travail de l'homme mais préalablement fourni par la nature (par opposition à " artificiel "). Le " naturel " est aussi prononcé par l'homme lorsque les événements qu'il perçoit sont conformes à l'ordre normal des choses (contrairement à " surnaturel "), c'est-à-dire que leur déroulement est explicable, soit via les lois qu'il connaît sur le fonctionnement de la nature, soit par l'intermédiaire de processus de raisonnement, et à l'issu desquels il trouve un résultat logique sur le phénomène observé.

Les définitions énoncées ci-dessus se rapportent à tout ce qui est *extérieur* à l'homme, c'est-à-dire à tout ce qu'il perçoit. Cependant, le terme " naturel " s'applique aussi dans la qualification des éléments qui sont intérieurs au vivant. Une personne possède par exemple des caractéristiques naturelles lorsque celles-ci sont présentes chez elle dès sa naissance. Cette intégration peut être héritée génétiquement des parents, ou encore

[5] Nous restons cependant dans le cadre des êtres vivants et son environnement, ce qui est exclut d'emblée les notions comme les nombres naturels issus des mathématiques.

est due à des raisons d'ordre anatomique. De ce fait, certaines caractéristiques qui sont naturelles pour une entité ne le sont pas pour d'autres. C'est ce qui différencie en général un animal (ex : naturellement agressif) d'un autre (naturellement inoffensif). En somme, nous sommes ici dans une discussion terminologique du " naturel " dans laquelle ce dernier se présente à l'intérieur d'un être d'une manière *individuelle* ou tout au plus semi-collective (i.e. valables pour certains groupes d'espèces seulement mais pas pour tout le monde).

Enfin, nous avons les éléments naturels qui sont *communs* à tout être vivant, et présents dans son physique, indépendamment de sa structure génétique et de son environnement d'évolution, etc. Ce sont principalement les besoins liés à la survie de ces êtres, indépendamment du temps et de l'espace. Un des exemples classiques liés à cette dernière catégorie est la manifestation de l'instinct naturel de préservation que nous reprenons plus loin.

Pour résumer, nous dégageons dans la présente discussion, une double classification " intérieur/extérieur " et " individuel/collectif " du naturel.

Par rapport à ce présent travail, c'est cette dernière définition relative au " collectif " que nous nous proposons d'étudier et ce, que les éléments manipulés soit intérieurs ou extérieurs à l'entité concernée.

L'homéostasie

L'homéostasie[6] est la faculté qu'ont les êtres vivants à maintenir ou à rétablir certaines constantes physiologiques (concentration du sang et de la lymphe, pression artérielle, etc.) quelles que soient les variations du milieu extérieur. L'homéostasie désigne l'état de stabilité qui est assuré par le milieu intérieur, c'est à dire l'ensemble des liquides extracellulaires (sang, lymphe et liquide interstitiel). Ce milieu intérieur est à l'interface entre les zones d'échange avec le milieu extérieur (poumon, intestin et rein) et les cellules de l'organisme. Chez les animaux à sang chaud, l'homéostasie comprend l'ensemble des réactions de maintien de l'équilibre fonctionnel : maintien de la température interne, maintien de l'équilibre quantitatif et qualitatif des entrées et des sorties. Son activité est liée au comportement visant à la *satisfaction de besoins.*

[6] L'homéostasie vient du grec *homeos* qui signifie " même " et *statis* qui signifie " rester ".

Deux cas peuvent se produire dans le mode de fonctionnement de l'homéostasie :

1. elle travaille en interne : les ressources dont elle a besoin pour l'équilibre se trouvent dans le corps même (ex : équilibre de la pression artérielle) ;
2. elle a besoin d'une contribution extérieure, soit dans le cadre d'un approvisionnement en ressources (ex : demande d'alimentation), soit au contraire pour éviter que des ressources parviennent dans le corps car il y en existe en quantité trop élevée (ex : se mettre à l'ombre en raison d'une chaleur excessive).

Un des exemples classiques de l'homéostasie est celui de la faim qui, du point de vue social, dépend de mécanismes d'ordre sociologique qui tendent à devenir réflexes (horaire des repas) et qui sont eux-mêmes déclenchés par des stimuli comme la distension de l'estomac, la mastication, la salivation, la déglutition, le goût, etc. Ce phénomène rappelle la fameuse expérimentation appliquée sur les chiens par le physiologiste Russe Ivan Pavlov, rapportée par Todes (2001).

L'instinct

C'est le pendant de l'homéostasie au niveau psychologique. L'instinct est un comportement inné dans un être vivant. Que ce soit instinct de vie, instinct de fuite, instinct de reproduction. Dans telle ou telle situation, il agit ou réagit de manière spontanée, sans qu'aucun apprentissage préalable n'ait déterminé sa conduite. Il met ainsi en œuvre des conduites inscrites dans son patrimoine génétique, mais sur lesquelles influent les circonstances.

Lorsqu'il s'agit de comportements psychiques, on préfère à cette notion d'instinct celle de pulsion[7] aux composantes plus complexes. Barbato (2001) a estimé que depuis le XIXè siècle, les instincts sont globalement définis comme *une disposition innée qui tend, à travers un ensemble de comportements, à répondre à certaines nécessités liées à la survie, même si la créature n'est pas consciente de cet objectif.* Ainsi, un instinct est présent et répond au même but chez toutes les créatures vivantes, indépendamment

[7] La pulsion correspond généralement à la poussée d'énergie du corps vers le psychisme. Le corps connaît des excitations internes qui déclenchent des besoins impérieux et amènent un état de tension. Les pulsions communiquent ces besoins du corps au psychisme. Selon Freud (1923), toutes les pulsions ont quatre caractères communs : la source, la poussée, le but qui est la satisfaction du besoin pour apaiser l'état de tension, et l'objet, outil de cette satisfaction.

de leur niveau de conscience ou de développement sur l'échelle de l'évolution. De ce point de vue, Barbato affirme pouvoir distinguer trois pulsions principales conformes aux caractéristiques d'un instinct : une pulsion de reproduction (sexuelle), une pulsion à s'adapter ou à se relier à l'environnement (sociale) et une troisième pulsion qui concerne la survie individuelle (préservation).

L'instinct social

La satisfaction de cet instinct est importante pour les êtres sociaux. Par définition même, ces êtres ont besoin d'être intégrés dans une société et ce, quelles que soient leurs activités. En reprenant ce que disait le psychologue Daco (1965), il n'y a pire pour un être humain que d'être rejeté par la société. Les humains sont de par leur nature même des êtres sociaux. Etre accepté peut aller d'une simple vie auprès d'une même espèce que soi (c'est-à-dire ne pas se sentir seul), à un désir d'avoir des amis, voire en procédant à des actions explicites dont l'objectif (psychologique) initial est en réalité d'être ensemble (création d'une équipe, intégration, rassemblement, etc.).

Le comportement dans un cadre collectif (Camalot 2000) n'est qu'une concrétisation à plus haut niveau de ce désir de se rassembler. S'intégrer dans une association ou dans une équipe quelconque peut a priori avoir des objectifs précis (par exemple, gagner un tournoi pour une équipe de football). Mais il y a également le désir plus profond d'être accepté par ses co-équipiers. Une équipe ayant un objectif collectif donné se doit d'abord de consolider la cohésion du groupe et d'établir une bonne entente psychologique entre les membres. Lorsque cette condition initiale n'est pas satisfaite, les résultats à plus haut niveau sont compromis.

I. 2. 4 A propos de la motivation

Une question relative à ce concept est de savoir si la motivation que l'agent montre est authentique ou intentionnellement modifiée par lui. Supposons par exemple l'existence d'une action `A` exécutée par un agent `G` et soit une motivation `M` qui selon `G`, serait a

priori la source profonde de A. Est-ce vrai ou faux ? Au niveau de l'agent, plusieurs possibilités se dégagent :

1. G croit que M est la source de A *// ignorance ou connaissance*
2. G sait bien que M n'est pas la source de A mais il fait croire que tel est le cas *// dissimulation intentionnelle*
3. G considère M comme étant la source de A *// incertitude et justification par une hypothèse*
4. etc.

Ces possibilités ne permettent pas de répondre d'une manière certaine que M est effectivement la source de A. En fait, du point de vue psychologique (Daco 1965), la motivation réelle supposons M_x (venant de l'Inconscient) existe toujours mais il n'est pas forcément possible de savoir si c'est la même motivation que celle qui est perçue au niveau de la conscience. Aussi, comme les cas sont variés, il convient pour nous de fixer un point de départ. Nous supposons dans un premier temps, que les deux motivations M et M_x sont bien identiques[8]. C'est par la suite (dans un travail futur) que le filtre de *non sincérité* pourra être étudié.

I. 3 Démarche bibliographique

Etant donné le domaine hybride du sujet, notre étude bibliographique porte à la fois sur les modèles d'agents réactifs, d'agents cognitifs et d'agents hybrides. Il convient pour nous d'identifier la manière dont la motivation y est actuellement définie et modélisée.

L'observation des modèles réactifs s'avère indispensable dans la mesure où les animaux adoptent souvent ce type de comportement (Bonabeau et Theraulaz 1994). Dans cet objectif, nous analysons les architectures d'agents existants, et la manière dont le mécanisme de sélection d'actions est modélisé. Notre analyse est davantage focalisée sur

[8] Comme la compréhension de cette idée ne semble pas toujours être évidente, nous devons apporter au lecteur la précision suivante : psychologiquement (Daco 1965), les motivations initiales (et réelles) d'une personne sont envoyées par ce qui est appelé l'Inconscient Collectif. Pour plusieurs raisons que nous ne développons pas ici, ces motivations initiales peuvent être déviées par une couche supérieure du mental, nommée " Inconscient Personnel " ou encore cachées volontairement par le Conscient de la personne. Dans cette thèse donc, nous supposons que ces " transformations " n'existent pas et que toute motivation qui arrive au conscient est authentique.

les animaux synthétiques, appelés aussi *animats* (Meyer et Guillot 1991) dont les lois de fonctionnement sont inspirées de mécanismes naturels, et dont les comportements présentent certaines des capacités d'autonomie et d'adaptation des animaux. Les détails de cette partie bibliographique se trouvent dans la section II. 2, page 28.

Pour le cas des agents cognitifs, nous proposons les modèles utilisant l'architecture BDI (Belief-Desire-Intention) comme base de travail (Section II. 3. 2). En effet, par rapport à notre problématique, cette architecture présente un intérêt particulier : le concept de désir qui, souvent, est lié à la notion de besoins et de motivations, etc.

Enfin, à propos de l'hybridisme (cf. Section II.4), nous exposons et analysons des exemples d'architectures d'agents hybrides existants. Leur compréhension a pour but de recenser les limites de ces modèles.

A part ces modèles de comportement, notre bibliographie porte également sur la connaissance des concepts liés à la motivation tels le désir, le besoin, les actions, etc.

I. 4 Méthodologie expérimentale

I. 4. 1 Les applications

Pour valider le modèle issu de cette thèse, simulons deux cadres d'application : des robots fourrageurs et des paysans.

Les robots fourrageurs

Le cas des robots fourrageurs fait partie de ce que l'on appelle couramment le *cas d'école* dans lequel le modèle créé dans la phase théorique est, certes, implémenté mais n'est validé que sur des scénarios de situations " simples " dont les données sont généralement puisées dans un laboratoire. Cependant, cette première utilisation permet d'avoir un premier aperçu de la cohérence sémantique du modèle alors que celui-ci peut être encore dans une phase assez élémentaire de la conception. Il permet ainsi d'avoir une première version d'implémentation qui fonctionne, permettant déjà d'éliminer les premiers bugs identifiables à ce niveau. Pour notre étude, l'intérêt de ce niveau est de valider la partie fonctionnelle de chaque agent : étude de ses besoins, de ses actions, de ses réactions, etc.

Les paysans

Le second cadre d'application, plus réel, concerne la simulation de l'activité quotidienne de paysans dans la région de Mangatany, une région située au Centre Nord de Madagascar. Partiellement financé par l'Agence Universitaire Francophone (AUF), ce projet sur Mangatany consiste à comprendre le phénomène d'érosion engendré par des " lavaka " (textuellement " trou ") existant dans cette région (Rakotondraompiana et al. 2001). A moyen terme, notre travail se propose d'apporter sa contribution sur la partie anthropique, c'est-à-dire là où ce sont les facteurs humains et/ou les animaux (modélisés comme agent) qui sont considérés lors du processus de la création du " lavaka ". Dans cette thèse, l'objectif est pour le moment de modéliser individuellement chaque acteur intervenant dans le processus, et ce, en commençant par les paysans.

Ce second niveau d'application est utile dans la mesure où les problèmes à résoudre dans le monde réel sont plus complexes que ceux du cas d'école. L'existence de la validation à ce deuxième niveau permet d'observer comment l'architecture que nous avons créée s'applique dans un monde réel, ce qui est normalement la finalité de toute modélisation. Notre travail peut par la suite servir d'outil d'aide à la décision par des utilisateurs.

A noter que le projet est mené conjointement avec une équipe multidisciplinaire venant de l'Université d'Antananarivo (Madagascar) et de la Suisse, regroupée dans un projet nommé ARP (Action et Recherche Partagée). En outre, du côté de l'Ile de La Réunion, le projet est financièrement supporté par l'association d'organismes [Université de l'Océan Indien / Commission de l'Océan Indien / Union Européenne], dans le cadre de son projet " Nouvelle Technologie Appliquée à l'Environnement ".

Note : D'une manière générale, le résultat recherché à partir de ces deux applications n'est pas totalement le même. Avec les fourrageurs, nous voulons montrer que notre modèle basé sur les motivations naturelles s'adapte bien dans un contexte *hybride* c'est-à-dire aussi bien aux agents réactifs qu'aux agents cognitifs. Avec l'ajout du cadre des paysans, nous cherchons à prouver que le modèle est bien générique.

I. 4. 2 A propos de la Simulation

Il existe bon nombre de définitions du terme " simulation " mais nous n'en citons ici que deux, celles qui sont les plus proches de nos objectifs :

- (Drogoul 1993) : la simulation est la démarche scientifique qui consiste à réaliser une reproduction artificielle, appelée *modèle*, d'un phénomène réel que l'on désire étudier, à observer le comportement de cette reproduction lorsqu'on en fait varier certains paramètres, et à en induire ce qui se passerait dans la réalité sous l'influence de variations analogues.
- (Hill 1993) : la simulation consiste à faire évoluer une abstraction d'un système au cours du temps afin d'aider à comprendre le fonctionnement et le comportement de ce système et à appréhender de ses caractéristiques dans le but d'évaluer différentes décisions.

En fait, ces deux définitions convergent vers un même principe : introduire dans le simulateur des paramètres collectés dans le monde réel, puis simuler un scénario au cours du temps et interpréter les résultats issus de divers scénarios afin de prendre une décision. C'est ce que nous utilisons dans notre étude du comportement des paysans à Madagascar.

I. 5 Conclusion du chapitre

Ce chapitre I a principalement permis d'éclaircir le cadre de positionnement de cette thèse et la définition précise des diverses notions à utiliser dans les chapitres qui suivent. Cet éclaircissement est indispensable. En effet, un concept donné peut être interprété de diverses manières, en fonction du domaine de recherche considéré (SMA, biologie, etc.) voire de la thématique traitée dans un même domaine. Nous souhaitons ainsi au travers de ce chapitre, non seulement préciser la démarche de notre travail, mais également éviter toute confusion dans les concepts à mettre en œuvre par une définition précise des divers termes tels *hybride* (et donc également de *réactif* et *cognitif*), *besoins* et *motivations*, *naturel*, *instinct* et *homéostasie*, etc.

Après ces précisions, nous pouvons alors entamer une analyse objective de l'état de l'art reflétant la place des motivations dans les modèles agents actuels. C'est ce que nous allons présenter dans le chapitre II.

Chapitre II

La motivation dans les modèles agents actuels

Ce chapitre rapporte l'état de l'art sur la place de la motivation dans les modèles de comportement. Aussi, après avoir donné une vue globale sur cette notion de motivation dans le cadre agent, nous parcourons successivement les modèles d'agents réactifs (Section II. 2), cognitifs (Section II.3) et hybrides (Section II.4). Pour chaque modèle, une analyse critique de cet état de l'art termine la section correspondant. Enfin, la section II.5 apporte, en sus, une synthèse globale des analyses précédentes.

II. 1 Une vue globale sur la motivation

II. 1. 1 Motivations, besoins et actions

La compréhension de la notion de motivation passe au préalable par celle de deux autres concepts importants : les besoins et les actions. Nous les définissons donc en premier lieu avant d'aborder la question de la motivation proprement dite.

Besoins

Selon le psychologue belge Joseph Nuttin (1985), un *besoin*, pour une entité donnée, se définit comme étant une *exigence* qui est inhérente au fonctionnement de cette en-

tité. Plus précisément, c'est une exigence fonctionnelle qui se présente sous forme d'une relation *requise* entre cette entité et l'environnement[1]. Dans le présent contexte, l'environnement dont il est question, c'est la nature, la société, une action ou une situation extérieure, etc. Plusieurs exemples peuvent être donnés : une personne a besoin d'amis, une fleur a besoin d'eau, une voiture a besoin d'une réparation, etc.

Dans ce travail, nous reprenons cette définition en le cadrant dans le contexte des besoins de l'homme et de l'animal.

Actions

La définition d'une *action* n'est pas précise bien qu'il s'agisse d'une notion qui paraisse évidente à première vue. A la base, et quel que soit le contexte, l'action est d'abord *modification* (Ferber, 1997). Ensuite, si elle vient du vivant, l'action correspond à un geste, une tentative *d'influer* sur le monde pour le modifier selon son désir. D'ailleurs, les dictionnaires (ex : Robert 2001) définissent généralement l'action comme ce que fait *quelqu'un* et par quoi il réalise une intention[2].

Cependant, ces définitions ne font référence qu'à un être vivant (traduit ensuite dans le présent document par l'agent). Or, même sans geste, le monde dans lequel ce vivant se trouve ne reste pas dans un état stable. Il est modifié via ses propres spécificités et lois que l'agent ne contrôle pas toujours. Cette instabilité et les divers changements qui en découlent sont aussi des actions, alors qu'ils ne proviennent pas des agents. Enfin, une modification subie par un agent ne provient pas forcément de son intention mais de facteurs extérieurs comme les lois de l'univers ou encore les autres agents.

Pour le modèle que nous allons mettre en œuvre dans ce travail, la notion d'action correspond à celle de *geste* car notre étude se focalise davantage sur la réponse d'un agent face à ses motivations. Nous verrons néanmoins lors de l'étude de la plate-forme ADK sur laquelle ce travail va se reposer, comment l'action telle qu'elle est donnée par les autres définitions ci-dessus, sera aussi en partie prise en compte.

---

[1] Plus précisément, le besoin fait référence à cette relation entité - environnement en tant que *requise*.
[2] Voir la définition précise en page 43.

Motivations

Les motivations peuvent être définies comme l'ensemble des raisons qui poussent un individu à accomplir une action. Au sens social, la motivation correspond à l'action de fournir à quelqu'un les raisons qui le poussent à poursuivre un objectif. Selon Nuttin (1985), la motivation désigne l'aspect dynamique et directionnel (sélectif ou préférentiel) du comportement. C'est la motivation qui est responsable du fait qu'un comportement se dirige vers tel objet ou tel endroit plutôt que vers tel autre.

Ce sont les besoins qui se situent à la base de la motivation humaine (Maslow 1998). Inversement, pour qu'un besoin puisse être satisfait via une ou plusieurs actions, il faut que les motivations correspondantes existent. La présence de besoins en elle-même ne suffit donc pas à déclencher une action. La satisfaction *effective* d'un besoin dépendra ensuite du degré de la motivation de l'individu à le satisfaire.

Synthétiquement, nous avons la relation suivante : (besoin $\Leftrightarrow$ motivation) $\Rightarrow$action.

Les motivations naturelles sont liées à la satisfaction des besoins naturels. Concrètement, ces motivations sont les résultats de pulsions provenant de l'agent et dont les besoins correspondants demandent à être satisfaits.

II. 1. 2 Agent et motivations

Pour un agent, la motivation est gérée par un composant appelé le *système motivationnel*. Ce système constitue le premier élément intervenant dans le fonctionnement du système *conatif*[3] d'un agent (Figure II.1). Le rôle du système motivationnel consiste à prendre en compte l'ensemble des motivations de l'agent puis à élaborer ce que Burloud (1936) appelle des *tendances* qui sont en fait les éléments qui poussent ou contraignent un agent à agir (ou qui, au contraire, l'empêchent d'agir). Puis en fonction des informations dont l'agent dispose, le système *décisionnel* (qui est aussi un autre sous-système du système conatif) évalue ces tendances, choisit les plus prioritaires et détermine les moyens à mettre en œuvre pour les satisfaire.

En terme de classification, il existe deux principales sources de motivations : interne (issues des besoins physiques ou psychologiques) et externe (issues de l'environnement

[3] Le système conatif, c'est l'ensemble des structures et des processus permettant à un agent, tant réactif que cognitif d'agir (Ferber 1997).

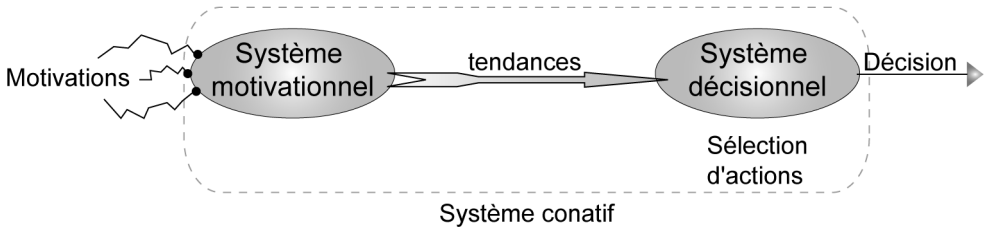


Figure II.1 : Structure d'un système conatif dont le premier composant est le système motivationnel

physique ou social). Cependant, la classification en fonction du critère " source " s'avère trop abstraite. Ferber (1997) propose une classification plus détaillée des motivations en fonction de quatre dimensions conceptuelles distinctes, illustrée à la figure II.2. Cette classification se présente comme suit :

- les motivations *personnelles* : liées à la satisfaction de plaisirs personnels (faim, sexuel, etc.). On les appelle également *motivation hédoniste*[4] ;
- les motivations provenant de *l'environnement* : les actions sont provoquées par les stimuli externes (une souris pour un chat, une fille pour un garçon, etc.). La notion de *désir* (ici, de l'entité présente dans l'environnement) peut également être située dans cette seconde catégorie.
- les motivations *sociales* ou *déontiques* : liées au souhait d'avoir une bonne opinion de la société sur soi, de bien faire son travail (motivations fonctionnelles), ou encore au souhait du respect des lois, de la réalisation des devoirs (règles déontiques). Cette catégorie de motivation peut être liée aux motivations personnelles[5] ;
- les motivations *relationnelles* : alors que les motivations sociales peuvent être liées à des concepts (lois, société, norme, etc.), les motivations relationnelles se situent purement entre des agents. La motivation d'un agent pour une action, c'est l'autre agent (pour cause de sentiment, de hiérarchie organisationnelle, etc.). Un des travaux marquant cette catégorie est ce-

[4] L'hédonisme, du grec *hedone* qui signifie " plaisir " est la doctrine morale qui fait du plaisir le principe ou le but de la vie. C'est aussi la motivation de l'activité économique par la recherche du maximum de satisfaction avec le minimum d'efforts (Larousse 1998).

[5] Par exemple, la non-finition d'un travail peut entraîner un non-paiement des salaires, ce qui pourrait empêcher la satisfaction d'un plaisir personnel (acheter à manger ou aller en vacances, etc.)

lui réalisé par Panzarasa et al. (1999), modélisant l'état mental d'un agent " sous l'influence " d'un autre.

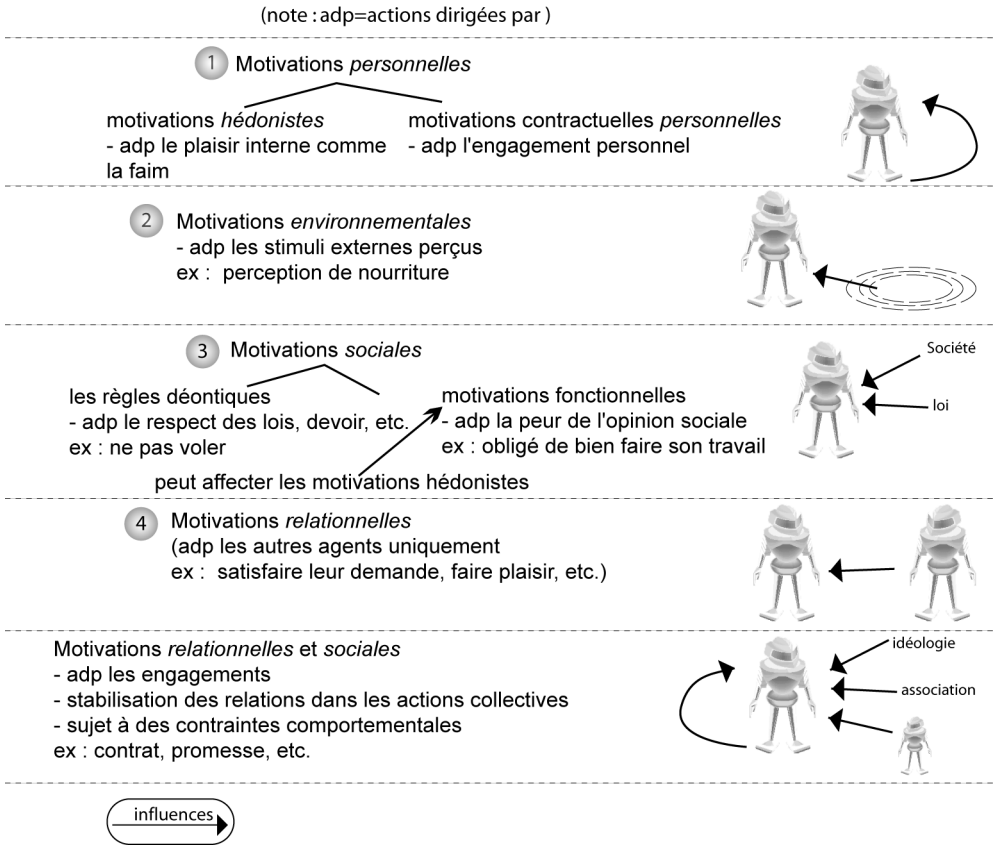


Figure II.2 : Les différentes catégories de motivations, proposées par Ferber (1997)

Cette classification de Ferber est en grande partie reprise par Calderoni (2002) dans son travail. Il considère aussi les quatre catégories de la classification avec seulement de légères modifications terminologiques :

- les motivations d'ordre *pulsionnel* (=personnel ou hédoniste pour Ferber), qui poussent l'agent à satisfaire ses besoins internes,
- les motivations d'ordre *perceptuel* (=environnementales pour Ferber), produites par ce que l'agent perçoit du monde extérieur,
- les motivations d'ordre *social* (idem pour Ferber), qui conduisent l'agent à satisfaire les besoins collectifs du groupe auquel il appartient
- les motivations d'ordre *fonctionnel* (relationnelles pour Ferber), traduisant les buts pré-établis par son concepteur.

Cette vue globale du concept de " motivation " étant présentée, les prochaines sections décrivent comment ce concept est introduit dans les divers modèles d'agents.

II. 2 Les modèles réactifs et la motivation

Dans cette section, nous étudions les agents réactifs (tels qu'ils sont définis à la section I. 2. 2 de ce document). Après avoir présenté notre cas d'étude sur ce type d'agents, à savoir les animats (Section II. 2. 1), nous expliquons comment est gérée la sélection de motivations (Section II. 2. 2). Enfin, des exemples de modèles réactifs (Section II.2.3) terminent cette section.

II. 2. 1 Etude du cas réactif : les animats

Les animats ...

Les *animats* sont définis par Guillot et Meyer (1998) comme des animaux synthétiques qui habitent dans des mondes synthétiques et qui vivent, se nourrissent, se reproduisent et meurent dans ces mondes. Ce sont des agents qui, soit sont simulés sur ordinateur, soit prennent la forme de robots matériels. Leurs lois de fonctionnement sont inspirées de mécanismes naturels et leurs comportements présentent certaines des capacités d'autonomie et d'adaptation des animaux.

...et les systèmes réactifs

Les systèmes réactifs sont aussi appelés *systèmes à base de comportements*[6] (de l'anglais Behavior-Based System ou BBS), contrairement aux systèmes cognitifs plus connus comme des *systèmes à base de connaissances* (de l'anglais Knowledge-Based System ou KBS). La théorie BBS fournit une nouvelle philosophie pour la construction d'agents autonomes, inspirée du domaine de l'éthologie, ce qui fait tout son intérêt dans l'étude des animats.

Dans les BBS, l'agent (ici l'animat) interagit directement avec l'environnement (généralement dynamique, complexe et imprévisible) qui constitue le domaine du problème. Cette prise en compte de l'environnement se traduit pour l'agent par un mécanisme de *perception* issu de ses capacités sensori-motrices, suivi successivement de *délibération* et *d'action.*

La triade perception - délibération - action

La perception peut être définie comme le processus par lequel l'agent acquiert des informations sur le monde qui l'entoure. Ces informations vont alors lui permettre d'élaborer son action en poursuivant ses objectifs. Formellement, si l'on suppose qu'il est possible de caractériser l'ensemble Σ des états possibles du monde, la perception correspond à la qualité de pouvoir classer et distinguer ces états. On considérera la perception comme une fonction qui associe à Σ un ensemble de valeurs appelées *percepts* (c'est-à-dire des informations obtenues à partir de la perception). Si l'on appelle $\mathtt{P}_a$ l'ensemble des percepts associés à un agent $\mathtt{a}$, alors la fonction de perception d'un agent peut être définie comme suit :

$$\mathtt{Percept}_a \ : \Sigma \rightarrow \mathtt{P}_a$$

qui associe un percept à chaque état du monde perceptible par cet agent. Un percept peut donc revêtir un aspect complexe et peut par exemple être sous la forme d'une composite (un vecteur) dont chaque composante renferme une information particulière (distance, message textuel, intensité, temps, etc.).

Le mécanisme de perception est suivi d'actions. Ces dernières tendent à modifier l'état du SMA (soit en agissant directement sur l'environnement, soit en émettant des messages à l'adresse d'autres agents). Les actions situées constituent le modèle le plus élémentaire

[6] Voir par exemple Gershenson et al. (2000) ou Behnke et al. (2000).

des actions réactives. Les actions d'un agent sont dites *situées* quand elles sont relatives à sa position dans l'environnement et à l'état du monde qu'il perçoit. Le principe repose sur un schéma de type *stimulus/réponse*. Toutes les actions n'ont pas exactement le même impact sur un agent réactif. On distingue généralement les comportements appétitifs (ex : `chercher_nourriture`) des actions de consommation (ex : `manger`). L'intérêt de cette distinction vient de ce que la phase d'appétence peut se produire en absence de stimuli environnementaux alors que les actes consommatoires dépendent entièrement de la présence de stimuli (Ferber, 1997).

Entre la perception et l'action se trouve la délibération[7]. C'est elle qui rend compte du comportement effectif de l'agent et constitue donc la partie la plus étudiée et la plus complexe. Dans la littérature agent cependant, il faut faire attention car le sens exact du terme " délibération " est très ambigu. En effet, le terme est aussi utilisé dans le monde du cognitif (pour représenter une phase de réflexion ou de raisonnement) et de l'hybridisme (par opposition à " réaction "). En fait, au niveau du réactif, la délibération correspond à une représentation abstraite d'un processus qui transforme une perception d'un environnement en une action vers un environnement. Cette façon de voir permet ainsi de considérer, même des agents extrêmement simples.

II. 2. 2 Motivations et sélection d'actions

Les motivations étant sources d'actions, la sélection d'actions apparaît être la suite logique de leur manifestation[8]. Nous la mettons dans cette section concernant les systèmes réactifs dans la mesure où la sélection d'actions a généralement été appliquée sur ces systèmes (Gershenson et al. 2000). La raison avancée par ces derniers est que ces agents, avec leur capteur, sont plus en contact direct avec l'environnement physique. Mais il est aussi possible de généraliser le processus en tenant compte des événements internes (ex : pulsion) qui se manifestent au sein de l'agent. Le problème de la sélection d'actions se résume à celui de déterminer l'action la plus appropriée à exécuter parmi un répertoire des actions possibles (par exemple `boire`, `dormir`, `se déplacer`, ...).

---

[7] Littéralement, d'après le dictionnaire Larousse (1998), la délibération est la réflexion qui précède la décision qui l'acte par lequel on est poussé à agir.

[8] Dans le cas précis des agents réactifs, la modélisation de la sélection de motivations inclut aussi celle de la sélection d'actions car selon Ferber (1997), les notions de motivations et d'actions sont intrinsèquement liées chez ces types d'agents.

Le *mécanisme de sélection d'actions* (plus connu sous l'abréviation anglophone de ASM pour Action Selection Mechanism) est un mécanisme de calcul qui peut produire une action choisie comme sortie à partir de différents stimuli introduits en entrée. Décider de l'action la plus appropriée nécessite alors une prise en compte de ces stimuli, qu'ils soient externes (par exemple un prédateur, une source de nourriture, un compagnon), ou internes (par exemple taux du sucre dans le sang, température du corps). Du point de vue de la motivation, il s'agit de déterminer le *degré de motivation* correspondant à chaque action de la liste.

La question fondamentale est de savoir quelle est la formule ou la fonction adéquate (le cas échéant) permettant d'arriver au choix. La présentation d'exemples de modèles réactifs dans la section II.2.3 aide à y apporter une réponse en tentant de dégager :

- quels sont les concepts utilisés lors de la construction des architectures,
- comment les motivations y sont prises en compte et modélisées,
- et comment le degré de motivation est déterminé.

II. 2. 3 Exemple de modèles

Dans cette catégorie " réactif ", on distingue notamment les modèles hiérarchiques, les modèles de sélection par compétition et les modèles d'action par combinaison tendancielle.

Les modèles hiérarchiques

La plus connue de ce type de modèle est sans doute l'architecture de Subsomption de Brooks (1986). Dans cette architecture, les comportements sont organisés de manière hiérarchique (Figure II.3). Cette hiérarchie a été mise en œuvre de façon à ce qu'un comportement de plus haute priorité inhibe un comportement de plus faible priorité. Ici, un comportement est une fonction qui, à un stimulus reçu d'un ou plusieurs senseurs, associe une action réalisée par un ou plusieurs effecteurs. Malgré la structure de dominance, chaque niveau est responsable d'un comportement complet, ayant accès à l'information sensorielle qu'il exige et la capacité d'envoyer des instructions aux effecteurs.

Le mécanisme de la sélection d'actions fonctionne comme suit : à chaque pas de temps, chacun des comportements prend connaissance des stimuli provenant des senseurs aux-

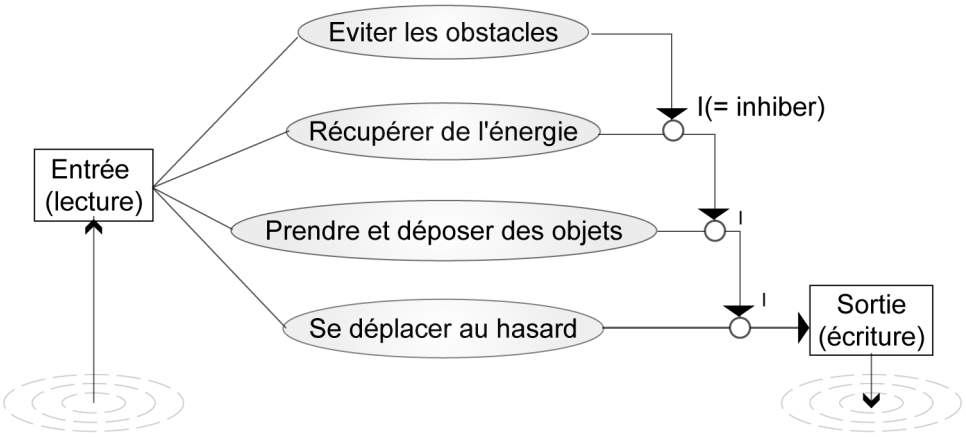


Figure II.3 : Un exemple de modèle basé sur l'architecture de subsomption.

quels il est connecté, puis calcule une réponse *locale* adéquate et soumet cette réponse à un module chargé d'arbitrer entre les diverses réponses qui lui parviennent des différents comportements. En règle générale, l'approche " subsomption " utilise une hiérarchie de comportements à priorités fixées : les commandes d'activation des effecteurs sont tout simplement celles émanant des comportements actifs prioritaires. Un comportement prioritaire inactif laisse la décision aux comportements moins prioritaires. Cela requiert du concepteur une connaissance exhaustive de tous les comportements possibles afin de fixer les priorités, ce qui peut rapidement devenir difficile si le nombre de comportements atomiques augmente.

Les modèles de sélection par compétition

L'un des plus célèbres modèles à base d'actions compétitives est le réseau de Maes (1991) baptisé ANA pour *Agent Network Architecture* (Figure II.4). Le modèle se compose d'un ensemble de modules, chacun correspondant à une action élémentaire telle que `fuir`,

`boire`, etc. Chaque module `i` se présente sous la forme `i=(`c_i`,` a_i`,` s_i`,` α_i`)` dans laquelle :

- le paramètre c_i contient une liste des préconditions qui doivent être satisfaites avant qu'un module devienne actif,
- le paramètre a_i se réfère à une liste additive émergeant de la conséquence de l'exécution du module
- le paramètre s_i entre en jeu lorsque des modules inhibent d'autres,
- le paramètre α_i correspond au niveau d'activation du module.

En outre, un agent ANA possède un certain nombre de motivations innées qui sont reliées aux activités consommatoires de l'agent. Les activités consommatoires, une fois exécutées, mènent à une réduction ou à une satisfaction de la motivation correspondante : `manger` satisfait la faim, `boire` étanche la soif, et ainsi de suite. Quelques activités appétitives mènent directement à une activité consommatoire ; d'autres sont incorporées dans des chaînes des activités qui mènent l'animat plus près du but recherché. Ainsi, par rapport à une nourriture, `manger` est préférable à `aller_vers`, qui à son tour, est préférable à celui de `se_diriger_vers` (mémorisation), lui-même préférable à `trouver`. A n'importe quel moment, chaque motivation sera caractérisée par un niveau d'activation de " faim ", de " soif ", de " peur ", etc.

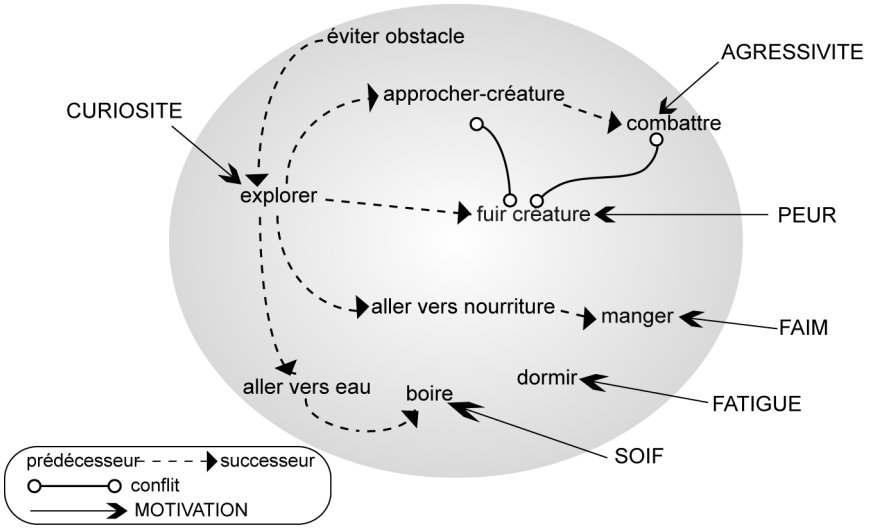


Figure II.4 : Le réseau d'activation des tâches du système ANA de Maes (1991)

Toujours dans le cadre des modèles de sélection par compétition, il existe celui proposé par Drogoul (1993), baptisé EMF (Etho Modelling Framework). Dans ce modèle, il y a une indépendance totale entre ce que Drogoul appelle les tâches. Une tâche correspond à une action macroscopique qui représente une certaine unité (ex : `chercher_nourriture`, `explorer`, etc. pour un robot) alors que les actions basiques utilisées pour les réaliser sont appelées *primitives*. Dans le choix des tâches, divers paramètres sont pris en compte : le poids de la tâche, le contexte d'application, les autres informations venant de l'extérieur, etc. Par exemple, dans l'application du modèle EMF sur le système MANTA de Drogoul et Ferber (1994), les paramètres d'activation d'une tâche sont les suivants :

- les stimuli internes et externes $x_i(t)$ permettant de sélectionner la tâche. Ici, les deux stimuli sont multipliés ;
- le poids $\omega_i(t)$ de la tâche au moment de la sélection, qui précise l'importance relative de la tâche par rapport aux autres ;
- le seuil `s` de son déclenchement ;
- le niveau d'activité $a_i(t)$ qui est donné par la formule suivante :

$$a_i(t) = \frac{\omega_i(t)}{\sum_{j=1}^{n} \omega_j(t)} x_i(t)$$

Lorsqu'une tâche est sélectionnée, elle devient la tâche courante. Lorsqu'elle est activée, la tâche précédente est suspendue et présente une activité de " veille ". Cette activité consiste à recevoir les informations provenant des capteurs et à calculer les valeurs des paramètres de sélection qui en dépendent afin de demeurer dans la compétition. Par ailleurs, chaque agent dispose d'une tâche spéciale qui spécifie son comportement *par défaut* lorsque aucune autre tâche n'est sélectionnable[9].

Le modèle EMF (Figure II.5) se distingue par l'existence de deux processus de rétroaction :

- la rétroaction *positive* : le système tend à se spécialiser dans le déclenchement préférentiel de certaines tâches : plus une tâche est

[9] Dans l'architecture de subsomption, cette tâche équivaut à l'action mise au niveau le plus bas de la hiérarchie.

déclenchée, plus elle aura une chance de se déclencher dans le futur. Lorsqu'elle est sélectionnée, son poids $\omega_i(t)$ est augmenté d'un incrément δ ;

– la rétroaction *négative* : une tâche qui s'exécute voit son poids diminuer pour faire en sorte que les autres tâches aient une chance d'être sélectionnées. Si une tâche a satisfait un besoin, le signal correspondant à ce besoin sera diminué.

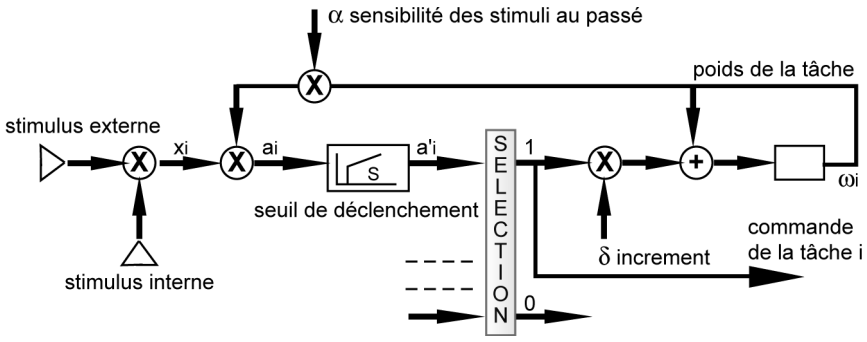


Figure II.5 : Diagramme de sélection de tâches dans le modèle EMF de Drogoul (1993)

Plus récemment, Gershenson et al. (2000) ont aussi travaillé sur les modèles à tâches compétitives avec un modèle nommé IBeNet (Internal Behaviour Network) ou réseau de comportements internes[10]. Celui-ci se base par contre sur une architecture à base de tableau noir distribué. Le terme de *distribué* est adopté en raison de l'existence de deux nœuds de tableaux qui sont en interaction : le nœud cognitif et le nœud motivationnel. Les tâches internes requises pour le contrôle de la sélection d'actions ne sont possibles qu'au travers de la coopération de ces deux nœuds. Les entrées des extérocepteurs[11] proviennent du système perceptuel du robot tandis que les entrées des propriocepteurs[12] proviennent des moyens internes. Les sorties des actuateurs sont dirigées vers le moteur du système.

---

[10] Structurellement, un comportement interne se définit comme un ensemble de règles de production (<si> <condition> alors <action>). Une règle de production correspond à un comportement élémentaire.

[11] L'extérocepteur est un récepteur sensoriel percevant des stimuli du monde externe. Ici, il sert d'interface entre le système perceptuel et le nœud cognitif (voir Figure II.6).

[12] Le propriocepteur est un récepteur d'une information concernant la localisation du corps, son mouvement, etc. Les propriocepteurs établissent l'interface entre la partie interne et le nœud motivationnel de l'architecture.

La structure d'un nœud à base de tableau noir est définie par cinq éléments de base : le tableau noir lui-même, les comportements internes, les registres d'activation des comportements internes (non présentés sur la figure II.6), les mécanismes d'interface/communication, et le mécanisme de compétition entre les différentes motivations. Le tableau noir, par lequel les comportements internes communiquent, est une structure de données partagée par laquelle ces comportements internes exécutent leurs actions finales. Les mécanismes d'interface/communication fonctionnent également via le tableau noir. Les comportements internes produisent les changements du tableau noir, qui mènent incrémentalement à la formation d'une solution dans le problème de la sélection d'actions.

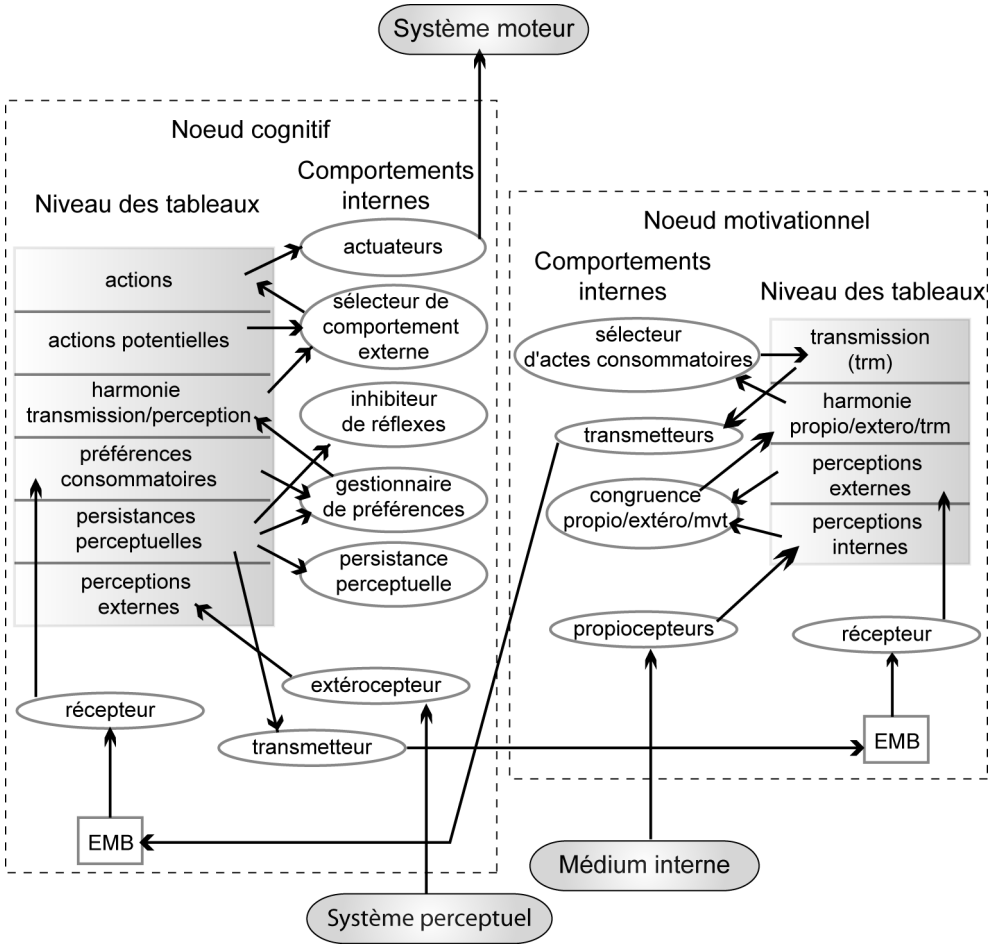


Figure II.6 : Structure de IBeNet, le réseau de comportement à base de tableaux noirs (Gershenson et al. 2000).

Au niveau du nœud cognitif, les comportements internes suivants ont été définis : persistance perceptuelle, gestion des préférences, inhibition des réflexes et sélecteur de comportements externes. Le nœud cognitif reçoit des signaux du système perceptuel (via les extérocepteurs) et les envoie au nœud motivationnel qui les reçoit par le mécanisme de réception. La boîte EMB (pour Event MailBox) agit comme une boîte aux lettres mettant les événements en file d'attente. Le nœud cognitif contient les processus de la représentation des signaux perceptuels, de l'intégration des signaux internes et externes, de l'inhibition réflexe de réponse, et de la sélection du comportement externe.

Le nœud motivationnel, quant à lui, reçoit des signaux en provenance du support interne (via les propriocepteurs) et du nœud cognitif (via le mécanisme de récepteur), puis, en retour, envoie des signaux au nœud cognitif par le mécanisme d'émission. Il contient la combinaison des signaux internes et externes. Au final, le comportement externe observé dans l'entité est fortement dépendant de ses états internes. Tous les états internes pour lesquels des signaux externes existent se concurrencent entre eux pour déterminer le comportement externe final que l'entité exécutera. Le tableau noir du domaine du nœud motivationnel organise les éléments de solutions sur quatre niveaux d'abstraction : perceptions internes, perceptions externes, congruence de propio/extero/transmission, et transmission.

Dans ce modèle, les actes consommatoires et appétitifs existent toujours et sont même associés à une même motivation. Mais contrairement à l'exécution de l'acte consommatoire, celle de l'acte appétitif ne diminue pas le degré de motivation.

Les modèles d'action par combinaison tendancielle

Dans ce type d'approche, il n'y a plus de tâches sous forme de modules comme dans les modèles précédents. L'expression du comportement d'agents passe par l' intermédiaire de l'environnement qui est lui, conçu comme un espace métrique. Lorsque des agents se déplacent, qu'il s'agisse par exemple de robots mobiles ou de véhicules qui tentent de s'éviter, leur comportement se manifeste *naturellement* comme un mouvement dans un espace euclidien. Les déplacements des agents sont modélisés sous forme de *vecteurs* (Simonin et Ferber 2001, Simonin et al. 2002) et plus particulièrement dans le cas présent, comme des additions de *vecteurs* dans un espace euclidien.

Au niveau de l'environnement, on applique la notion de *champ* qui sert en physique à expliquer l'origine des forces et donc le mouvement des particules. Ici, le principe général est que des objets émettent des signaux dont l'intensité est proportionnelle au carré de la distance au but. Ces signaux sont diffusés dans tout l'espace et définissent un potentiel `U(p)` en tout point de l'espace. On considère alors que les buts définissent des champs de potentiels attractifs alors que les obstacles correspondent à des champs de potentiels répulsifs. L'Equation (II.1) en présente l'expression formalisée dans laquelle $\texttt{dist}_{influence}$ correspond à une barrière à partir de laquelle l'influence des obstacles ne se fera plus sentir.

$$\begin{aligned} &U(p) = U_{attr}(p) + U_{repul}(p) \\ &U_{attr}(p) = k * dist(p,\, p_{but})^2 \\ &U_{repul}(p) = \begin{cases} k' * \frac{1}{dist(p,p_{obs})^2} \, si \, dist(p, p_{obs}) \leq dist_{influence} \\ 0 \, sinon \end{cases} \end{aligned} \tag{II.1}$$

L'intérêt de cette technique est qu'il est possible de combiner plusieurs comportements. Par exemple, il est possible de définir deux comportements : `aller_vers_but`, et `éviter_obstacle` (Figure II.7). Le comportement global décrit par le vecteur $\overrightarrow{X}$ peut alors être défini par l'équation vectorielle :

$$\overrightarrow{X} = \frac{\alpha\overrightarrow{But} + \beta\overrightarrow{Obstacle}}{\alpha+\beta}$$

où les vecteurs $\overrightarrow{But}$ et $\overrightarrow{Obstacle}$ représentent respectivement l'attraction vers un but et le contournement d'un obstacle. L'intensité de ces vecteurs dépend du potentiel produit par le but et par l'obstacle. Les paramètres α et β correspondent à des facteurs critiques qui dépendent de la position de l'agent.

II. 2. 4 Analyse synthétique

Au vu des modèles présentés précédemment, nous constatons que les systèmes réactifs basés sur les animats considèrent largement l'existence des motivations naturelles dans le comportement des agents. En effet, ces systèmes basent toujours leurs applications sur des notions assimilées à des stimuli internes (Ferber 1997) telles que la faim, le sommeil, la fuite d'un prédateur (qui correspond à la manifestation d'un instinct naturel de préservation), etc. Cela implique que ces modèles attachent une importance particulière aux sources naturelles du comportement animal/humain pour expliquer le comportement des agents.

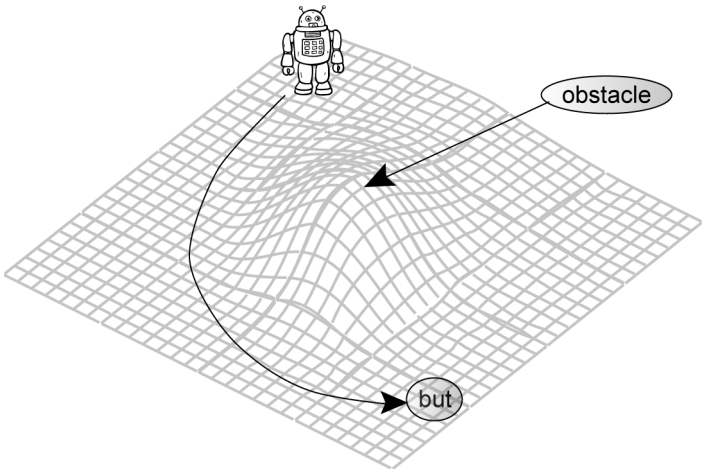


Figure II.7 : Superposition des champs attracteur (le but) et répulseur (l'obstacle) et trajectoire résultante d'un agent mobile.

Les stimuli internes constituent ainsi un facteur important dans la sélection d'actions. Le problème de la sélection d'actions est pratiquement résolu lorsque l'agent a pu déterminer le *niveau d'activations* (appelons-le `na`) de chacune des actions de l'agent. C'est le `na` qui alors caractérise le *degré de motivation* d'un agent à exécuter ou non une action donnée. La valeur du `na` est connue en multipliant ou en additionnant les stimuli internes et externes de l'agent. Puis est choisie l'action ayant le `na` le plus élevé (supposons `na_max`). Le problème se complique lorsque plusieurs actions ont toutes les mêmes `na_max` c'est-à-dire que l'agent a la même motivation à les exécuter toutes. Nous sommes en présence de tâches compétitives. Le modèle EMF de Drogoul (1993) propose de choisir d'une manière aléatoire entre les actions qui ont le même niveau d'activation. Cela implique pourtant que seule une action à la fois pourra être exécutée.

L'approche par combinaison tendancielle est selon nous celle qui devrait être la plus approfondie. D'un côté, elle permet de proposer une première solution au problème précédent de Drogoul en offrant la possibilité d'exécuter plusieurs actions à la fois. Ensuite, grâce aux notions de champs et de vecteurs qui s'y trouvent, cette approche modélise bien le mécanisme naturel du vivant : *attraction/répulsion*. C'est le cas des animaux sociaux dans lesquels la formation du collectif est possible grâce uniquement à une interattraction entre les membres et une disparition d'une répulsion interindividuelle agressive (Bonabeau et Theraulaz 1994). C'est aussi le cas chez les humains, où ce phénomène d'attraction/répulsion envers une autre entité (objet ou personne) est

inné. Dans le cadre SMA, ce comportement des systèmes biologiques a servi de source d'inspiration dans la fabrication dans bien de modèles d'agents. Pour ce faire, le chercheur expérimente sur des agents ce qu'il perçoit dans le monde animal comme celui des fourmis (Drogoul et Ferber, 1994), des termites (Quinqueton et Hamadi, 1999) ou encore des araignées (Chevrier, 2002).

Dans ce présent cadre réactif, le point sur lequel nous contribuerons concerne l'enrichissement des critères qui déterminent le degré de motivation des agents. En fait, la solution de *choix au hasard* présentée par Drogoul ne devrait être adoptée qu'en dernier recours après que tous les autres critères sont épuisés.

II. 3 Les modèles cognitifs et la motivation naturelle

D'une manière analogue à la section II. 2, cette section établit la place qu'occupe la motivation naturelle dans les modèles d'agents cognitifs. Alors que chez les agents réactifs, le cas observé est l'animat, nous nous intéressons ici aux modèles BDI.

II. 3. 1 Principe des agents cognitifs

L'existence des agents cognitifs est liée avec l'arrivée de l'Intelligence Artificielle Distribuée (IAD) cognitive. Cette dernière avait eu pour origine la volonté de faire évoluer les concepts et les outils de l'Intelligence Artificielle (IA) traditionnelle pour synthétiser la part sociale ou interactionnelle de la connaissance et de l'intelligence humaines (Bond et Gasser, 1988). En découlent la notion de représentation des connaissances (de soi, du monde, des autres, de ses buts) et la notion de contrôle par l'agent de son comportement, donc de sa perception et de ses actions, en fonction justement de ses représentations (Sadek, 1994). Mais contrairement à l'IA classique, l'IAD fait appel à un grand nombre de concepts tels que la coopération, la coordination d'actions, la négociation, les conflits, etc. (Camalot 2000). Ces concepts trouvent leur origine dans la sociologie, dans la biologie ou dans l'éthologie et aussi, à l'instar de cette thèse, dans les concepts de représentation de connaissances, comme la psychologie cognitive que nous prenons d'ailleurs en compte dans cette thèse.

Les agents cognitifs ont une symbolique explicite de représentation de leur environnement. Ils sont dirigés par des intentions, des buts et des plans explicites qui guident

leur comportement. Ce sont des éléments leur permettant d'évaluer et de choisir parmi plusieurs actions.

Nous pouvons leur donner entre autres les caractéristiques suivantes :

- ils ont une représentation explicite du monde, et éventuellement une mémoire du passé.
- ils peuvent planifier ou encore effectuer des actes intentionnels tels que l'engagement, etc. (voir la notion d'items cognitifs ci-dessous)

Pour manifester leur partie sociale, les agents cognitifs forment intentionnellement un ou plusieurs groupes. Les raisons du regroupement peuvent passer d'un simple *désir* d'être ensemble autour d'un café (motivé par un désir social de bas niveau), jusqu'à un regroupement intentionnel en vue d'atteindre des objectifs communs. Nous parlons dans ce dernier cas de coopération intentionnelle (Camalot 2000) dans laquelle plusieurs facteurs tels que la planification des tâches, la négociation en situation de conflit, etc. interviennent. En outre, il suffit de peu d'agents cognitifs pour accomplir des tâches complexes. Cette situation est considérée comme " normale " dans la mesure où les agents cognitifs sont considérés comme *autosuffisants* (Ferber 1996) vu qu'ils sont capables de " stocker " les informations dont ils ont besoin, puis d'anticiper, de raisonner et d'agir uniquement à partir de ces informations, en dehors de tout événement.

Avant de parler d'architectures ou de comportements cognitifs, il est important de définir les items cognitifs, appelés aussi les *cognitons*[13] que nous reprenons plus tard dans la thèse. D'une manière simple, les cognitons sont les unités élémentaires mises en œuvre par l'agent pendant le raisonnement ou la prise de décision (Ferber 1997). Les *motivations* ainsi que bon nombre d'éléments tels que *croyance, intention, but, désir*, en font partie. Les cognitons sont reliés à d'autres cognitons pour former une configuration mentale dynamique évoluant avec le temps. Leur interaction permet de décrire et d'analyser le comportement d'un agent. Ils interviennent dans la détermination de l'état mental d'un agent.

[13] Ce terme est construit à partir du vocabulaire " cognitif " et de la terminaison " on " qui désigne quelque chose d'insécable, une particule élémentaire.

II. 3. 2 Etude de cas : les modèles BDI

Nous avons explicitement choisi d'étudier BDI[14] dans le cadre cognitif. La raison principale de ce choix porte sur l'existence de la notion de *désir* (une notion plus ou moins proche de la motivation) comme un des points fondamentaux de l'architecture.

Depuis bon nombre d'années, l'architecture BDI, appelée aussi une architecture délibérative par Ferber (1996), constitue sans doute une référence dans la conception d'architectures d'agents cognitifs (Rao et Georgeff 1992, Jennings 1993, Panzarasa et al. 1999, Seow et How 2002). BDI renferme une description de ce que les croyances, les désirs et les intentions ont besoin pour faire partie de l'état mental de l'agent. Il ne dit rien sur la manière dont ces notions seront précisément intégrées (Werner 1996). Les architectures à base de BDI sont généralement axées sur la représentation symbolique que les agents se font du monde. En se basant sur le désir des agents, la conception de ces architectures est généralement réalisée au niveau des intentions individuelles et du comportement et ce, pour atteindre le but.

Le " Belief " (croyance)

Les croyances décrivent l'état du monde du point de vue d'un agent, et donc la manière dont il se représente son environnement ainsi que lui-même. Il faut bien distinguer ses propres croyances et celles des autres. Les premières peuvent être plus cohérentes avec l'état dans lequel se trouve l'agent alors que les secondes ne sont que partielles et éventuellement erronées.

Le " Desire "

Le dictionnaire Larousse (2000) définit le désir[15] comme la tendance vers la possession ou la réalisation de quelque chose. Il peut parfois être confondu avec les notions de besoins, de buts et surtout de motivations. Plus loin encore, Brazier et al. (1999) définissent, dans leur architecture nommée DESIRE, le désir comme une grande envie se reflétant au *souhait* ou au *vouloir*.

---

[14] BDI est l'abréviation de Belief-Desire-Intention.

[15] Le mot *désir* vient du latin *desiderare* qui signifie " regretter l'absence de "

Un agent peut " héberger " des désirs qui sont impossibles à réaliser. Les désirs peuvent être dirigés par les préférences et sont les seules attitudes de motivation sujet à l'incohérence (contradiction entre souhait et réalité). Des désirs peuvent se rapporter à l'état (désiré) du monde (et des autres agents), mais également aux actions que l'on souhaite exécuter (rapprochement avec l'intention).

L'Intention

Elle décrit ce qui meut un agent et correspond à la volonté *consciente* d'accomplir un but ou d'effectuer une action. Les intentions décrivent ce qui permet à un agent de passer à l'acte.

On distingue deux formes d'intention (Searle 1983) :

- l'intention *immédiate* qui est intrinsèquement liée à l'exécution effective d'une action. Les facultés intentionnelles des animaux (pour ceux qui en ont) se limitent généralement à ce niveau.
- l'intention *future* (processus à durée non nulle) qui suppose une planification future et se trouve à la source d'un comportement[16], en particulier des autres intentions à formuler pour atteindre le but initial. Elle est intéressante du point de vue des agents intentionnels[17] car elle suppose un mécanisme d'anticipation d'action et de planification des événements futurs. L'intention est aussi associée à des engagements qui dirigent et contrôlent les futures activités de l'agent.

Quelle que soit la forme de l'intention, un *but* lui est toujours associé : un agent va intentionnellement accomplir une action parce qu'il croit que ce qu'il va faire va lui permettre de satisfaire un but.

---

[16] L'intention, tout comme la motivation, est aussi source de comportement. La différence est que la motivation permet de *justifier* le déclenchement de l'action (indépendamment des moyens par lesquels elle sera mise en œuvre) tandis que l'intention est liée à sa réalisation effective par l'agent. En fait, les deux notions se trouvent dans une même chaine de comportement mais à des étapes différentes.

[17] Ce cas suit donc ici la définition d'un agent intentionnel, adoptée par Moulin et Chaib-Draa (1996) et présentée à la section I.2.2.

II. 3. 3 Désir et motivation

En rappelant que le point qui nous intéresse dans le cadre BDI est celui du " désir ", nous tentons dans cette section, de comprendre sous quel aspect cette notion peut se rapprocher (voire être confondue) avec la motivation.

Le désir comme un but

Dans bon nombre d'architectures, le désir correspond au *but* de l'agent. Ce but lui sert de " guide " au moment où il raisonne, où il forme ses intentions et ses engagements. Dans l'architecture IRMA[18] de par exemple (Figure II.8), le désir est pris en compte au moment où l'agent est en phase de délibération ou en phase de construction de son plan. Le processus se déroule en parallèle avec le contrôle des incompatibilités entre le but et les opportunités présentés à l'agent. D'autres architectures basées sur BDI comme PRS (Georgeff et Ingrand 1989) ou GRATE* (Jennings 1993) ont suivi ce mécanisme.

Le désir est donc ici à la fois vu comme un but et une source d'un ensemble d'actions possibles pour l'agent. Il constitue aussi un élément de contrôle lors de l'évaluation progressive de ses actions, en vue de leur réalisation finale. Rao et Georgeff (1992) vont plus loin en considérant le désir comme l'action elle-même, ou plus précisément l'ensemble des actions que l'agent se doit d'accomplir.

Le désir comme une attitude mentale

Kiss (1992) et Brazier et al. (1999) présentent directement les désirs comme les motivations de l'agent en utilisant le terme *d'attitude motivationnelle*. Autrement dit, le désir fait partie intégrante de l'attitude mentale de l'agent.

Panzarasa et al. (1999) ont aussi considéré le désir comme une attitude mentale. Dans ce travail, la proposition `desire(a, Φ)` signifie que l'agent `a` adopte une attitude mentale `désir` vis-à-vis de Φ où Φ est aussi une attitude mentale (désir, croyance, connaissance, but, etc.). Et pour eux, c'est l'ensemble des désirs qui constitue la motivation *primaire* des actions. Ils définissent ensuite les buts comme un sous ensemble de désirs consistants (c'est-à-dire réalisables).

[18] L'architecture IRMA, acronyme de " Intelligent Resource-bounded Machine Architecture " semble être le pionnier des travaux portant sur les BDI (Haddadi et Sundermeyer 1996).

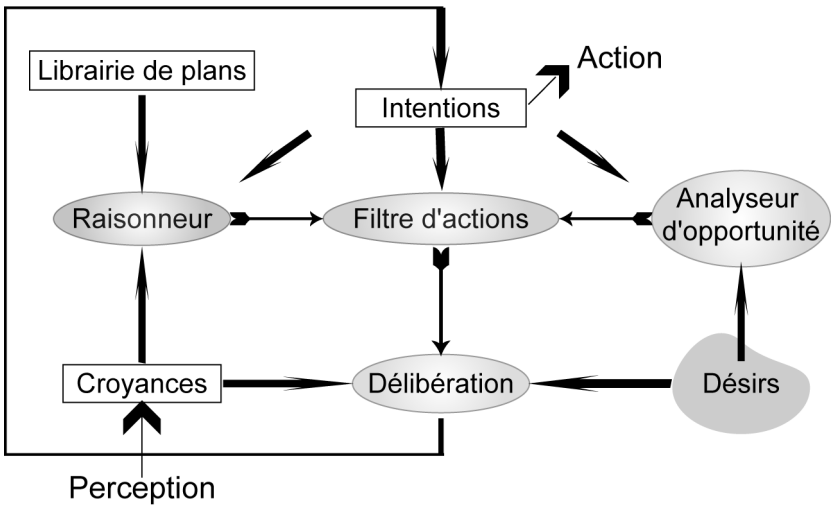


Figure II.8 : Architecture simplifiée de IRMA créée par Bratman et al. (1988).

En somme ...

Tous les modèles que nous venons de citer reposent sur le modèle BDI. Bien qu'il existe une légère différence dans la manière d'énoncer le concept, tous ces travaux s'accordent globalement à considérer le désir comme source d'actions. A ce titre, il peut être considéré comme une motivation. Cependant, il faut s'entendre sur le terme " motivation " au sein des agents cognitifs. En fait, ce n'est pas la motivation naturelle à laquelle nous faisons référence dans cette thèse. Ici, les motivations sont des cognitons, c'est-à-dire des unités élémentaires mises en œuvre pendant le raisonnement ou une prise de décision (cf. Section II.3.1).

II. 3. 4 Analyse synthétique

Au vu des éléments développés dans toute cette section portant sur les agents cognitifs, la notion de motivations *naturelles* est moins accentuée voire presque inexistante, sans doute parce que les travaux sur le plan cognitif sont axés sur la formalisation du monde, et focalisés sur le raisonnement de l'agent (c'est-à-dire à un niveau de traitement plus haut). En conséquence, les modélisateurs ont tendance à faire abstraction de l'existence des pulsions de base innées dans les agents. Ces motivations naturelles qui sont sources

initiales d'action sont alors implicitement considérées comme existantes et déjà " acceptées " par l'agent. A partir de cette " acceptation ", les travaux portent ensuite sur la manière dont ces actions sont réalisées via divers concepts propres aux humains tels l'organisation intentionnelle, la négociation, etc.

Le désir est cependant une voie ouvrant vers la modélisation du naturel. Nous utilisons bien le mot *tendance* car jusqu'ici, il est pris seulement comme un but résultant d'un plan tandis que le caractère émotionnel du désir naturel[19] tel qu'il l'est dans le monde humain/animal ne se reflète pas au niveau cognitif.

Il existe également des cas où la motivation naturelle est " ressentie " au niveau cognitif, mais d'une manière " floue " et pas toujours explicite. Prenons l'exemple d'une association en nous intéressant ici aux divers facteurs individuels et sociaux qui motivent les agents à y adhérer. Une personne qui entre dans une association aura pour tâche de coopérer avec les membres pour assurer les objectifs de celle-ci. Nous sommes ici dans un cadre cognitif de coopération (Camalot 2000). Mais il est possible que l'adhésion ait pour but (plus naturel cette fois) de trouver des amis. De même, le travail portant sur les attitudes mentales sociales, effectué par Panzarasa et al. (1999) vise a priori à modéliser l'influence d'un agent sur un autre. Là aussi, nous sommes dans un cadre cognitif. La notion d'influence telle qu'elle est présentée dans ce travail introduit cependant à la base, une relation humaine qui a pour but d'attirer la sympathie d'un autre agent (en acceptant son influence), d'autant plus que, se faire renvoyer peut être ressenti émotionnellement comme un rejet (Andriamasinoro et Courdier 2002b). Déjà dans ces deux exemples, nous sommes dans un environnement social (humainement parlant) et de ce fait, nous nous situons progressivement dans un cadre plus naturel. L'aspect cognitif montré au départ peut ainsi " dissimuler " divers paramètres qui sont liés aux motivations naturelles (ici, sociales) de l'agent lui-même. Mais ces paramètres n'ont jamais été réellement explicités.

Dans l'ensemble, il demeure vrai que l'intégration de la motivation naturelle dans le cognitif reste à un niveau peu avancé.

---

[19] Le désir naturel fait référence à la notion d'attraction naturelle vis-à-vis d'une entité (objet ou personne). La motivation environnementale de Ferber (1997) se rapproche de cette définition.

II. 4 Les modèles hybrides et la motivation naturelle

Nous terminons le parcours des trois catégories de modèles par celui des agents hybrides. Et comme dans les deux premières catégories, nous essayons également dans cette section de dégager la présence de la notion de motivations chez ces modèles d'agents.

II. 4. 1 Principe de base de l'hybridisme

Bon nombre de chercheurs ont argumenté que ni un comportement totalement délibératif ni un comportement totalement réactif ne s'avère approprié lorsqu'il s'agit de créer un agent. En effet, n'agir que d'une manière réactive, c'est ne pas considérer un objectif global que l'on pourrait se fixer et se donner ensuite les moyens pour y parvenir. C'est une absence de prévision des actions, ce qui pourrait arriver à une instabilité du système. D'un autre côté, n'agir que d'une manière délibérative, c'est se restreindre à une suite d'actions prédéfinies, ne tenant pas compte des situations instantanées qui se produisent. L'idéal est de pouvoir basculer entre les deux comportements en fonction des circonstances. C'est dans ce but que les chercheurs ont été motivés à étudier l'hybridisme.

Le précepte adopté pour la modélisation du comportement hybride consiste à intégrer les facteurs réactifs (et principalement la dynamique de l'environnement) dans une planification issue d'une délibération, que l'agent doit initialement accomplir. A la base de ce plan, l'agent dispose d'une certaine motivation se situant dans son état mental, à savoir la réalisation de son but via un plan. Un exemple de problème traité consiste à étudier comment aller d'un point A vers un point B avec les contraintes suivantes :

- entre les deux points, il existe des obstacles, que ceux-ci soient connus ou imprévus (Ferguson 1992a, Wehrle 1994, Au et al. 1998, Mavromichalis et

Vouros 2000, ...). Cette situation nécessite une surveillance continue de l'environnement par l'agent ;
- l'environnement contenant ces obstacles peut être centralisé (Kurihara et al. 1997) ou distribué (Moulin et al. 2001) ;
- l'agent peut exécuter son plan, soit individuellement (Au et al. 1998), soit dans un contexte de collaboration (Tambe et al. 1999) ;
- étant cognitif (et donc possédant des cognitons[20]), l'agent est capable de raisonner sur le monde dans lequel il se trouve. Cependant, il est possible qu'il n'ait qu'une information partielle à propos de ce monde.

Les références présentées ci-dessus montrent bien que beaucoup de travaux, suivent la forme hybride telle qu'elle est montrée ci-dessus. Cependant, la plupart de ces travaux se focalisent davantage leur attention sur des aspects comme la structure d'un plan, d'une organisation, etc. plutôt que sur l'aspect hybride même du modèle (c'est-à-dire étude du réactif / cognitif). C'est par exemple le cas de Au et al. (1998) dont la contribution concerne plutôt le domaine de la planification que de l'hybridisme. Dans cette section, nous ne mettons en évidence que les modèles utilisés dans des travaux visant réellement à contribuer dans l'état de l'art sur l'hybridisme. Cependant, dans l'analyse synthétique (Section 0), les deux types de modèles seront pris en compte.

II. 4. 2 Exemples de modèles

D'une manière générale, les architectures hybrides sont basées sur des couches hiérarchiques. L'approche à base de couches est idéale dans la mesure où il est possible d'exprimer des fonctionnalités spécifiques à chaque couche durant la conception et l'exécution du système en temps réel. Les couches basses sont en contact direct avec l'environnement, et les couches hautes constituent la zone de réflexion ou de délibération. Au niveau réactif, un agent exécute les actions élémentaires, en fonction de ses perceptions et des événements reçus. Au niveau cognitif, un agent peut exécuter des plans individuels ou collectifs pour atteindre ses buts lesquels ne sont pas toujours triviaux.

D'autres couches peuvent s'y ajouter comme par exemple un troisième niveau qui englobe le mécanisme permettant de gérer la partie sociale et interactionnelle de l'agent, ou encore la zone de données que l'agent utilise dans son comportement (représentation

[20] Voir Section II.3.1 à propos des cognitons

symbolique du monde, etc.). Les modèles qui suivent en sont des exemples. Chacun de ces modèles a sa propre spécificité (ex : modélisation de l'état mental, gestion du choix de comportement, etc.). Ce qui nous intéresse ici, c'est comment l'aspect hybride du modèle est géré.

Touringmachines (Ferguson 1992)

Ferguson (1992a, 1992b) a développé une architecture hybride appelée TOURINGMACHINES (Figure II.9). Cette architecture se compose :

- d'un sous-ensemble de perceptions et d'actions (les deux se connectant directement à l'environnement),
- et de trois couches de contrôle : la couche *réactive*, la couche de *planification*, et la couche de *modélisation*. Chaque couche est constituée de processus d'activités indépendants, qui s'exécutent d'une manière concurrente.

La *couche réactive* génère les actions potentielles en réponse aux événements qui arrivent trop rapidement pour que d'autres couches puissent les traiter. Cette couche est implémentée comme un ensemble de règles de *situation-action*, dans le style de l'architecture de subsumption de Brooks (Section II.2.3, page 31).

La *couche de planification* construit des plans et sélectionne des actions à exécuter afin de réaliser les buts de l'agent. Cette couche contient deux composants : un planificateur et un centre de mécanisme d'attention. Le planificateur intègre la génération et l'exécution de plans, et emploie une bibliothèque de plans partiellement élaborés, ainsi qu'une carte topologique du monde, afin de construire les plans qui accompliront le but principal de l'agent. Le but du centre du mécanisme d'attention est de limiter la quantité d'information que le planificateur doit traiter, et ainsi améliorer son efficacité. Elle procède par le filtrage des informations qui proviennent de l'environnement et rejette celles qui ne sont pas pertinentes.

La *couche de modélisation* englobe les représentations symboliques de l'état cognitif des autres entités dans l'environnement de l'agent. Ces modèles sont manipulés afin d'identifier et de résoudre des conflits de buts, des situations dans lesquelles un agent peut ne plus réaliser ses buts, en raison de l'interférence inattendue.

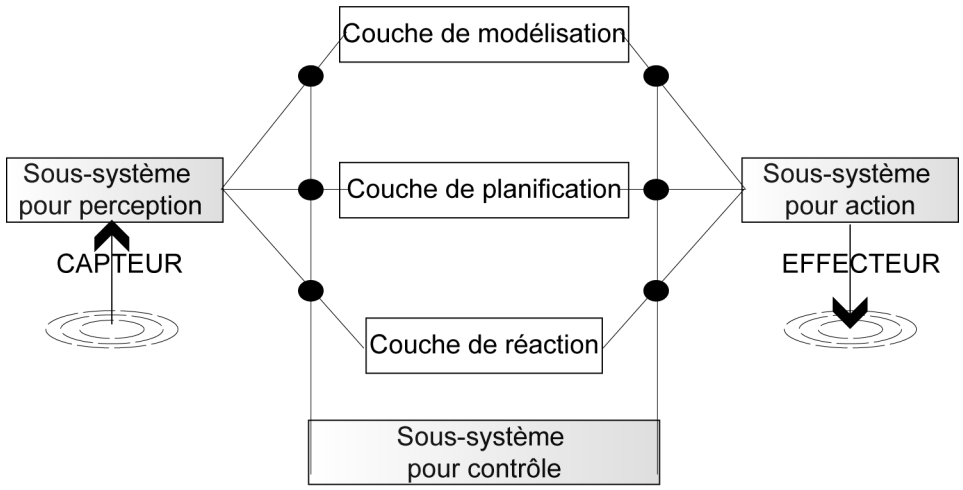


Figure II.9 : L'architecture Touringmachines de Ferguson (1992).

Les trois couches peuvent communiquer entre elles par l'intermédiaire de messages. Elles sont encastrées dans un cadre de contrôle dont le rôle consiste à négocier entre les couches pour traiter les éventuelles propositions contradictoires d'actions provenant des différentes couches. Pour cela, ce cadre de contrôle utilise des règles de contrôle.

Interrap (Müller 1994)

Comme TOURINGMACHINES, l'architecture INTERRAP[21] de Jörg Müller (1994) est aussi une architecture à base de couches (Figure II.10). Elle contient principalement deux couches horizontales : une couche contenant des bases de connaissances (KB pour Knowledge Base), et une autre contenant les divers composants de contrôle (CU pour Control Unit). Chacune de ces couches est encore subdivisée en plusieurs couches verticales, chacune de ces dernières représentant successivement un niveau d'abstraction plus élevé par rapport à celle du dessous[22]. En outre, chaque couche dans le CU interagit avec celle des KB à leur niveau respectif.

Au niveau le plus bas du CU se situe le composant qui contrôle l'interface avec le monde (WI pour World Interface), ainsi que le modèle du monde associé (WM pour World Mo-

[21] INTERRAP signifie INTEgration of Reactive behavior and Rational Planning

[22] Nous avons aussi remarqué une forte similitude structurelle de ce modèle avec celui de Glaser et Morignot (1997) nommée COCOMAS.

del) dans la base de connaissances. Comme son nom l'indique, le WI contrôle l'interface entre l'agent et son environnement, et traite ainsi les actions, la communication et les perceptions.

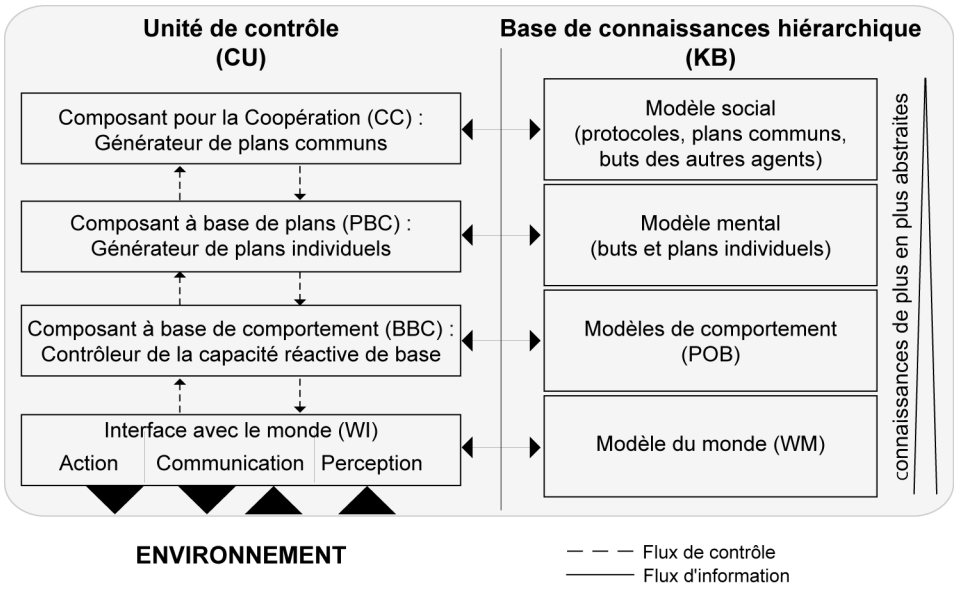


Figure II.10 : L'architecture d'un agent Interrap de Müller (1994).

Au-dessus du WI se trouve le composant à base de *comportement* (BBC pour Behaviour-Based Component). Ce composant a pour rôle d'implémenter et de contrôler la capacité réactive de base de l'agent. Il manipule un ensemble de modèles de comportements (POB pour Pattern Of Behaviours). Un POB est une structure contenant :

- une condition préalable qui définit le moment où le POB doit être lancé,
- diverses conditions qui définissent les circonstances dans lesquelles le POB doit être considéré comme ayant réussi ou ayant échoué,
- une post-condition,
- et un corps exécutable, qui définit quelle action devrait être entreprise si le POB est exécuté. L'action peut être une primitive ayant pour résultat un appel au WI, ou peut impliquer un appel à une couche de plus haut niveau pour générer un plan.

Au-dessus du BBC se trouve le composant à base de *plans* (PBC pour Plan-Based Component). Ce composant contient un planificateur qui peut produire des plans d'agents

individuels en réponse aux demandes du composant BBC. Le KB de cette couche contient un ensemble de plans, y compris une bibliothèque.

La couche la plus élevée est le composant pour la *coopération* (CC pour Cooperation Component). Ce composant peut générer des plans communs qui satisfont les buts d'un certain nombre d'agents, en élaborant des plans choisis à partir d'une bibliothèque de plans. Ces plans sont produits en réponse aux demandes du PBC ci-dessus.

Dans INTERRAP, le contrôle est à la fois géré par les données et les buts de l'agent. La partie perceptuelle est gérée par le WI, et s'avère être la conséquence d'un changement dans le WM. En raison de ces changements, divers POB peuvent être activés, abandonnés, ou exécutés. Suite à l'exécution du POB, le PBC et le CC peuvent être appelés pour générer des plans et des plans communs respectivement afin de réaliser les buts de l'agent.

Mrr-planning (Kurihara et al. 1997)

L'architecture MRR -planning (pour Multi-agent Real-time Reactive Planning) développée par Kurihara et al. (1997) met un accent particulier sur le basculement de l'agent entre son comportement réactif et délibératif[23], tout en tenant compte de l'état de l'environnement ainsi que du facteur *temps* dans l'exécution des plans.

Le modèle MRR-planning a été évalué sur une version étendue du " TileWorld " (Pollack et Ringuette 1990) dont le but est de simuler un environnement très dynamique et non déterministe (c'est-à-dire que le prochain état de l'environnement ne peut pas être déterminé par son état actuel). Pour rappel, le " TileWorld " est système composé d'un échiquier avec des cases vides ou contenant des obstacles ou des trous, et un agent qui transporte un certain nombre de tuiles. L'agent doit mettre toutes les tuiles dans les trous, avec le minimum de déplacements possibles. Les trous et les obstacles changent aléatoirement de position. Dans sa version étendue (c'est-à-dire appliquée au MRR-planning), des robots ennemis ainsi que des zones de recharge d'énergie ont été ajoutés. Par ailleurs, chaque trou contient un score qui est variable en fonction des trous. La finalité est de faire le maximum de scores en parcourant le maximum de trous, tout en tenant compte de divers autres buts : éviter les obstacles, fuir les robots ennemis, se

[23] Rappelons-nous la discussion portant sur le concept de " délibération " présentée dans cette thèse à la page 29.

recharger en énergie lorsque cela s'avère nécessaire, ou encore déposer ou prendre une tuile qui apparaît soudainement près du robot. Ces buts sont réalisés dans un cadre de *planification réactive* (c'est-à-dire permettant de répondre au changement dynamique de l'environnement) alors que la *planification délibérative* est adoptée quand le robot dispose du temps nécessaire. Ici, cette planification consiste à trouver la route la plus appropriée (optimisée) pour faire un maximum de scores.

MRR-planning est caractérisé par trois agents (Figure II.11) : un agent comportement (`B-agent` pour Behavior agent), un agent de planification (`Pagent` pour Planning agent) et un agent sélecteur de comportement (`BSagent` pour Behavior Selection agent). L'algorithme consiste en :

- un cycle d'exécution d'une planification par les `Pagents` et dispersion d'énergie d'activation au sein des `Bagents`, lesquels sont nécessaires pour les `Pagents` dans l'accomplissement de leurs plans ;
- un cycle de sélection d'un `Bagent` (par les `BSagents` qui les contiennent) en fonction de l'énergie d'activation,
- un cycle d'activation d'un `Bagent` pour exécuter une action.

Dans la figure II.11, il est intéressant de voir le basculement entre le délibératif et le réactif. Ce basculement se fait (ou non) par la comparaison de la somme des activations de chaque `Bagent`, avec un seuil donné. Si le seuil est franchi, on bascule vers une planification réactive. Le basculement réactif/délibératif se fait par l'évaluation par le robot, de la position des obstacles et des autres robots ennemis. Il " sait " la durée du déplacement de ces robots. Le temps est géré par un `tick`, la plus petite unité. L'évaluation temporelle se fait par la connaissance du nombre de `tick` qu'un robot met pour faire un pas. De ce fait, il est possible de connaître approximativement leur position et de décider s'il faut faire une planification réactive ou une planification délibérative.

ICagent (Mavromichalis et Vouros 2000)

Le modèle ICAGENT[24] de Mavromichalis et Vouros (2000) s'est focalisé sur la partie raisonnement des agents hybrides, et plus particulièrement sur la représentation de l'état mental d'un agent lorsque celui-ci se trouve dans une situation où il doit basculer dynamiquement entre la réaction et la délibération. La partie cognitive repose sur

[24] ICAGENT est l'abréviation de Intelligent Collaborative Agent

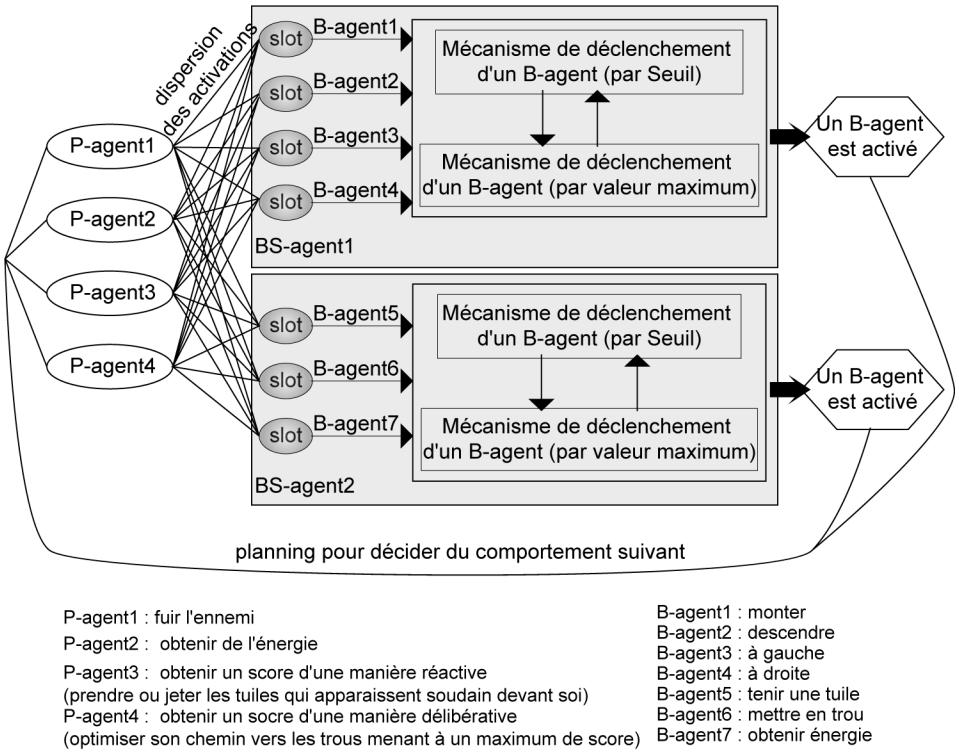


Figure II.11 : Le Mrr-planning évalué sur le cas du " TileWorld " de Pollack et Ringuette (1990).

le modèle BDI largement évoqué à la section II.3.2. Comme dans MRR-Planning, les agents basculent entre les planifications réactives et des planifications délibératives. Le premier cas de planification stipule qu'un agent ajuste son plan durant son exécution, en tenant compte de son état mental et les intentions qu'il a formées pour satisfaire un désir particulier. Le second cas signifie qu'un agent planifie ses actions pour accomplir des buts complexes et considère les désirs alternatifs ou à long terme qu'il peut avoir. Cela implique une réconciliation[25] (Grosz et Kraus 1996, Sullivan et al. 2000) de ses désirs entre eux, mais aussi entre les désirs et les intentions.

Les agents doivent raisonner à propos de leurs désirs et de leurs intentions, tout en focalisant leur attention sur les changements environnementaux. Durant le processus de délibération, un agent évalue des options du plan alors que dans le processus de

[25] Réconcilier les désirs et les intentions, c'est trouver les intentions adéquates pour réaliser les désirs. L'agent va ainsi évaluer toutes les options possibles et former les intentions à cet égard. Si le désir peut être satisfait, il est alors transformé en but.

réaction, l'agent vise à accomplir un but spécifique du moment, en formant l'intention de le réaliser. Le problème du basculement est donc celui de la génération du plan.

Comme MRR-Planning, ICAGENT a aussi été évalué sur le TileWorld de Pollack et Ringuette (1990) sauf que le modèle ICAGENT utilise, en outre, des cognitons (rappel Section II.3.1 pour cette notion).

ICAGENT (Figure II.12) comprend deux modules : l'unité de contrôle de la délibération (DCU pour Deliberation Control Unit) et l'unité de contrôle de l'élaboration et de la réalisation des plans (PERCU pour Plan Elaboration and Realization Control Unit). Existe également une base de connaissances (KB) qui reflète l'ensemble des croyances de l'agent. Le contexte correspond aux croyances dont l'agent dispose actuellement sur ses buts, ses capacités, ses intentions, etc.

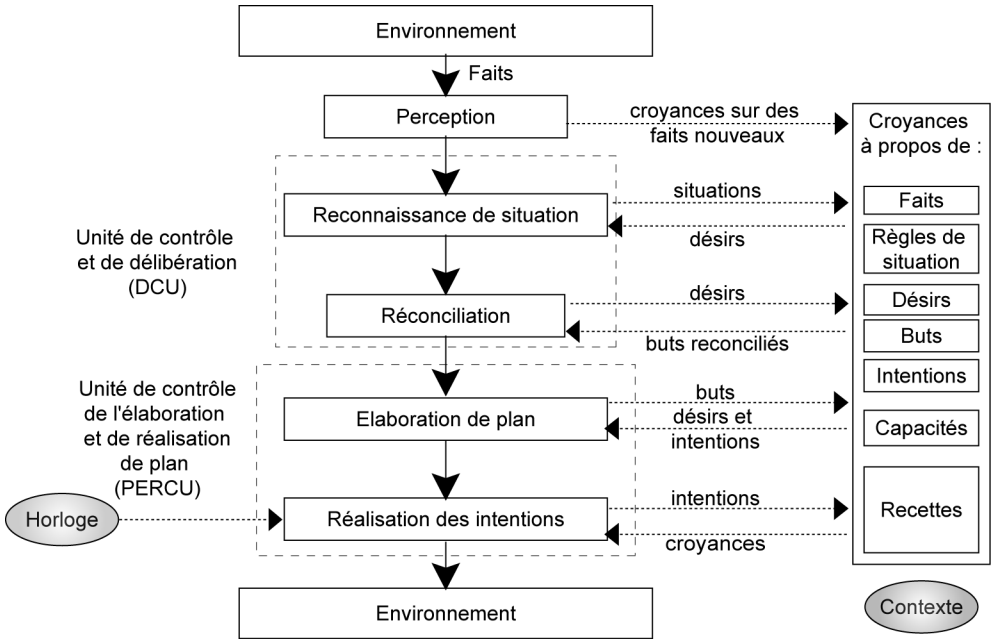


Figure II.12 : L'architecture ICagent utilisée avec une plateforme BDI et appliquée sur le TileWorld.

Nous remarquons dans la figure II.12 une similitude fonctionnelle avec la figure II.10 de INTERRAP : l'unité de contrôle fonctionne en " étroite relation " avec les bases de connaissances. Néanmoins, des différences existent entre ICAGENT et INTERRAP. Ce dernier distingue la délibération de la réaction par la classification des buts (réaction, planification individuelle, planification coopérative) et en distinguant les composants

(couches) mentionnés ci-dessus. Mais contrairement à ICAGENT, il ne considère pas la réconciliation des désirs et des intentions des agents, une réconciliation qui impliquerait le traitement des incompatibilités entre les plans. En outre, INTERRAP n'entremêle pas arbitrairement le comportement délibératif et réactif. En effet, le PBC invoque toujours les procédures du BBC. Il n'existe pas de possibilités qui permettraient, soit d'invoquer des réactions au niveau du BBC, soit d'établir un plan au niveau du PBC lui-même.

En nous référant aux modèles présentés précédemment, la plupart des architectures hybrides sont construites sous forme de hiérarchies de couches, chacune d'elles définissant des fonctions, des décisions ou des comportements spécifiques. Remarquons aussi que la partie réactive de ces modèles se trouve au niveau le plus bas de la hiérarchie (pour faire face à l'environnement) et la partie cognitive (centre de la planification) se trouve à un niveau plus élevé. D'autres couches peuvent éventuellement s'ajouter par la suite. Avec INTERRAP de Muller (1994) par exemple, c'est la couche gérant les éléments sociaux qui se situe au niveau le plus élevé.

II. 4. 3 Analyse synthétique

Modèles hybrides et motivations

Tout comme les modèles BDI, aucune des architectures hybrides présentées n'explicite la notion de motivation naturelle. Elles intègrent plutôt directement le mécanisme de gestion de comportement. Même lorsque nous regardons au niveau du réactif, l'aspect motivation semble être " oublié ". Les concepts introduits dans les agents réactifs (pulsion, etc.) ne sont pas identiques à ceux introduits dans la partie réactive d'une architecture hybride. Pour cette dernière, nous parlons généralement de *planification réactive* c'est-à-dire de la manière dont un plan initial sera réalisé dans un temps un peu plus court en raison d'événements survenant dans l'environnement.

De même, la modélisation de l'environnement, importante, comme nous le savons, pour le comportement réactif, suit ce même constat. Dans un environnement physique, la gestion de l'évitement d'obstacles se fait pour le cas hybride par une simple mise en place de règles d'évitement posées par le concepteur telles que `tourner` avec une vitesse et un seuil de déviation donnés, etc. L'obstacle lui-même est considéré comme un simple objet que l'agent doit éviter mais ce dernier n'intègre aucun aspect de *répulsion* par rapport à cet obstacle, contrairement à la modélisation faite sur les animats. D'une

manière plus générale, la notion d'attraction/répulsion envers un objet n'est pas encore prise en compte dans les modèles d'agents hybrides.

Cependant, un autre angle d'analyse peut aussi être donné. En fait, cette notion d'évitement d'obstacles laisse quand même entrapercevoir une présence " floue " de la motivation naturelle dans les modèles hybrides. En effet, l'évitement est une manifestation de l'instinct de préservation. Prenons par exemple le cas des modèles qui prennent le mouvement des avions comme cadre de simulation (Zeghal 1994, Tambe 1996). Leur scénario contient un évitement entre deux avions qui se rencontrent, ou encore une attaque ou une fuite face à un ennemi qui se rapproche. Ces phénomènes nous rappellent exactement le comportement des animats mais à un niveau plus élevé. Le point commun entre les deux niveaux de scénarios est que leur comportement est à la base guidé par leurs instincts naturels. Un pilote d'avion n'évite pas un autre pilote parce que tel est le plan de l'équipe. Il le fait en cherchant tout d'abord à se préserver. C'est par la suite, à un niveau supérieur de comportement, que le pilote va savoir *comment* il va éviter l'autre avion (ex : à gauche ou à droite ? à quelle vitesse ?). Mais actuellement, seule la couche supérieure est conceptuellement intégrée d'une manière *générique* dans les modèles hybrides. Pour la couche inférieure, les scénarios reflétant l'existence de la motivation naturelle sont gérés en fonction de l'application.

La gestion du basculement réactif/cognitif

La grande question dans la gestion du comportement hybride consiste à savoir ce qu'il faut intégrer dans le module de décision de l'agent afin que celui-ci puisse basculer " raisonnablement " entre les niveaux réactif et cognitif.

Le *temps* peut déjà être un premier critère valable qu'il est possible d'avancer comme raison d'un basculement. Le facteur temporel implique que l'agent doit traiter un événement dans un délai limité : plus ce délai se rapproche, plus l'agent devra opter pour un comportement réactif. Mais au-delà de ce facteur, il faut reconnaître qu'il n'y a encore aucun accord sur la détermination de l'ensemble des *paramètres* menant à l'un ou l'autre des deux comportements. Wooldridge (2002) stipule que la partie réactive semble avoir une priorité sur la partie cognitive. Mavromichalis et Vouros (2000) semblent le confirmer en avançant que la partie cognitive n'est qu'un plan initial de base permettant de réaliser le but fixé, mais la réalisation des détails du plan doit se faire d'une manière

réactive c'est-à-dire en fonction de l'environnement tout en gardant comme repère le but fixé par le plan.

Malec (2000) considère que le basculement est fonction des tâches ou des fonctions qui sont prédominantes pour l'agent, ainsi que l'architecture que le concepteur pense utiliser (faire connaître les préférences du concepteur). De ce fait, l'espace des choix possibles d'architectures est limité par les décisions initiales de conception : une fois l'architecture choisie et les langages de représentation définis, le créateur se doit d'obéir aux contraintes de cette architecture. D'autant plus que toujours selon Malec, une architecture assume habituellement un équilibre fixe entre la réactivité et la délibération, c'est-à-dire que les fonctionnalités à implémenter aux deux niveaux doivent être prédéfinies. Cependant, cette démarche ne permet pas la prise en compte de phénomènes d'apprentissage ou d'auto-organisation.

Bref, le débat reste ouvert. Une autre tendance qui nous semble intéressante serait de laisser l'agent décider du pourquoi, ou du comment de ce basculement et ce, en fonction cette fois de ses motivations. En effet, même s'il existe un plan défini initialement, ce plan n'est que le résultat d'une motivation. S'il existe entre-temps d'autres plans qui permettent à un autre degré[26] de satisfaire cette même motivation, ce second plan serait adopté et l'agent acceptera les conséquences de la dynamique environnementale qui s'en suivent.

II. 5 Synthèse sur la modélisation de la motivation

II. 5. 1 Comparaison générale

Après une analyse séparée des trois types de modèles étudiés dans ce chapitre, nous présentons ici une synthèse qui est établie à partir d'une vue générale des modèles hybrides actuels présentés.

D'une manière synthétique, nous pensons que les approches menées dans la modélisation du comportement hybride se résument à une succession hiérarchique de trois vues par

[26] Par exemple, un employé peut changer de travail pour une rémunération meilleure, mais la motivation initiale ne change pas : bien travailler pour bien vivre.

rapport à la manière dont l'agent va exécuter son plan courant. En suivant une approche descendante, nous décomposons les trois niveaux de vue comme suit (Figure II.13) :

- la vue jusqu'au niveau 3 signifie rester à un niveau purement cognitif de l'analyse en ne considérant que la description d'un but que l'agent va réaliser via un plan.
- la vue jusqu'au niveau 2 permet de détailler davantage le déroulement réel du plan, notamment avec la prise en compte de la dynamique de l'environnement (gérée par la partie réactive) durant toute la phase de l'exécution de ce plan.
- la vue jusqu'au niveau 1 modélise le comportement de l'agent dans ses moindres détails en tenant même compte des processus naturels qui motivent et guident les agents.

A part la vue horizontale, nous avons également une division verticale de la figure. Cette division résume les différents aspects reliés à l'hybridisme :

- la première colonne présente l'ensemble des processus qui guident le comportement général de l'agent hybride.
- la deuxième colonne montre les deux types de comportement que l'agent hybride adopte : la planification et la réaction face à l'environnement. La dernière colonne présente les concepts (non exhaustifs) qui sont utilisés dans le cadre SMA pour modéliser le processus hybride en général.

La figure II.13 nous permet ainsi de remarquer que du point de vue de la modélisation, l'assemblage du réactif et du cognitif ne forme pas forcément l'hybridisme, en raison notamment du 1^{er} niveau qu'il faut donc intégrer. C'est ce que nous nous proposons de faire dans cette thèse.

II. 5. 2 Le pourquoi de l'état de l'art actuel

Le fait que la notion de motivation ne soit pas explicitée par le concepteur dans son modèle hybride ou cognitif peut nous amener à poser un certain nombre de questions, notamment à la question d'en connaître les raisons. La recherche de ces raisons est importante car si c'est le concepteur qui fait mouvoir l'agent, alors c'est l'autonomie même de ce dernier qui est remise en question.

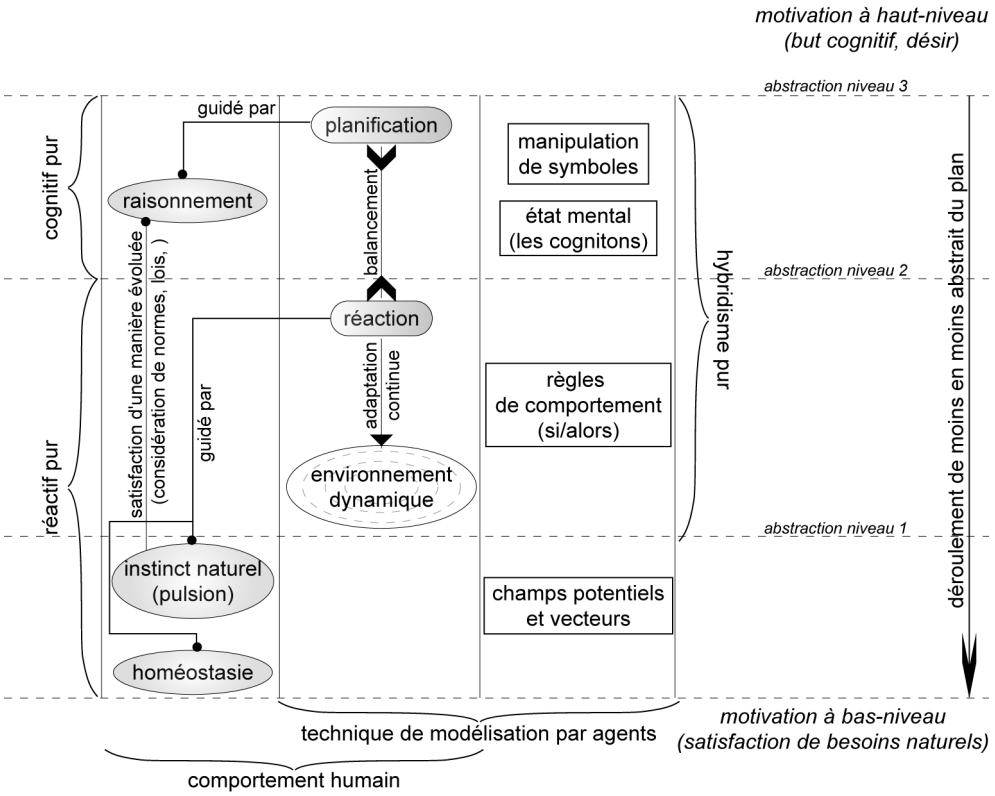


Figure II.13 : Vue globale sur l'état de l'art actuel dans la conception hybride.

Des éléments peuvent être avancés pour expliquer la situation. Par exemple, le concepteur peut supposer que comme l'objet de son travail concernant ces architectures hybrides porte sur la modélisation du comportement, il n'est donc pas nécessaire pour lui d'entrer dans les détails des motivations. Une explication plus psychologique peut aussi être avancée : le *phénomène de projection* que nous expliquons ci-après.

Les entités d'un univers multi-agents sont entièrement le fruit de conceptions humaines, d'une création délibérée dans laquelle les caractéristiques sont étroitement contrôlées. Tout ce qui se passe dans la " tête " de ces agents artificiels est entièrement défini. En réalité, et pour un événement donné, lorsque le concepteur fabrique son agent, il lui attribue la ou les actions possibles qu'il aurait exécutées s'il était à la place de l'agent. Ce phénomène s'appelle une *projection psychologique* (Daco 1965) du concepteur sur l'entité qu'il conçoit. En fait, lorsque le concepteur effectue (consciemment ou inconsciemment) une projection sur son agent, il considère que les besoins qu'il ressent sont aussi ceux de

ses agents (c'est l'effet direct de la projection). De ce fait, il pense qu'il ne lui reste plus qu'à donner à l'agent les actions potentielles à entreprendre face à ces besoins.

L'exemple clair que nous pouvons donner est celui du robot qui évite un obstacle (Brooks 1986, Kodjabachian et al. 2000, Calderoni 2002). Psychologiquement, le robot " n'éprouve " pas réellement le besoin d'éviter l'obstacle, c'est l'humain concepteur qui introduit dans son robot le mécanisme d'évitement en cas d'obstacle (c'est-à-dire que si l'humain était face à un mur, il aurait, lui, évité l'obstacle). L'autre exemple est celui de l'agent fourmi dont l'état de *faim* déclenche la quête de nourriture et par la suite, la manière d'en trouver.

Le problème général de la projection est que le concepteur fait abstraction d'une certaine autonomie des " sensations " de l'agent et décide des actions de cet agent. Certes, nous acceptons le fait qu'en tant que concepteur, l'humain propose les actions possibles à exécuter (sinon comment l'agent pourrait-il s'activer ?). Cependant, l'idée est que ce concepteur " liste " seulement les actions possibles que les agents peuvent exécuter et il appartient à ces derniers de sélectionner l'action adéquate et qui plus est, au moment " choisi " par ceux-ci et non par le concepteur[27]. Il est clair que le phénomène de projection est plus intense lors de l'étude des modèles cognitifs et hybrides que lors de celle des modèles à base d'animats où il y a plus de " relâchement " du naturel dans le concepteur.

Une première proposition serait donc *d'uniformiser* ce " relâchement " dans tous les modèles, d'où l'idée du modèle que nous formulons au chapitre III.

II. 6 Conclusion du chapitre

Ce chapitre a eu pour objectif de comprendre comment, jusqu'à présent, la motivation en général, et la motivation naturelle en particulier, ont été modélisées dans le cadre SMA. Et comme nous travaillons sur des agents hybrides, l'étude s'est donc portée, à la fois, respectivement sur les modèles d'agents réactifs, cognitifs, et hybrides. Des exemples de modèles déjà conçus nous ont ainsi permis de savoir à quel point le concept de motivations naturelles n'a été principalement mis en évidence au niveau générique que dans les agents réactifs, et notamment les animats, alors que le concept de naturel existe

[27] Cette sélection peut passer par des étapes intermédiaires comme l'apprentissage.

également dans les autres types d'agents. Il convient donc, à la suite de ce constat, de passer réellement à une phase où la mise à un niveau générique de la motivation naturelle dans les agents autres que réactifs doit être mise en œuvre. Dans le chapitre III, nous élaborons une première ébauche de proposition de modèle liée à cet objectif.

Chapitre III

Proposition d'un modèle

Suite aux analyses faites dans le chapitre II, nous apportons dans ce chapitre une première de proposition à mettre en œuvre dans la modélisation de la motivation. La section III.1 définit cette proposition. La section III.2 introduit cet ensemble de propositions en présentant tout d'abord la pyramide du psychologue Maslow, cette pyramide étant le modèle de départ de notre travail. La section III.3 montre par la suite comment cette pyramide est adaptée dans un cadre de modélisation agent, tout en respectant l'état de l'art, aussi bien dans un cadre psychologique que SMA.

Ce chapitre a pour but d'énoncer le modèle et de progressivement introduire la formalisation de sa partie générique. C'est dans le chapitre IV que, par la suite, l'ensemble des formalisations est détaillé.

III. 1 Cadre de proposition

L'idée générale de la proposition consiste à expliciter la motivation naturelle même dans les agents hybride et cognitif. Cette motivation y sera alors prise en compte autant que dans les modèles animats actuels, aboutissant à une uniformisation de cette partie naturelle dans tous les types d'agents. Cette idée d'uniformiser vient du fait que l'humain provient d'abord biologiquement de l'animal. Même si par la suite il est cognitif, ce sont les notions telles que pulsion, instinct, etc. qui doivent être à la base de tout comportement[1].

[1] Revoir aussi l'Iintroduction Générale de cette thèse.

Concrètement, nous proposons d'établir un point de liaison entre cette partie " haute " de l'agent (en partant des modèles BDI existants) et cette partie " basse " (fondée sur les animats). Il s'agit de " connecter " la notion de " désir " (ou but) provenant de la couche haute vers celle de " motivation " située dans la couche basse. Cette connexion est envisageable dans la mesure où finalement, le désir tel qu'il est conçu dans BDI est aussi une motivation mais relative à des actions de " haut-niveau " (c'est-à-dire cognitif). C'est la relation entre les deux niveaux de motivations qu'il faut ensuite établir.

A propos des agents hybrides qui intègrent des notions de plan et d'organisation, la partie réactive va aussi intégrer cet aspect naturel. Nous n'allons donc plus abstraire cet aspect derrière de simples règles de type `si/sinon` venant de la part du concepteur. Conformément à l'analyse de la figure II.13 (page 60), l'agent agira toujours en toute connaissance de cause (ex : `éviter` =manifestation de l'instinct). C'est par rapport à la motivation que se fera également le basculement entre le réactif et le cognitif. En outre, par rapport aux architectures hybrides présentées à la section II.4.2 (INTERRAP, ICAGENT, ...), nous introduisons au travers de la pyramide de Maslow, un modèle à base de couches de motivations, composé de besoins et non plus de comportements ni de fonctions. Nous sommes donc plus axés sur la partie motivationnelle que sur la partie décisionnelle du système conatif de l'agent.

Dans la partie animat, la proposition ne concerne plus le concept de motivations naturelles (puisque celui-ci est déjà présent). L'idée serait plutôt de fournir dans la mesure du possible des critères additionnels dans le mécanisme de sélection de comportement. L'ensemble des critères fournis dans ce mécanisme de sélection doit être générique (ne dépend pas de l'application) mais suffisamment poussé pour permettre à l'utilisateur du système de se préoccuper au minimum des stratégies de sélection. Pour ce faire, le modèle devra fournir les règles de positionnement de chaque motivation. Un algorithme bien défini au sein de l'agent[2] manipulera ensuite en permanence l'ensemble des motivations.

La mise en œuvre de notre proposition se fait au travers de l'adaptation et de l'extension d'un modèle reconnu dans le domaine de la psychologie, à savoir la pyramide des besoins conçue par le psychologue américain Abraham Maslow (Maslow 2000). A partir de ce modèle, nous construisons un modèle informatique supportant les concepts abstraits énoncés précédemment. Nos motivations dans l'adoption de ce modèle sont d'ordre pu-

[2] C'est cet algorithme qui représente la proactivité de l'agent.

rement informatique et non psychologique, La discussion sur le plan psychologique ne relève pas de notre compétence. Cependant, il est important de travailler à un niveau générique pour " préparer " le modèle afin que celui-ci puisse, dans le futur, être enrichi par les psychologues eux-mêmes ou par d'autres disciplines relatives aux êtres vivants, telle que la sociologie, l'éthologie, etc.

La figure III.1 résume schématiquement notre cadre de proposition.

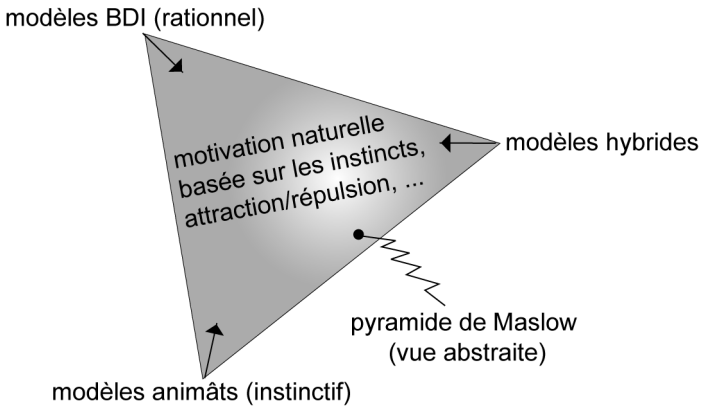


Figure III.1 : Cadre de la proposition conceptuelle : converger les types de modèle étudié au Chapitre II vers le concept de naturel.

III. 2 Maslow : une pyramide psychologique ...

Comme il est dit précédemment, notre modèle agent se base sur la pyramide du psychologue américain Abraham Maslow. Au travers de cette pyramide, le psychologue Maslow illustre le processus de motivation à travers cinq grands besoins humains, schématisés par la figure III.2.

En résumé, nous avons (de bas en haut de la pyramide) :

1. les besoins *physiologiques* (liés à la faim, à la soif, à la respiration, ...),
2. les besoins de *sécurité* (avoir une maison, un emploi, ...),
3. les besoins *sociaux* (être aimé, être intégré, ...),
4. les besoins *d'estime* (être reconnu, ...)
5. et les besoins *d'autoréalisation* (montrer son potentiel, ...)

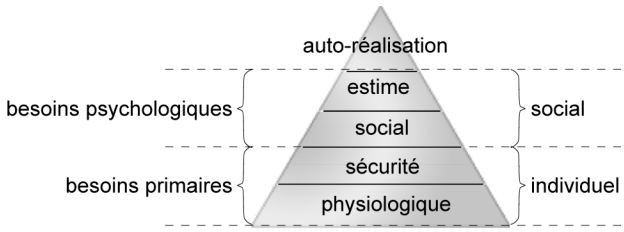


Figure III.2 : Architecture de la pyramide des besoins de Maslow.

Les deux premiers font partie des besoins dits *primaires* alors que les derniers sont appelés besoins *psychologiques*.

Pourquoi est-ce une pyramide ?

L'observation que l'on peut faire devant une pyramide est que pour qu'elle se tienne droite, elle doit avoir une base solide, car une erreur de construction du soubassement entraînera un affaissement de l'ensemble. De même, à chaque étage, une anomalie de construction aura pour conséquence un effondrement des étages situés au-dessus et ainsi de suite jusqu'au sommet. L'existence d'une faille étendue et profonde peut également, à n'importe quel étage de la pyramide, avoir un effet destructeur de l'ensemble. Cette digression architecturale, permet une première approche de l'interdépendance des besoins : grosso modo, il faut que les besoins essentiels soient satisfaits pour qu'apparaissent les besoins du dessus.

Notre ligne de travail suit cette idée de base puis par la suite l'adapte en fonction de nos besoins tout en se conformant à l'état de l'art dans le cadre des SMA.

Remarquons que les cinq besoins tels qu'Abraham Maslow les a définis sont des besoins *abstraits*. En effet, la description qu'il a faite lui-même montre bien que chaque niveau peut théoriquement contenir un nombre infini d'instances de ce besoin abstrait. Du point de vue conceptuel, cette abstraction permet d'aboutir à une classification des besoins, c'est-à-dire à une distinction entre deux besoins $\mathtt{b}_1$ et $\mathtt{b}_2$ n'appartenant pas à une même classe. La particularité de cette classification est qu'elle n'est pas le résultat d'une simple hypothèse conceptuelle de notre part dans la mesure où l'étude de la pyramide provient d'une étude faite sur le monde réel.

III. 2. 1 Les cinq niveaux de besoins de Maslow

Dans les sous-sections qui suivent, nous détaillons chaque niveau de la hiérarchie présentée précédemment. En réalité, Abraham Maslow a réalisé ses études sur les humains. Cependant, Maslow (1998) lui-même reconnaît l'homme comme étant un animal en quête de besoins. Cette acceptation étant venue du même auteur, nous nous sommes donc " autorisé " à étudier aussi la pyramide dans le cadre des besoins des animaux. A propos de ces derniers, nous présentons nos arguments principalement à partir de l'ouvrage de Bonabeau et Theraulaz (1994) qui constitue une référence en la matière.

Les besoins physiologiques

Les besoins physiologiques sont communs à tous les vivants (animal/humain). Ils sont à l'origine des échanges nécessaires avec l'extérieur, ce qui permet au vivant et à l'homme en particulier de perpétuer sa vie. Ils sont liés à l'instinct de survie : besoin de boire, de manger, de respirer, de se reposer, de procréer, etc. Maslow les place à la base de sa pyramide des besoins. Un manque, une privation de ces besoins aura obligatoirement un impact sur les autres besoins, car la construction des étages supérieurs est alors impossible. *Si nous ne pouvons pas nous nourrir par exemple, nous oublions tous les autres besoins*.

L'homéostasie définie à la section I.2.3 de ce document explique en grande partie les besoins physiologiques. Celui qui a vraiment faim change de comportement : ses perceptions changent, il sent et voit la nourriture mieux qu'auparavant, sa mémoire de la nourriture s'améliore et il est capable de se souvenir des bons plats avec une acuité nouvelle. Ses émotions changent. En bref, la nourriture devient presque son seul désir, son seul besoin. Un autre besoin physiologique est le besoin de respirer. Si quelqu'un est privé d'air, il fera tout pour en avoir. Les autres besoins disparaissent du champ de sa conscience : il veut respirer à tout prix.

Les besoins de sécurité

Ils correspondent à cette nécessité de se sentir rassuré, aussi bien affectivement que matériellement. Chez les animaux (ou chez l'instinct animal de l'homme), ce besoin se manifeste sous plusieurs formes : être loin du danger (ex : la proie vis-à-vis du

prédateur), être à l'abri, être protégé par sa famille, être en sécurité dans un territoire bien délimité. Pour la plupart des humains, ce besoin est exprimé au travers de l'emploi, des économies, de l'assurance, de la famille, etc. Un chômeur, n'ayant pas de sécurité ne pourra pas construire l'étage supérieur. De plus, si ce demandeur d'emploi ne touche pas d'indemnisation, il aura des problèmes pour assurer ses besoins de maintien de la vie et la pyramide humaine s'écroulera. Une grande partie de ce qui entoure l'homme provient de son désir de sécurité : du réfrigérateur aux serrures, en passant par l'armoire pharmaceutique, etc.

Maslow (2000) décrit les besoins de sécurité de la façon suivante : si les besoins physiologiques sont relativement bien satisfaits, une nouvelle série de besoins surgit : sécurité, stabilité, protection, libération de la peur, de l'anxiété, du chaos ; le besoin de structure, d'ordre, de lois, de limites, d'être protégé par quelqu'un de fort, etc.

A noter qu'il existe des besoins qui remplissent les deux niveaux : besoins physiologiques et de sécurité. Dès que des situations d'urgence ou d'inquiétude se présentent, ces besoins se font plus pressants, par exemple en cas de guerre, de dépression, de vague de crimes, de désorganisation sociale, de catastrophes, etc.

Les besoins d'amour et d'appartenance

Le besoin d'amour, d'affection, d'appartenance consiste à donner et à recevoir de l'affection, à aimer et à être aimé, à aider et à être aidé, à être un membre estimé d'un groupe, etc. Faire partie d'un groupe, c'est y être considéré comme membre à part entière, c'est partager un certain nombre d'idéaux communs, ou encore, être assuré du soutien du groupe. Lorsque la plupart des gens ont satisfait leurs deux premières séries de besoins, ils auront *faim de relations avec les autres*, faim d'affection, d'amour, d'amis, de compagnon ou de compagne. Ils chercheront à *avoir une place* dans un groupe ou dans leur famille, alors que ces besoins étaient absents lorsqu'ils manquaient de l'essentiel.

L'insatisfaction de ces besoins se ressent au travers de la solitude, du sentiment d'être sans amis, sans racines, de manquer de tendresse et d'amour. Devant le danger, ces sentiments sont exacerbés. C'est la camaraderie " à la vie, à la mort " : des soldats devant le même ennemi, la force des gangs, de la police, des groupes terroristes, etc. L'origine de la " mode " et la formation des groupes, sont liées à ces besoins.

Chez les animaux, le besoin d'appartenance est fort chez les espèces sociales (abeilles, fourmis, certains oiseaux migrateurs, etc.) et est plus faible dans d'autres catégories (chiens, chats, etc.). Mais quelle que soit la catégorie citée, la présence des autres (que ces derniers soient des congénères ou non) demeure toujours un besoin pour éviter la solitude. Un cas particulier de relation est par exemple celui de l'homme et du chien, l'un ayant besoin de la compagnie de l'autre[3].

Les besoins de reconnaissance

Le besoin de reconnaissance et d'estime se manifeste par le besoin d'un statut particulier, d'être reconnu, de réputation, en particulier dans les groupes auxquels nous appartenons. A cela s'ajoute le besoin de respecter et d'être respecté par la société.

Maslow décrit ce besoin de la manière suivante : il y a tout d'abord le désir de puissance, de réalisation, d'adéquation, de maîtrise et de compétence, de confiance face au monde, d'indépendance et de liberté. Ensuite vient le désir de réputation ou de prestige à tel point que l'on voit des personnes commettre des meurtres uniquement pour voir leur nom imprimé à la une des journaux.

Chez les animaux, c'est le chef de la troupe qui éprouve surtout ce besoin. Il exige le respect et entre en guerre avec celui qui ose " braver ses ordres ". Une mère éprouve aussi ce besoin de respect venant de ses petits. Une société animale a aussi besoin que ses territoires soient respectés. C'est pour cela que cette société les défend lorsque des invasions extérieures surviennent.

Les besoins de réalisation de soi.

Le besoin de réalisation de soi correspond au sentiment d'acquérir tout ce dont nous sommes capables, d'accomplir au mieux une tâche qui est à notre mesure. Le besoin d'harmonie, d'ordre, de beauté, d'équilibre. Pour un homme, il est lié à l'insatisfaction de réaliser son potentiel. Les besoins de sécurité, de rapport, d'amour, de respect, et d'amour-propre, etc. sont satisfaits à tel point qu'il est libre de développer pleinement son potentiel et d'être ouvert à de nouvelles expériences. Les problèmes deviennent des moyens de développer ses potentialités, plutôt qu'un fait irritant. Souvent, on parvient

[3] Nous faisons ici abstraction des raisons de cette relation.

partiellement à ce stade. Pendant des vacances, par exemple ou pendant des loisirs qui nécessitent beaucoup de dépenses.

Il existe beaucoup de synonymes à l'expression " l'actualisation de soi " : individualisation de soi, réalisation de soi, satisfaction de soi, réalisation totale de son être et de ses potentialités, et ainsi de suite.

Le besoin de réalisation de soi n'existe pas chez les animaux.

A noter que contrairement aux quatre premiers niveaux de la pyramide, ce dernier niveau n'est pas en soi approfondi dans cette thèse. Le niveau 5 est propre à l'humain. Le niveau animal s'arrête au niveau 4. D'ailleurs, comme son nom l'indique, ce niveau constitue la réalisation des autres besoins mais à un degré plus élevé qu'il faut approfondir dans un autre cadre de travail. En quelque sorte, il englobe d'une manière descriptive tous les autres niveaux. Pour ces raisons, nous ne l'exploitons pas dans notre cadre d'études.

Néanmoins, ce niveau va rester présent pour les raisons suivantes :

- certains besoins, quoique purement humains, se situent à ce niveau. A ce titre, il devrait être possible à notre modèle de situer leur degré d'importance lors du traitement du comportement ;
- il fera l'objet de travaux futurs car une fois réellement approfondi, le niveau 5 enrichit considérablement les critères de comportement de l'agent. Cela permettra au modèle Maslow de se rapprocher davantage des réalités ;
- si la capacité des animaux se limite au niveau 4, c'est tout simplement parce que c'est aussi la limite maximale de la catégorie d'instincts des vivants. Le fait que le niveau humain se trouve à 5 et celui de l'animal à 4 n'est donc pas en lui-même une lacune au modèle. C'en est même une caractéristique intrinsèque.

Pour résumer, le niveau 5 sera maintenu mais ne fait pas l'objet d'exploitation proprement dite (en tant que autoréalisation) mais est utilisé lorsque cela s'avère nécessaire pour une cohérence du modèle agent global par rapport à la pyramide réelle.

III. 2. 2 Les raisons du choix de la pyramide

Avant tout, il n'est pas inutile de rappeler que cette pyramide est le résultat d'un travail réalisé par le Psychologue Abraham Maslow à partir d'une étude approfondie qu'il a faite des êtres humains, tout comme les biologistes ont fait des études approfondies sur la société de fourmis. De même que les travaux sur ces fourmis ont ensuite été modélisés par les chercheurs dans le monde des agents, notre idée est également de modéliser les concepts de la pyramide de Maslow pour simuler les comportements humain et animal.

En entrant à présent plus en détail, voici les raisons essentielles pour lesquelles nous avons utilisé la pyramide.

Raison 1 : elle intègre le concept de besoins des agents

Cette Pyramide décrit le concept de besoins, une notion essentielle si l'on veut étudier le comportement d'un agent. Remarquons un début de rapprochement entre la théorie de Maslow et la définition de base même d'un agent, proposée par Ferber (1997) : un agent (indépendamment de son type) est une entité qui est mue par un ensemble de tendances. Ces tendances se présentent sous la forme d'objectifs individuels (ex : besoins d'actualisation) ou d'une fonction de satisfaction (ex : être en sécurité), voire de survie (ex : besoins physiologiques), qu'elle cherche à optimiser. Le comportement de l'entité tend à satisfaire ses objectifs (ici, les besoins à différents niveaux).

Nous défendons le fait que tout objectif d'un agent a un lien avec la satisfaction des besoins dits *abstraits* de Maslow et ce, quel que soit le cadre d'études. En effet, un agent qui entre dans un système n'y est pas " innocemment " présent. Il s'y trouve pour satisfaire (implicitement ou explicitement) des besoins (s'exprimant sous forme de buts ou de désirs) qu'il va manifester au travers d'actions individuelles ou collectives. Par exemple, lorsqu'un agent footballeur (Kitano et al. 1997) agit dans une équipe pour marquer un but, il le fait, non pas parce que voir un ballon au fond des filets est universellement reconnu comme important, mais que le fait de le réaliser lui permet de satisfaire certains besoins plus profonds en lui-même (autosatisfaction, avoir l'estime de ses partenaires, jouir d'une rémunération importante, etc.).

Raison 2 : elle est un modèle hiérarchique orienté besoins

La plupart des modèles hiérarchiques contiennent à différents niveaux, un ensemble d'actions. Celles-ci sont considérées comme des nœuds, des composants, etc., chaque niveau d'actions ayant des niveaux de contrôle associés (Brooks 1986, Müller 1994, Behnke et al. 2000). La pyramide de Maslow par contre est une structure hiérarchique de besoins. Chaque niveau est structurellement indépendant des autres et n'a donc pas besoin de ceux-ci pour être traité. La dépendance se situe au niveau fonctionnel où la caractéristique de la pyramide montre qu'un niveau plus bas doit être traité et satisfait avant de monter à un niveau plus élevé.

La pyramide étant orientée objectif (satisfaction des besoins abstraits), aucune action n'est préalablement connue à chaque niveau. Ceci a pour avantage de pouvoir adapter facilement le système dans divers contextes. En fait, l'objectif de chaque niveau est abstrait et indépendant d'une application particulière. Une même action est valable à différents niveaux.

Prenons le cas d'une personne en train de boire (→action). Elle exécute `boire`, soit pour satisfaire sa soif (→besoin de niveau 1), soit parce qu'elle se trouve dans une cérémonie où la politesse lui impose socialement de ne pas refuser une gorgée : `refuser` impliquerait, pour lui, d'être mal considéré (→insatisfaction du besoin de niveau 4). La place de `boire` n'est donc pas figée à un niveau particulier, tout dépend de son utilité dans le comportement.

Enfin, le nombre de niveaux de la pyramide est fixé à *cinq*. Cela empêche le concepteur de tomber dans un modèle au nombre infini de niveaux hiérarchiques, toujours complexe à gérer.

Raison 3 : elle intègre le réactif/cognitif et individuel/social

Techniquement, la pyramide intègre à la fois les êtres ayant des comportements réactif et cognitif. En voici les justifications. Déjà, la sémantique initiale donnée par Abraham Maslow a été fondée à partir de l'étude du comportement humain (i.e. contenant l'aspect cognitif). Or, nous constatons que bon nombre d'éléments utilisés dans la modélisation du comportement des agents réactifs se trouvent également dans cette pyramide. Prenons au hasard les travaux de Brooks (1986) sur les robots : l'action la plus prioritaire est

celle d'éviter des obstacles (besoins de sécurité pour Maslow) ou celle de récupérer de l'énergie (besoins physiologiques du robot pour Maslow). De même pour les fourmis, le fait d'apporter de la nourriture pour les autres (Bonabeau et Theraulaz 1994) constitue un " désir inconscient " de vivre en collectivité et de réaliser le besoin d'aider, d'aimer et d'être aimé par ses congénères, et plus globalement de la société dans laquelle elles se trouvent (besoins de niveau 3 et de niveau 4 pour Maslow). Cet instinct motivant une vie en collectivité est bel et bien inné chez ces animaux. Nous retrouvons donc bien ces comportements d'agents réactifs à l'un ou l'autre des niveaux de la pyramide sauf au niveau 5. En effet, lorsque l'on regarde attentivement les humains et les animaux, seuls les niveaux 1 à 4 intègrent les besoins des deux espèces, le dernier niveau étant propre à l'humain.

Raison 4 : elle tient compte des catégories de motivations

La classification de motivations, présentée à la section II. 1. 2 par Ferber (1997) et suivie par Calderoni (2002), se retrouve au travers de la pyramide de Maslow :

- les motivations *pulsionnelles* correspondent naturellement aux besoins primaires de la pyramide. Ces besoins proviennent de l'instinct ou de l'homéostasie
- les motivations *perceptuelles* (ou environnementales) n'ont de sens qu'en étant reliées à la satisfaction d'une ou plusieurs motivations pulsionnelles. Elles ne sont pas fonctionnellement " indépendantes ". Par exemple, la découverte d'une nourriture s'associe à la faim à l'intérieur de l'agent.
- les motivations d'ordre *social* se traduisent par une manifestation des niveaux sociaux de la pyramide.
- les motivations *fonctionnelles* sont, selon Ferber, liées à des raisons hédonistes, que les avantages ou les désagréments soient immédiats ou lointains. Nous revenons donc aux motivations d'ordre social évoquées précédemment.

Ces quatre raisons mentionnées, notons cependant qu'elles ont aussi des limites dont il faut tenir compte lors de la conception agent.

III. 2. 3 Les limites de la pyramide

La question qui pourrait se poser est de savoir jusqu'où la hiérarchie proposée dans la pyramide est statique et pertinente pour que nous puissions ainsi la prendre comme modèle de base. Par exemple, est-il toujours vrai que le besoin de manger (physiologique) est toujours plus important que le besoin d'amour ou de respect (niveau 3 ou niveau 4) ?.

En fait, tel n'est pas toujours le cas dans la réalité, une boucle pouvant exister entre les besoins. Maslow lui-même a mentionné plusieurs cas pour lesquels cet ordre n'est pas respecté[(4)]. Par exemple, le rôle social d'un individu peut le motiver à se consacrer davantage au soin de son image (estime) qu'à la satisfaction de sa faim. Ou encore, quelqu'un ayant eu un problème d'amour dans une phase précaire de son développement psychique pourrait plus privilégier celui-ci au détriment d'autres besoins. Ainsi, des facteurs peuvent " troubler " la hiérarchie précédemment adoptée pouvant remettre en cause le modèle lui-même. Nous acceptons l'existence de ces facteurs. Cependant, nous préférons, en tout cas dans un premier temps, ne pas entrer dans ces détails qui risquent de nous entraîner dans des débats psychologiques dans la présente thèse. Les envisager à cette étape de la création pourrait entraîner une incompréhension de notre modèle agent. Au départ donc, nous acceptons que la sémantique et l'ordre des besoins dans la pyramide sont corrects.

III. 3 Maslow : un modèle agent

III. 3. 1 Introduction

Les spécifications de la section III. 2. 1 montrent que sur chaque niveau, il existe d'un côté les besoins communs à l'homme et à l'animal et de l'autre, les besoins propres à l'homme. De même, les besoins communs touchent toujours directement une ou plusieurs parties (physiologiques ou psychologiques) du corps de l'être tandis que les besoins propres à l'homme peuvent concerner des objets extérieurs. Prenons simplement le cas de la serrure à installer à la porte d'une maison. Pour une personne, ce besoin d'avoir une serrure (qui se manifeste par le plan de l'installer) vise en fait à satisfaire son besoin interne de sécurité. Si la serrure se brise, alors le besoin de sécurité est menacé. Par contre,

[4] Voir Maslow (2000) à partir de la page 264.

à la suite du même événement, un animal qui se trouverait dans cette même maison ne sentirait aucune menace même si la serrure se brisait. La raison en est qu'il n'y a aucun lien de cause à effet établi par l'animal entre son besoin interne et l'existence de la serrure. Il en serait éventuellement de même pour un visiteur qui ne fait que passer dans la dite habitation[(5)].

III. 3. 2 Les besoins pyramidaux PN

Les types de besoins

C'est dans cette existence de divers aspects du besoin et dans cette coexistence de l'animal et de l'homme sur chaque niveau de la pyramide (notée désormais par Π) que nous établissons les faits suivants :

1. il existe les besoins communs à l'homme et à l'animal. Ce sont des besoins *innés* c'est-à-dire des besoins qui ne sont aucunement issus de leur activité courante ;
2. la satisfaction de ces besoins innés ne pourra se faire qu'à partir de l'environnement dans lequel l'agent vit. Cette satisfaction dépendra donc de l'environnement de l'application. Exemple, pour satisfaire leur faim respectif, un koala va chercher une feuille d'eucalyptus aux alentours tandis qu'un être humain pourra aller au restaurant situé dans la rue en face de sa maison. Le point commun entre les deux, c'est cette satisfaction de la faim.
3. ces besoins innés sont à satisfaire, soit en *temps réel* (c'est-à-dire à chaque moment de la vie de l'agent), soit de manière *prévisionnelle* (c'est-à-dire liée à des besoins d'anticipation qui visent à prévenir l'insatisfaction éventuelle à moyen ou long terme de ces besoins innés).
4. le point commun entre le besoin de satisfaire en temps réel et le besoin d'anticiper, c'est qu'ils sont tous les deux liés à la base par la satisfaction des besoins innés.

---

[5] Le conditionnel est de rigueur car le comportement du visiteur pourrait être différent. Par exemple, en constatant la serrure brisée, il pourrait conseiller à ses hôtes de faire immédiatement changer la serrure. Toutefois, ce comportement fait encore intervenir plusieurs paramètres que nous n'approfondissons pas dans cette thèse tels que : la connaissance (de la raison de la brisure), le degré de relation entre le visiteur et l'hôte, etc.

En nous basant sur ces quatre faits, nous pouvons essayer de recenser des limites à la classification des besoins de cette pyramide. Voici quelques unes qui nous semblent identifiables pour la thèse :

- la hiérarchie de Maslow est statique et ne reflète pas forcément la réalité dans la séquence des besoins à satisfaire ;
- chaque niveau de la pyramide est trop abstrait ;
- la limite entre certains niveaux est floue (ex : physiologique et sécurité, social et estime, ...)

Les propositions pour réduire ces limites sont les suivantes :

- ajouter une décomposition *horizontale* des besoins à la décomposition verticale initialement définie par Abraham Maslow, pour avoir une sélection plus dynamique des besoins à chaque niveau ;
- commencer à détailler les besoins identifiables à chaque niveau de la pyramide tout en restant à un niveau générique.

C'est à partir de cette proposition d'extension de la pyramide que nous mettons en œuvre notre modèle agent que nous avons également nommé MASLOW[6]. Cette décomposition horizontale se base sur l'existence des trois *types* de besoins suivants :

- les LN (pour Low-Need ou besoins de bas niveau), initialement introduits dans (Andriamasinoro et Courdier, 2002a), et qui correspondent aux besoins innés. Ce sont les LN qui constituent le point commun entre tous les agents MASLOW. Ils existent indépendamment des agents, et de l'application courante. Ils se trouvent donc à un niveau générique de notre modèle[7].

- les HN (pour High-Need ou besoins de haut niveau) qui font référence au besoin d'anticiper la satisfaction (ou plutôt l'insatisfaction) des LN. Chaque agent pouvant avoir sa manière d'organiser cette anticipation en fonction de son environnement, les HN dépendent de l'application.
- les Medium-Need (pour MN ou besoins de niveau intermédiaire) qui se trouvent entre les deux besoins ci-dessus et correspondent à la satisfaction

[6] Notre modèle agent porte le même nom MASLOW que le psychologue. Pour distinguer les deux termes, le modèle agent sera écrit en petites majuscules.

[7] L'acronyme MASLOW donné à notre modèle signifie " Multi-Agent System based on Low-Needs " qui signifie que tout comportement de l'agent est basé sur les besoins de bas niveau.

des LN en temps réel. Pourquoi " entre " les deux besoins ? Tout d'abord, les MN sont très proches des LN. En effet, ils en sont pratiquement la manifestation directe dans le cadre applicatif. Comme nous l'avons dit, les LN sont permanents et demandent à être satisfaits immédiatement, c'est-à-dire en temps réel. Mais pour cela, il faut ensuite se conformer au contexte dans lequel l'agent se trouve. Les MN se trouvent aussi avec les HN car ils dépendent tous deux du contexte applicatif et ils sont basés sur les mêmes ensembles de besoin : les besoins innés, à savoir les LN.

Ces différents types de besoins diffèrent dans leur sémantique mais ils sont tous abstraits dans un concept générique nommé PN pour " Pyramidal Need " ou encore *besoin pyramidal*. Pour pouvoir grouper l'énoncé des spécifications communes à MN et HN dans ce travail, nous les regroupons dans un *pseudo-type* appelé AN (pour " Application Need ") ou les *besoins du niveau application*.

Terminologie importante

Puisqu'il existe des besoins (LN, MN, HN) à divers niveaux de la conception, il existe donc également des motivations liées à ces niveaux. Nous désignons par " motivation naturelle " les motivations associées aux LN et " désir " celles associées aux AN. Le terme désir a été choisi car il est souvent associé avec la notion de but (la satisfaction de ce désir) ce qui est conforme au présent contexte : les motivations liées aux AN ont un but de base : la satisfaction des besoins LN.

Voici quelques exemples faisant intervenir chaque type de besoin :

1. pour un agent X, le besoin " faim " (en général) est un LN physiologique qui peut se manifester à un niveau MN par un besoin d'aller au restaurant (qui est donc le contexte applicatif) et à un niveau HN (pour le même besoin) par le besoin d'aller travailler, ce travail lui permettant de vivre et donc de satisfaire sa faim.
2. pour un agent fourmi, le besoin " faim " (en général) est un LN physiologique qui se manifeste à un niveau MN par un besoin de manger la miette

de pain qu'il trouve sur son chemin, ou à un niveau HN (pour le même contexte) par un besoin d'approvisionner son nid.

3. pour un agent Y, le besoin " être protégé " (en général) est un LN sécurité qui se manifeste à un niveau MN par un besoin (actuel) de boucher le toit (qui fuit) de sa maison en raison de la pluie qui tombe (et qui est en train de s'infiltrer par ce toit), ou à un niveau HN (dans un autre contexte) par le besoin de renforcer la serrure de sa porte pour prévenir d'éventuels voleurs qui s'introduiraient chez lui.

Deux MN, MN_1 et MN_2 peuvent être associés à un même LN (supposons *social*) mais dans des circonstances ou dans des objectifs différents les uns des autres. Par exemple, un MN se manifeste lors d'une discussion avec les collègues du bureau alors qu'un MN' l'est avec des gens (voire inconnus) dans la rue. Ces deux cas sont guidés par le besoin permanent qu'est le besoin social.

Les processus liés aux besoins

En fonction de leur type, chaque besoin est manipulé par un ou plusieurs processus dont des exemples (non exhaustifs) sont présentés ci-après :

- pour les LN, il s'agit de l'instinct et l'homéostasie. Cela semble évident dans la mesure où nous sommes purement au niveau des motivations naturelles.

- comme les HN proviennent d'anticipation, aboutissant à diverses organisations, répartitions de rôles, etc. adoptées par la société, ils sont manipulés par le processus de raisonnement, de planification, de comportement génétiquement préprogrammé[8], etc.
- comme il est dit précédemment, un MN se situe entre les LN et les HN. Ils sont donc liés à tous les processus précédemment énoncés.

[8] Chez les agents réactifs, le cas le plus évident de comportement préprogrammé que nous identifions dans un cadre d'anticipation est celui de l'approvisionnement collectif du nid par les fourmis ou les abeilles (Bonabeau et Theraulaz, 1994). Ce besoin d'approvisionner entre dans le cadre des HN tels que nous le définissons ici à savoir *anticiper*, même si cette anticipation ne provient pas de l'intention de l'agent (le fait que la satisfaction d'un besoin soit intentionnel ou non n'entre pas dans la définition d'un HN).

III. 3. 3 Relation entre les PN

Elle s'énonce comme suit : la satisfaction d'un AN (situé à un niveau applicatif) est *toujours* motivée à la base par la satisfaction d'au moins un LN (situé à un niveau générique).

L'Equation (III.1) donne un premier aperçu de ces divers besoins.

$$\begin{aligned} &\Pi = \{PN\} = \{LN\} \cup \{AN\}\, avec\, \{AN\} = \{MN\} \cup \{HN\} \\ &\forall\, AN \in \Pi, \exists\, LN \in \Pi \,/\, LN = motivation(AN) \end{aligned} \tag{III.1}$$

Dans la figure III.3, et en partant des HN (c'est-à-dire du haut-niveau), nous remarquons ici une évolution progressive du processus vers le naturel.

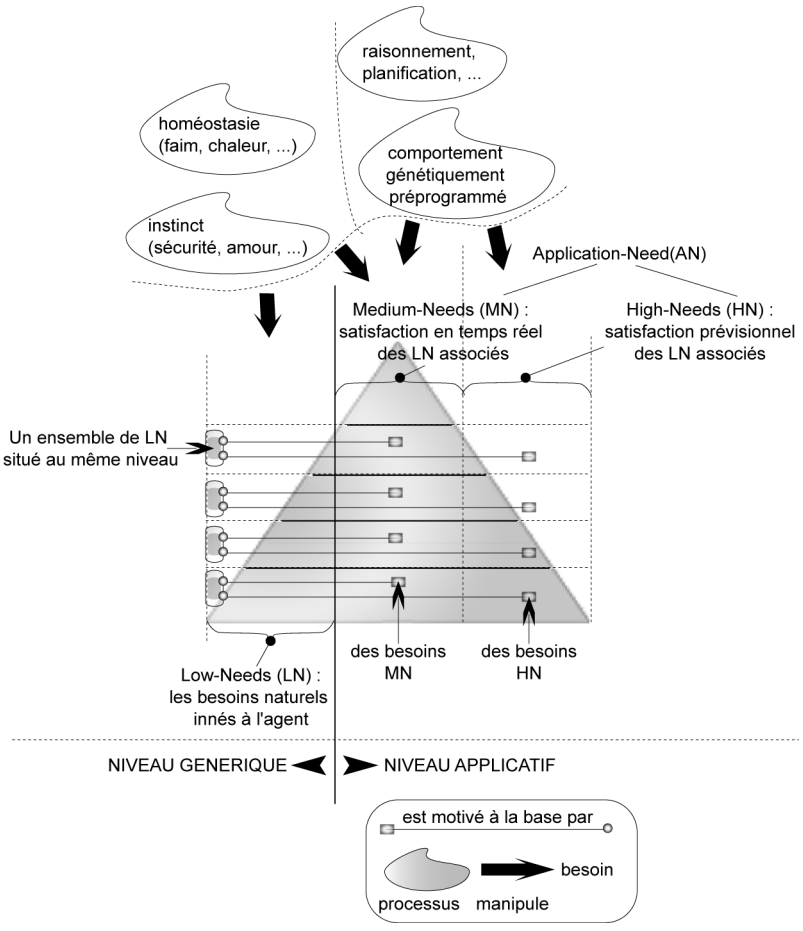


Figure III.3 : La pyramide du modèle agent MASLOW

Chaque PN du modèle possède une place donnée dans Π. Chaque place est identifiée par deux paramètres : *niveau* (coordonnée verticale) et *catégorie* (coordonnée horizontale). Le paramètre `niveau` est identique à celui de la pyramide de Maslow tandis que le paramètre `catégorie` constitue une classification des PN situés à un même niveau. Dans un cadre LN, ces paramètres sont prédéfinis (voir Section III.3.4).

Si nous écrivons provisoirement[9] un PN par $\texttt{PN}_{niveau\,,\,catégorie}$, nous pouvons traduire la relation de base entre un LN et un AN par l'Equation (III.2) à savoir que lorsqu'un AN est motivé par un LN, cela signifie qu'il obtient directement les niveaux et la catégorie de ce LN.

$$\begin{array}{l} soit\ LN_{n,c} \in \Pi, \\ si\ AN_{x,y} \in \Pi / LN_{n,c} = motivation(AN_{x,y}) \Rightarrow x := n\ \wedge\ y := c \end{array} \tag{III.2}$$

Si un AN correspond à *deux* LN, il est *dupliqué* (fonction `dup`) `1 (=2-1)` fois pour que l'autre LN ait sa " copie " (le premier LN étant associé au AN " original "). D'une manière générale, si un besoin AN est motivé par `k` LN, cet AN est dupliqué *k-1* fois et nous avons alors la formulation de l'Equation (III.3).

$$\begin{array}{l} si \exists\, LN_1, ..., LN_k,\ AN_x \in \Pi,\ / LN_1 = motiv_1(AN_x), ..., LN_k = motiv_k(AN_x) \\ \Rightarrow créer(AN_{x2}) = dup(AN_x), ...,\ créer(AN_{xk}) = dup(AN_x) \\ \wedge\, LN_2 = motiv_2(AN_{x2}),\ \ldots,\ LN_k = motiv_k(AN_{xk}) \end{array} \tag{III.3}$$

Dans cette équation, la valeur `k-1` est obtenue par le fait que l'indice des AN dupliqués va de 2 à k (=`k-2+1` duplications)

Deux instances de AN par duplication évoluent en même temps sauf qu'ils ne possèdent pas le couple d'attributs `niveau/catégorie`. L'intérêt de la duplication est de garder la trace du AN si toutefois l'agent ne l'utiliserait plus comme moyen de satisfaire un LN donné alors qu'il le conserverait encore pour d'autres LN.

[9] " Provisoirement " signifie que la notation qui suit est utilisée uniquement dans cette section, et afin de faire comprendre l'idée formelle de la motivation.

III. 3. 4 La liste des besoins génériques LN

Comme nous le savons, les LN se situent à un niveau générique de notre modèle. Mais que signifie exactement ici besoin générique ?

Le côté LN de Π est en réalité constitué par une liste (notée `ListLN`) de besoins naturels que nous avons recensée dans le monde réel. Chaque occurrence de LN correspond à un besoin naturel et elle possède un niveau et une catégorie prédéfinis dont les valeurs sont détaillées plus bas. Bien entendu, cette liste n'est pas exhaustive, et même le contenu que nous avons établi jusqu'ici peut être sujet à discussion. L'essentiel est que pour tout ajout ou modification d'éléments dans `ListLN`, il faut savoir à quel niveau et à quelle catégorie de Π il faut les faire correspondre. La sémantique des besoins naturels donnée dans `ListLN` a pour but unique d'aider les utilisateurs dans leur initialisation du modèle. L'agent en lui-même n'en tiendra pas compte lors de son fonctionnement. Il ne considérera que les niveaux ou catégories des LN correspondants.

Cette attribution du *niveau* et de la *catégorie* aux LN est difficile surtout à un cadre générique. L'enrichissement de cette liste par des scientifiques tels des psychologues, des sociologues voire des biologistes est sans doute souhaitable pour nous faire avancer dans notre classification. Actuellement, nous construisons les éléments de `ListLN` à partir des travaux d'Abraham Maslow (étant donné que ce dernier a déjà établi une certaine classification de ces besoins) ainsi que des travaux généraux sur des animats. Les besoins du niveau 3 et du niveau 4 ont été en outre basés sur les travaux de Solange Elfassy[10] (1999).

L'écriture d'un LN se fait toujours à partir de `ListLN` comme il est indiqué dans l'Equation (III.4).

$$ListLN.need\_libelle = (libellé,\ niveau,\ catégorie) \tag{III.4}$$

[10] Solange Elfassy, une experte internationale du BIT (Bureau International du Travail) a effectué des travaux dans le cadre d'un programme du PNUD à Madagascar. Ce programme dénommé MAG 97/008 porte sur la réduction de la pauvreté au sein de la population malgache. Dans sa méthodologie de travail, elle a utilisé la pyramide de Maslow et a en même temps énoncé des classifications au sein du niveau social de la pyramide. C'est cette classification que nous reprenons ici.

Le paramètre `libellé` permet d'identifier le LN d'une manière unique (sous une forme textuelle) au sein de Π. Il explicite à l'utilisateur la sémantique du besoin, telle que la *faim*, *l'énergie*, la *victoire*, etc. Lors de l'écriture formelle des PN, le libellé passé en paramètre s'écrit en majuscules.

Voici à présent le contenu non exhaustif de `ListLN`.

Niveau 1 : les LN physiologiques

Ce sont les besoins en *eau*, en *nourriture*, en *repos* et en *procréation*[11]. Ces besoins sont tous considérés comme de premier niveau et de première catégorie.

need_water = (WATER, 1, 1)
need_food = (FOOD, 1, 1)
need_rest = (REST, 1, 1)
need_procreation = (PROCREATION, 1, 1)

Niveau 2 : les LN de sécurité

La première catégorie que nous mettons dans ce niveau 2, c'est le besoin d'évitement d'obstacles. La raison en est qu'il est le plus étudié dans le monde des agents et que même dans la vie réelle, l'humain ou l'animal doit éviter l'obstacle avant de pouvoir avancer dans bon nombre d'objectifs au même niveau que lui tels la fuite d'un prédateur. Les autres besoins de ce niveau sont respectivement le besoin de *s'éloigner d'un danger* et le besoin *d'être protégé* (par quelqu'un ou par quelque chose).

need_obstacle_away = (OBSTACLE_AWAY, 2, 1)

need_danger_away = (DANGER_AWAY, 2, 2)
need_protection = (PROTECTION, 2, 2)

[11] Ce dernier est juste énoncé ici car il est déjà mentionné dans le cadre des agents. Mais il n'est pas étudié dans les applications de la thèse car nous ne traitons pas encore les normes de conduite qui différencient l'homme de l'animal.

Niveau 3 : les LN sociaux

Selon Elfassy, le besoin social se classifie en fonction de l'entité avec qui on est en relation sociale. Pour chaque classification, il y a d'un côté le besoin d'aimer, d'accepter et d'aider, et d'un autre côté le besoin d'être aimé, d'être accepté et d'être aidé par l'entité en question. En voici la liste :

1. le besoin de sa famille (catégorie 1),
2. puis le besoin d'amis (catégorie 2),
3. le besoin de collègues (catégorie 3),
4. le besoin d'un groupe social (catégorie 3),
5. le besoin d'une communauté (catégorie 3),
6. le besoin du groupe ethnique auquel on appartient (catégorie 4),
7. le besoin de sa patrie (catégorie 5).

Nous avons ainsi :

need_family_acceptation = (FAMILY_ACCEPTATION, 3, 1)
need_love_family = (LOVE_FAMILY, 3, 1)
need_help_family = (HELP_FAMILY, 3, 1)
need_family_love = (FAMILY_LOVE, 3, 1)
need_family_help = (FAMILY_HELP, 3, 1)

need_friends_acceptation = (FRIENDS_ACCEPTATION, 3, 2)
need_love_friends = (LOVE_FRIENDS, 3, 2)
need_help_friends = (HELP_FRIENDS, 3, 2)
need_friends_love = (FRIENDS_LOVE, 3, 2)
need_friends_help = (FRIENDS_HELP, 3, 2)

need_colleagues_acceptation = (COLLEAGUES_ACCEPTATION, 3, 3)
need_love_colleagues = (LOVE_COLLEAGUES, 3, 3)
need_help_colleagues = (HELP_COLLEAGUES, 3, 3)
need_colleagues_love = (COLLEAGUES_LOVE, 3, 3)
need_colleagues_help = (COLLEAGUES_HELP, 3, 3)

need_society_acceptation = (SOCIETY_ACCEPTATION, 3, 3)
need_love_society = (LOVE_SOCIETY, 3, 3)

need_help_society = (HELP_SOCIETY, 3, 3)

need_society_love = (SOCIETY_LOVE, 3, 3)

need_society_help = (SOCIETY_HELP, 3, 3)

need_community_acceptation = (COMMUNITY_ACCEPTATION, 3, 3)

need_love_community = (LOVE_COMMUNITY, 3, 3)

need_help_community = (HELP_COMMUNITY, 3, 3)

need_community_love = (COMMUNITY_LOVE, 3, 3)

need_community_help = (COMMUNITY_HELP, 3, 3)

need_ethnic_acceptation = (ETHNIC_ACCEPTATION, 3, 4)

need_love_ethnic = (LOVE_ETHNIC, 3, 4)

need_help_ethnic = (HELP_ETHNIC, 3, 4)

need_ethnic_love = (ETHNIC_LOVE, 3, 4)

need_ethnic_help = (ETHNIC_HELP, 3, 4)

need_country_acceptation = (COUNTRY_ACCEPTATION, 3, 5)

need_love_country = (LOVE_COUNTRY, 3, 5)

need_help_country = (HELP_COUNTRY, 3, 5)

need_country_love = (COUNTRY_LOVE, 3, 5)

need_country_help = (COUNTRY_HELP, 3, 5)

Niveau 4 : les LN sur l'estime

Ce besoin se divise en deux catégories :

- le besoin d'être reconnu pour ce que l'on est, pour son droit à la liberté, etc. (catégorie 1)
- et le besoin d'estime et de réputation par l'intermédiaire des autres : statut, pouvoir, etc. (catégorie 2)

En voici la liste :

need_respect = (RESPECT, 4, 1)

need_acknowledgement = (ACKNOWLEDGEMENT, 4, 1)

need_freedom = (FREEDOM, 4, 1)

need_esteem = (ESTEEM, 4, 2)

need_status = (STATUS, 4, 2)

need_power = (POWER, 4, 2)

need_appreciation = (APPRECIATION, 4, 2)

III. 4 Conclusion du chapitre

Ce présent chapitre a introduit une première description du modèle que nous proposons dans cette thèse. Ce modèle, nommé Maslow, se base principalement sur la pyramide des besoins du psychologue américain Abraham Maslow. En rapprochant la notion de besoins et de motivations, cette pyramide, avec les justifications que nous avons apportées dans ce chapitre, s'avère être une intéressante approche. L'idée qui en est véhiculée est que la satisfaction de tout besoin, qu'il soit exprimé par des agents artificiels réactifs ou cognitifs, est toujours motivée à la base par celle d'au moins un besoin naturel. Ces besoins naturels (que nous avons appelés LN) correspondent ainsi à une spécification générique de notre approche en dressant progressivement une liste non exhaustive des besoins naturels trouvés dans le monde réel. Ils pourront alors par la suite être utilisés, au niveau applicatif par des agents de n'importe quel type, via ce que nous avons appelé les MN et les HN. Ces besoins sont introduits car il est clair que la pyramide n'est pas à suivre stricto sensus car il y a toute une dynamique qui n'est pas forcément spécifiée par cette hiérarchie (statique) composant la pyramide.

Le chapitre IV complète ce chapitre en présentant, cette fois-ci, une formalisation plus détaillée de tous les éléments intervenant dans Maslow.

Chapitre IV

Formalisation détaillée des concepts

Ce chapitre a pour objectif d'écrire d'une manière formelle les concepts que nous utilisons par la suite dans le chapitre V lors de la construction du système générique de Maslow. Trois importants concepts sont mis ici en évidence : les besoins, les actions et la motivation.

Dans un premier temps, nous nous consacrerons à l'écriture des PN. Cependant, étant donné que les LN sont déjà présentés par l'Equation (III.4), page 81, nous passerons directement à l'écriture des AN. En fait, à part les notions de *libellé*, de *niveau* et de *catégorie*, héritées de la description d'un LN, d'autres paramètres y interviennent également, à savoir *l'état*[1], le *type* et le *rang*. La section IV.1 décrit l'état[2] alors que la section IV. 2 présente les notions de *type* et de *rang*, suivie par la suite de la formalisation complète d'un AN.

Dans un second temps (Section IV.3), nous formaliserons l'autre concept important lié aux besoins et aux motivations : les *actions*.

---

[1] L'état comme nous le verrons contiendra à son tour d'autres sous-paramètres.

[2] La description des états au niveau des AN ne signifie pas que les LN n'ont pas d'états. Le fait est que dans l'état courant de nos travaux, nous n'avons pas encore tout à fait en main la manière de déclarer un état à un niveau LN du modèle.

IV. 1 Les états des besoins pyramidaux

IV. 1. 1 Notions de base

Un besoin AN d'un agent peut prendre l'un des cinq états suivants :
`insuffisant`, `limit_b` (“ b ” pour “ basse ”), `suffisant`, `limit_h` (“ h ” pour “ haute ”), `excessif`.
Ces états sont disposés le long d'un axe, en suivant l'ordre dans lequel ils sont énoncés ci-dessus. Deux états qui se suivent sur cet axe sont dits *consécutifs*. L'axe en question, le long duquel glisse un *curseur*, est limité aux extrémités par deux points `minPoint` et `MaxPoint`. C'est la position courante (notée `ccpos` pour “ *current cursor position* ”) de ce curseur qui désigne la *valeur courante* de AN dans l'axe. Par la suite, c'est l'état associé à cette valeur qui constitue *l'état courant* du besoin (nous verrons plus bas comment un état est associé à une valeur de l'axe). Ainsi, la condition à respecter pour que ce système du curseur sur un axe fonctionne est que la valeur du besoin doit être toujours de type *numérique*.

Lorsqu'un AN est à l'état `suffisant` (resp. `insuffisant`), nous le notons `AN.isSufficient()` (resp. `AN.isUnsufficient()`). De même pour les autres états, nous avons les notations `AN.isExcessive()`, `AN.isLimit_l()` ou `AN.isLimit_h()`. L'état courant quant à lui est identifié par une variable notée `AN.currentState`.

Parler d'un besoin revient automatiquement à la manière de formuler sa satisfaction (ou son insatisfaction). Ce concept sera détaillé à la section IV. 1. 4. Mais pour en comprendre le principe, deux notions doivent d'abord être introduites : *le scénario des états* et *la forme des états*.

IV. 1. 2 Scénarios de mise en place des états

Dans la disposition des états sur l'axe, il n'est pas nécessaire que les cinq états soient tous présents. L'ensemble d'états réellement présents dans l'axe dépend en fait de la

sémantique du AN modélisé au niveau applicatif. Les *scénarios* de dispositions possibles sont (Equation (IV.1)) :

- `INS_EXC` (pour `insuffisant` à `excessif`) dans lequel les cinq états sont tous présents dans l'axe,
- `INS_SUF` où les états qui sont présents sont consécutivement `insuffisant`, `limite_b` et `suffisant`,
- `SUF_EXC` où les états qui sont présents sont consécutivement `suffisant`, `limite_h` et `excessif`.

$$
\begin{aligned}
&INS\_EXC =< insuffisant, limit\_l, suffisant, limit\_h, excessive > \\
&INS\_SUF =< insuffisant, limit\_l, suffisant > ou < insuffisant, suffisant > \\
&SUF\_EXC =< suffisant, limit\_h, excessive > ou < suffisant, excessive >
\end{aligned}
\tag{IV.1}
$$

Mais comment exactement décrire un état ?

IV. 1. 3 Description et forme des états

Les états peuvent, en fonction des besoins, être représentés sur l'axe sous deux formes : la forme *intervalle* et la forme *point*.

Description des états sous la forme “ intervalle ”

Comme son nom l'indique, chaque état d'un AN s'écrit ici sous forme *d'intervalle* de valeurs, noté `I(etat)` avec `I(etat)=[`a_{etat}`,` b_{etat}`]` (avec bornes d'intervalle non forcément fermées). Lorsque le curseur se trouve dans `I(etat)`, c'est-à-dire que

`ccpos==x /x∈I(etat)`,

cela signifie que le AN associé à l'axe est actuellement à l'état `etat`. Nous allons appeler BFI les besoins dont les états respectent cette forme.

Dans notre travail, nous émettons l'hypothèse que deux intervalles décrivant respectivement deux états consécutifs sont toujours *contigus* et ne se chevauchent jamais. Ainsi, la borne supérieure b_i d'un intervalle `I(etat`$_i$`)` est en réalité toujours égale à la borne inférieure a_{i+1} de l'intervalle `I(etat`$_{i+1}$`)` où `i` indique ici la position des états dans l'axe. Pour chaque ensemble d'états consécutifs, ces deux limites sont donc données dans une

seule et même variable appelée *délimiteur de 2 états*. Nous adoptons alors pour ces variables des noms comme `IL` pour désigner la limite commune entre `insuffisant` et `limite_b`, `LS` pour `limite_b` et `suffisant`, etc.

En outre, remarquons que dans l'hypothèse ci-dessus, si b_i est fermé, alors a_{i+1} est automatiquement ouvert (et vice-versa). Il faut donc, pour chaque délimiteur, connaître son côté fermé et/ou ouvert. Pour cela, nous adoptons la notation `delim<` pour indiquer que c'est b_i qui est *fermée* (et donc a_{i+1} automatiquement ouverte) et la notation `delim>` pour indiquer que c'est a_{i+1} qui est *fermée* (et donc b_i automatiquement ouverte). A propos des points `minPoint` et `MaxPoint`, étant donné qu'ils ne délimitent pas deux états, nous pouvons immédiatement écrire le symbole "] " ou " [" au lieu de " < " ou " > ".

L'écriture d'un état sous forme d'intervalle se réalise comme dans l'exemple formulé par l'Equation (IV.2).

$$desc\_form = \{]delim_1,\ delim_{12} <,\ delim_{23} >,\ ...,\ delim_{ij}]\}\ //\ BFI \tag{IV.2}$$

Description des états sous la forme " point "

Dans cette description, chaque état correspond, non plus à un intervalle, mais à un point. Il y a donc autant d'états que de *valeurs* possibles que peut prendre le curseur. La description est montrée par l'Equation (IV.3). Nous allons appeler BFP les besoins dont les états sont décrits sous cette forme.

$$desc\_form = \{valeur_{etat1},\ ...,\ valeur_{etat2}\}\ //\ BFP \tag{IV.3}$$

Une spécialisation de cette seconde forme est celle de la description dite *booléenne* dans laquelle les AN associés (appelés donc BFB) ne peuvent être que dans deux états : `insuffisant` (équivalant à `faux`) et `suffisant` (équivalant à `vrai`). Dans ce cas particulier, l'état `limite` n'existe pas. La description d'un BFB se fait sous forme d'une proposition booléenne (Equation (IV.4)). Si la proposition est vraie, alors le besoin est à l'état `suffisant` (=1). Autrement, il est à l'état `insuffisant` (=0).

Corollaire : un BFB est toujours associé à un scénario `INS_SUF`.

$$desc\_form = \{proposition\_booléenne\} \;//\; BFB \tag{IV.4}$$

Les BFB servent à représenter une situation de type " tout ou rien ". Par exemple, si une personne a besoin d'un stylo, alors ce besoin n'est à l'état `suffisant` que lorsque cette personne tient effectivement le stylo *dans* sa main.

Schéma synthétique

La figure IV.1 résume les notions précédemment évoquées.

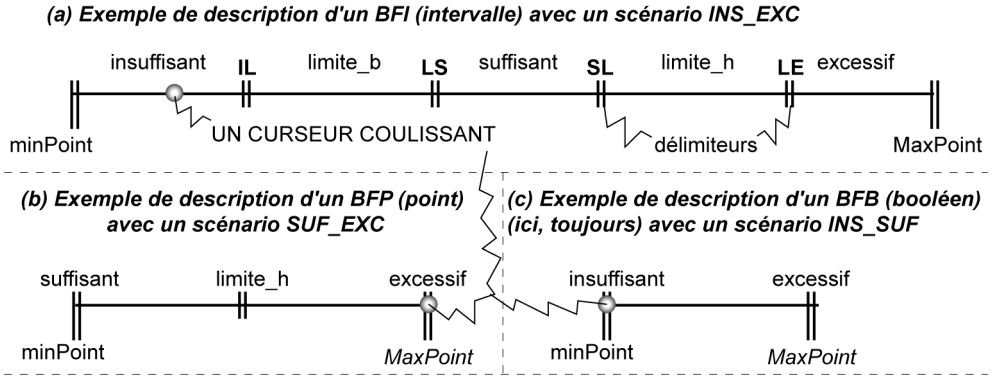


Figure IV.1 : Exemple de description d'états et de scénarios dans un AN

IV. 1. 4 Satisfaction et insatisfaction des besoins

Le fait qu'un besoin soit satisfait ou non dépend de la position du curseur sur l'axe et plus particulièrement sa position par rapport à la zone de satisfaction que nous définissons ci-après.

La *zone de satisfaction* d'un AN, c'est l'ensemble des valeurs sur l'axe, pour lequel le AN est *satisfait*. Déterminer cette zone revient à connaître deux points maximum (noté `maxSat`) et minimum (noté `minSat`) entre lesquels le besoin est dit *satisfait* (noté `AN.isSatisfied()`). En dehors de ces points, le besoin est dit *insatisfait* (noté `AN.isUnSatisfied()`).

L'algorithme de détermination de `maxSat` est le suivant :

- dans un scénario `INS_SUF` (resp. `SUF_EXC`), `maxSat=MaxPoint` (resp. `maxSat=minPoint`) et ce, quelle que soit la forme du AN (BFI, BFP ou BFB).

- dans le scénario `INS_EXC`, `maxSat` correspond au milieu de l'intervalle correspondant à l'état `suffisant`, intervalle appelé aussi la *zone idéale*.

Lorsque `ccpos==maxSat`, le besoin est dit *pleinement satisfait* (noté `AN.isFullySatisfied()`)

Corollaire : lorsqu'un AN est un BFP ou un BFB :
`AN.isSufficient()` $\Rightarrow$ `AN.isFullySatisfied()`.

La détermination de `minSat` est un peu plus compliquée que celle de `maxSat`. En effet, comment le situer ? Si l'on prend l'exemple du scénario `INS_SUF` avec un BFI, faut-il situer `minSat` sur LS ou sur IL ? Dans le premier cas, cela revient à dire que l'état `limite_b` correspondra automatiquement à une insatisfaction du besoin, ce qui n'est pas du tout certain. Inversement, si on place `minSat` sur `IL`, alors cela signifie que l'état `limite_b` correspond à une satisfaction, ce qui n'est pas certain non plus. Un état *limite* sert généralement comme une sorte de " pont " entre la suffisance et l'insuffisance.

En conclusion à ce problème sur la détermination de `minSat`, nous adoptons la solution suivante :

- pour un BFI, nous mettons le point `minSat` quelque part dans l'état *limite*. Ce " quelque part ", c'est l'utilisateur qui va le déterminer au niveau applicatif. Il y aura alors autant de `minSat` que d'états limites. Pour le scénario `INS_SUF`, `minSat` peut être pris entre `IL` et `LS` et pour le scénario `SUF_EXC`, entre `SL` et `LE`. Pour le scénario `INS_EXC`, il y aura deux `minSat` : un `minSat_b` dans `limit_b` et un `minSat_h` dans `limit_h`.
- pour un BFP, il convient à l'utilisateur de choisir si `minSat` doit être égalisé avec l'état `limite` (" b " ou " h ") ou avec l'état `suffisant`.
- pour un BFB, et comme il n'y a que deux états possibles, il n'y a donc qu'un seul point de satisfaction, qui est `minSat=maxSat=MaxPoint`.

La figure IV.2, reprenant les exemples de besoins de la figure IV.1, page 91, illustre la place des deux points ci-dessus pour chaque forme de besoin modélisé.

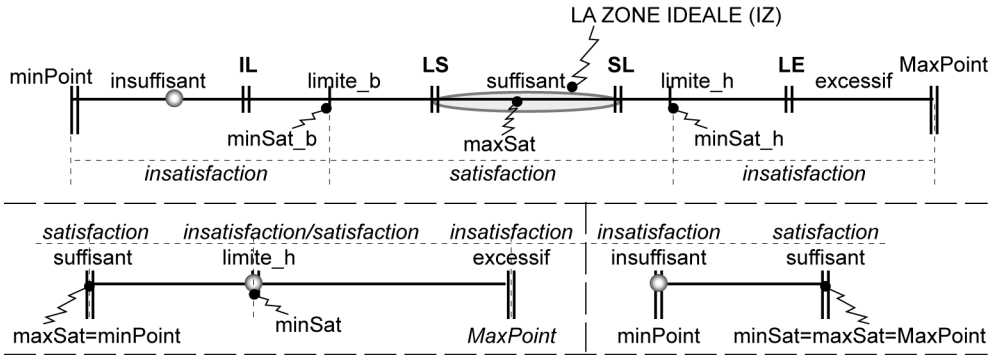


Figure IV.2 : Délimitation de la zone de satisfaction/insatisfaction pour chaque forme d'états

IV. 1. 5 Formalisation complète d'un état

L'écriture de l'état d'un AN de libellé `libelle` se résume à l'énoncé :

- de `lib_form`, le libellé de la *forme* de l'état (les valeurs possibles étant `interval`, `point` ou `boolean`),
- du *scénario* adopté (`INS_SUF`, `INS_EXC`, `SUF_EXC`),
- de la *description* numérique ou booléenne de la forme en question
- ainsi que du ou des points `minSat` associé(s) (Equation (IV.5)-a). Pour le cas de la forme booléenne (Equation (IV.5)-b), il n'est point besoin d'énoncer, ni le scénario (toujours `INS_SUF`) ni le `minSat` (toujours {1}).

$$\begin{aligned} &state\_libelle = (lib\_form/scenario,\ \{desc\_form\},\ \{minSat,\ [minSat]\})\ (a)// \ BFI,\ BFP \\ &state\_libelle = (boolean,\ \{proposition\})\ (b)\ //\ BFB \end{aligned} \tag{IV.5}$$

Voici un exemple d'écriture de l'état d'un besoin MN de libellé `FAIM`, que nous supposons un BFI :

```
state_faim=(interval/SUF_EXC, {[0, 2.5<, 7.5>, 10]}, {6.2})
```

Ici, la valeur des bornes quantitatives (c'est-à-dire `0`, `2.5`, etc.) est donnée à titre d'exemple mais il appartient à l'utilisateur de fournir les données exactes.

IV. 2 Formalisation des MN et des HN

Avant d'aborder l'écriture générale des AN, passons brièvement en revue les paramètres intervenant dans la formalisation, autre que les paramètres dérivant de LN (c'est-à-dire libellé, niveau, catégorie) et les états. Il s'agit en occurrence du *type* et du *rang* des besoins.

IV. 2. 1 Le type et le rang

Le type

Ce sont les types introduits initialement à la page 75, à savoir les LN, les MN et les HN. Le paramètre *type* peut ainsi prendre les valeurs `type_LN`, `type_MN` ou `type_HN`. Inversement, une fonction notée `type(PN)` nous permettra de retourner le type d'un PN donné.

Le rang

Tout comme le *niveau* et la *catégorie*, le *rang* désigne un emplacement physique d'un AN dans Π. Rappelons tout d'abord que le paramètre `niveau` (place verticale) est identique à la spécification donnée par Abraham Maslow. Ensuite, le paramètre `catégorie` (place horizontale) constitue un positionnement des PN situés à un même niveau. Le `rang` (également une place horizontale) correspond à un positionnement des AN appartenant à une même catégorie. La différence entre `catégorie` et `rang` est que pour un AN, la valeur de `catégorie` est automatiquement connue car c'est celle du LN sur lequel il est basé (rappel Equation (III.2), page 80), tandis que celle de `rang` est attribuée par l'utilisateur. Il arrive en effet que la place de divers besoins rattachés à un même LN dépende, soit du contexte applicatif, soit de la manière dont l'utilisateur interprète son besoin. A ce moment-là, c'est `rang` qu'il faut manipuler.

IV. 2. 2 Ecriture générale

Soit `ListLN.ln` la motivation associée au AN à modéliser, la formulation d'un AN est mentionnée dans l'Equation (IV.6). Dans cette équation, `libelle` est le libellé du AN,

`state_libelle` la description de la forme de son état, `type` son type et `rang` son rang dans Π. Ce dernier paramètre peut être omis s'il est égal à 1.

$$\begin{array}{l} AN = (libelle,\ ListLN.ln,\ type,\ rang,\ state\_libelle) \\ AN = (libelle,\ ListLN.ln,\ type,\ state\_libelle)\ //\ si\ rang == 1 \end{array} \quad \text{(IV.6)}$$

Pour pouvoir écrire l'Equation (IV.6), `state_libelle` devra donc au préalable être initialisé, via la formalisation déjà présentée dans l'Equation (IV.5).

En reprenant le besoin MN=`FAIM` de la section IV.1.5, nous avons donc :

```
state_faim=(interval/SUF_EXC, {[0, 2.5<, 7.5>, 10]}, {6.2})
MN={FAIM, ListLN.need_food, type_MN, 1, state_faim}
```

Dans l'exemple ci-dessus, le MN est de niveau 1 et de catégorie 1. Ces valeurs ont été obtenus via `ListLN.need_food`. En revanche, la valeur `rang` =1 dans l'exemple n'est qu'un choix de notre part en tant qu'utilisateur.

A présent que les besoins PN sont formalisés, procédons à la description du concept qui lui est indissociable : les actions.

IV. 3 Les actions

Il existe deux catégories d'actions :

- les *primitives* (notées PR) qui constituent le niveau de granularité le plus fin de la catégorie. Une PR ne peut être interrompue lorsqu'elle est exécutée. Pour un agent, la prise en compte d'un événement quelconque pour un agent ne peut se faire qu'entre l'exécution de deux PR.
- les *actions composées* (notées AC) qui sont formées d'une ou de plusieurs PR et/ou d'autres AC appelées *sous-actions* de AC. Une AC n'est pas directement exécutable. Seules les PR qui la composent le sont.

L'ensemble des actions est noté Δ (Equation (IV.7)).

$$\Delta = \{PR\} \cup \{AC\} \quad \text{(IV.7)}$$

IV. 3. 1 Les motivations dans les actions

Les actions et les besoins (c'est-à-dire les PN) étant définis, voyons à présent comment la notion de *motivation* est formulée : à chaque état d'un AN correspond en fait une action notée `AN.action[state]` (Figure IV.3). Ces actions peuvent être différentes par état ou identiques pour un groupe d'états, voire pour tous les états. Lorsque AN est à l'état `etat`, alors `AN.action[state]` est appelée *action courante* de AN. Si une seule action est utilisée pour tous les états du AN, cette action est appelée *action par défaut* (notée `AN.defaultAction` pour le AN en question. En donnant une valeur à `AN.defaultAction`, l'utilisateur initialise automatiquement tous les `AN.action[state]`.

Comme une `AN.action[state]` peut aussi être une AC, la satisfaction d'un AN peut passer par l'exécution d'autres actions qui sont elles aussi liées à d'autres besoins.

Enoncé : La relation de base entre *motivations* et *actions* s'établit de la manière suivante : lorsque l'état d'un AN de l'agent se trouve dans l'état `etat`, alors l'agent est motivé à exécuter l'action `AN.action[state]` correspondante, avec comme objectif le rapprochement du curseur de AN le plus près possible de `maxSat`.
Il reste par la suite pour l'agent à déterminer lesquelles parmi toutes les motivations engendrées par tous ces AN de Π il va prendre en compte en priorité, c'est-à-dire celles dont il va effectivement lancer l'action correspondante. Les détails de ce problème se trouvent dans la section V.3, lorsque nous parlerons du degré de motivation et du processus de sélection des besoins et des actions.

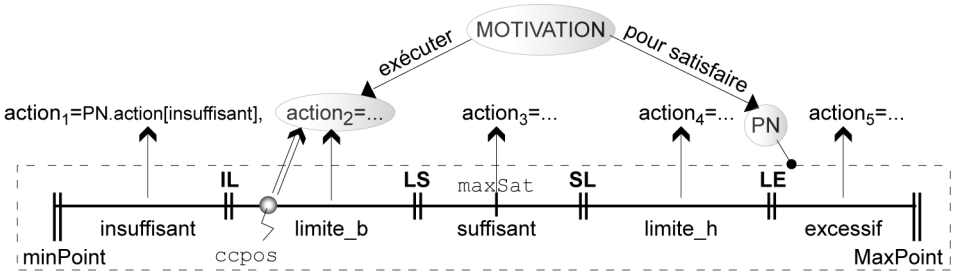


Figure IV.3 : Relation de base entre motivation et action

IV. 3. 2 L'interconnexion des actions

Les actions peuvent être reliées entre elles par ce que l'on appelle des *connecteurs* (Figure IV.4). Actuellement, notre modèle en contient 4 : le `then`, le `xor`, le `and` et le `imp`. Voici les notations essentielles de cette interconnexion.

Soient $a_1, a_2 \in \Delta$:

- pour connecter a_1 et a_2, on écrit `connector(`a_1`, `a_2`)`
- pour tester si a_1 et a_2 sont effectivement connectées via `connector`, on écrit `isConnector(`a_1`, `a_2`)`
- un connecteur est *réciproque* si `connector(`a_1`, `a_2`)` $\Rightarrow$ `isConnector(`a_2`, `a_1`)`

Chaque connecteur a un fonctionnement spécifique :

- le `then(`a_1`, `a_2`)` permet à a_2 de s'exécuter après a_1. Ce connecteur n'est pas réciproque.
- le `imp(`a_1`, `a_2`)`, non réciproque cette fois, signifie que l'exécution de a_1 entraîne automatiquement l'exécution de a_2 (c'est-à-dire que a_1 agit uniquement comme une sorte de pointeur vers a_2). La situation qui se passe en général est que lorsque `isImp(`a_1`, `a_2`)`, c'est parce que $\exists a_3 \in \Delta /$ `isImp(`a_1`, `a_3`)` c'est-à-dire qu'une action n'est jamais en `imp` avec une seule action car autrement, le `imp` n'aurait aucun intérêt. D'une manière générale, lorsque $\exists\ a_x \in \Delta /$`isImp(`$a_1$`, `$a_x$`)`, alors a_1 est toujours une AC et a_x fait partie des sous-actions de a_1.
  Une propriété particulière de `imp` est la *transitivité* à savoir que :
  si (`isImp(`a_1, a_2`)` $\wedge$ `isImp(`a_2, a_3`)` alors `isImp(`a_1`, `a_3`)`.
  Il existe aussi un prédicat `isImpDirect(`a_1`, `a_2`)` qui retourne *vrai* lorsque `isImp(`a_1`, `a_2`)` n'est pas le résultat d'une transitivité.
- le `xor(`a_1`, `a_2`)` empêche a_1 et a_2 de s'exécuter en même temps, en raison de *conflits d'actions* c'est-à-dire lorsque ces deux actions ne peuvent être exécutées simultanément par l'agent. Ce connecteur est réciproque. Nous verrons néanmoins plus tard qu'il existe des actions qui s'excluent automatiquement lors de la sélection d'actions alors qu'il en existe qui doivent être explicitement mises en conflit par l'utilisateur.

Lorsqu'une action `act` est mise en conflit avec une AC, alors `act` est automatiquement en conflit avec toutes les sous-actions de cette AC. Telle est la *loi du conflit* (Equation (IV.8)).

$$si\ \exists a\ \in\ \Delta\ /\ isImp(a,\ a_1)\ \wedge\ isXor(a,\ a_2)\ alors\ isXor(a_1,\ a_2) \qquad \text{(IV.8)}$$

– le `and(`a_1`,` a_2`)` permet aux deux actions de s'exécuter en même temps. En fait, c'est le " connecteur par défaut " qui se caractérise justement par une absence de connexion entre deux actions, ce qui leur permet cette exécution simultanée. Ce connecteur est réciproque.

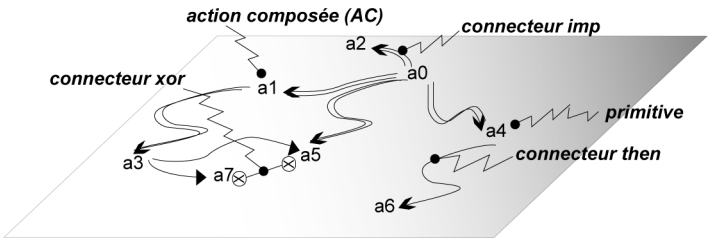


Figure IV.4 : Exemple schématique de connexion des actions

Avant de passer au détail de la formalisation d'une action (Section IV.3.4), introduisons brièvement la notion de temps. Nous en parlons ici car le temps intervient dans ce travail, notamment, lors de l'exécution de certaines actions issues des motivations. Aussi, avons-nous intégré un modèle minimal permettant de gérer le temps dans ce contexte. A noter cependant que la notion de temps n'est pas encore réellement approfondie dans MASLOW, notre objectif n'étant pas de proposer un modèle de planification ou de gestion de temps quelconque.

Remarque : Au niveau applicatif, le temps n'est pas forcément pris en compte dans l'action des agents. La raison en est que soit l'action n'est pas planifiée, soit l'agent n'intègre pas la notion de temps, ce qui est par exemple l'hypothèse que nous avons adoptée pour les agents réactifs de ce travail.

IV. 3. 3 Le temps dans les actions

Pour notre modèle, un temps est une structure de type `Time` définie comme suit :

`Time={libelle, année, mois, jour, AT, heure, minute, seconde}` où `AT` est uniquement un séparateur.

Une occurrence du temps est liée à l'exécution d'une action (cf. Section IV.3.4 pour les détails). Lorsque l'un ou plusieurs des paramètres ci-dessus vaut -1, cela signifie que le ou les paramètres en question ne sont pas pris en compte au moment de l'exécution de l'action associée. Plus précisément, soit `t∈Time` et `act` l'action associée :

- si `t.année` vaut -1, `act` peut être exécutée à n'importe quelle année de l'activité de l'agent.
- si `t.mois` vaut -1, `act` peut être exécutée à n'importe quel mois. Dans ce cas, si `t.année` ne vaut pas -1, alors `act` peut être exécutée à n'importe quel mois de cette année. Autrement, `act` est exécutable à n'importe quel mois de n'importe quelle année.
- ce processus se poursuit avec les autres paramètres de la structure `Time` (sauf `libelle` qui ne fait qu'expliquer textuellement à quoi correspond ce `Time`). Nous pouvons alors aboutir à un bon nombre de combinaisons d'occurrence de temps.

La valeur par défaut d'un temps non initialisé est

`t∈Time={-1, -1, -1, AT, -1, -1, -1}`.

Il existe ainsi plusieurs manières de représenter la structure `Time` notamment pour cacher implicitement des paramètres qui valent -1. Ainsi, nous pouvons avoir :

`Time={libelle, AT, heure, minute, seconde }` // pas d'année, ni de mois ni de jour

`Time={libelle, année, mois}` // pas de jour, ni d'heure, ni de minute, ni de seconde. Lorsque les 3 derniers paramètres ne sont pas mentionnés, le `AT` peut être omis. etc.

Cette notion de temps étant introduite, passons maintenant à l'action proprement dite.

IV. 3. 4 Formalisation des actions

Les paramètres

Toute `action`$\in \Delta$ s'écrit formellement comme suit :

`action={action_name, precondition, pn_satisf, start_time, end_time, liste_conflits}` avec :

- `action_name` est le nom de l'action.
- `precondition` est un AN qui doit être satisfait pour que l'action puisse être exécutée. Bien que ce soit un AN, sa distinction avec le paramètre `pn_satisf` ci-dessous est qu'elle est prise en compte non pas lors de l'évaluation de l'importance des besoins mais plutôt lors du processus de sélection des actions[3].
  Lorsque `precondition` n'existe pas, cela signifie que l'action peut toujours être exécutée. On note `precond(a)` la précondition d'une action `a` et `preCondOK(a)` le prédicat qui retourne `vrai` si `precondition` est vérifiée ou n'existe pas.
- `pn_satisf` équivaut au PN que l'agent est actuellement en train de satisfaire via cette action. Autrement dit,
  si `a==PN.action[state]`$\Rightarrow$`pn_satisf(a)=PN`.
  Deux actions peuvent donc posséder le même `pn_satisf`. Lorsque tel est le cas, cela signifie que chacune d'elles est automatiquement reliée à des états différents de ce même PN.
- `start_time` $\in$`Time`, fait référence au temps le plus tôt du début de l'action. Si ce paramètre n'est pas défini, l'action peut commencer à n'importe quel moment de l'activité de l'agent.
- `end_time` $\in$ `Time`, correspond au temps le plus tard pour terminer l'action. Si ce paramètre n'a pas de valeur, l'action peut se terminer à n'importe quel moment.
- `liste_conflits` contient la liste des actions que l'agent ne peut pas exécuter en même temps que cette action.

[3] Ce mécanisme d'évaluation de priorité sera développé dans la section V.3.1 lorsque nous construirons le modèle générique.

Tous ces paramètres constituent le *contexte* dans lequel l'action se trouve.

Voici maintenant quelques prédicats relatifs au temps :

- `programmed(action)` : retourne `vrai` si les deux paramètres liés au temps sont en même temps initialisés
- `isTimeOK` : retourne `vrai` si :
  - `programmed(action)` est `faux`
  - ou `programmed(action)` est `vrai` et que le temps d'exécution de l'action est arrivée.

Différence formelle entre PR et AC

Nous avons dit qu'une PR est directement exécutable. En réalité, lorsque l'action est une PR, le paramètre `action_name` pointe vers une *fonction* qui, elle, est réellement exécutable. Il y a donc ici distinction entre cette *fonction* et la notion de *contexte* énoncée ci-dessus. Une fonction peut être associée à n'importe quelle autre PR, ce qui n'est pas le cas des contextes. Lorsqu'une PR est une sous-action d'une AC, alors elle est une PR uniquement de cette AC (une PR " n'appartient pas " à deux AC). En revanche, la *fonction* de cette même PR peut être dans d'autres PR qui " appartiennent " à d'autres AC.

Au niveau de la formalisation par rapport à l'action, il n'y a rien à ajouter puisque la fonction est déjà implicitement représentée par `action_name`.

Pour les AC, le contexte existe également (puisque ce sont les mêmes paramètres provenant des actions), mais il n'y a pas de *fonction* puisqu'une AC n'est pas directement exécutable. Par contre, il existe un paramètre nommé `liste_sous_actions` qui pointe vers l'ensemble des sous-actions d'une AC. Au niveau de la formalisation, nous avons ainsi :

$$\texttt{AC} = \texttt{action} \cup \texttt{liste\_sous\_actions}.$$

Si des paramètres n'ont pas de valeur, ils sont remplacés par le mot clé `n/a` (" not available "), voire supprimés s'ils se trouvent en fin de liste. Par exemple, la formulation

`action={action_name, precondition, pn_satisf}`

signifie que les paramètres concernant le temps et ceux des conflits ne sont pas initialisés.

Et si on est à un niveau AC, la formulation ci-dessus implique également que le paramètre `liste_sous_actions` n'est pas non plus à ce moment précis initialisé[4].

IV. 4 Conclusion du chapitre

Ce chapitre revêt un aspect uniquement technique. Il a eu pour ambition de formaliser point par point les éléments de base du modèle MASLOW que nous avons initialement introduit et proposé dans le chapitre III. Il s'agit principalement des besoins, des motivations et des actions. Ces éléments formels seront ensuite repris dans le chapitre V pour pouvoir modéliser l'architecture et le comportement de notre agent générique hybride.

[4] Ce qui est, comme nous le verrons, généralement le cas, ce paramètre étant initialisé au moment de la mise en place des connecteurs `imp`, c'est-à-dire en fonction de l'application.

Chapitre V

Vers un modèle générique hybride à base de motivations

Maintenant que les concepts sont formalisés (chapitre IV), il s'agit à présent de décrire leur fonctionnement, soit individuellement, soit en interaction les uns avec les autres. Ce chapitre détaille ce fonctionnement en divisant notre explication en deux points principaux : *l'architecture* (Section V.2) et *le comportement* (Section V.3) de l'agent que nous mettons en œuvre.

V. 1 Préambule

Jusqu'ici, lorsque nous parlions du modèle MASLOW, nous faisions uniquement référence à l'existence d'une pyramide Π sur laquelle sont répartis des besoins PN de divers types : LN, HN, MN. La raison en est que c'est Π qui constitue le point central de départ de la motivation de l'agent. Cependant, il existe par la suite d'autres composants qui s'ajoutent à Π et qui interagissent avec elle, le tout formant l'architecture réelle de l'agent. En fait, le système MASLOW est le résultat de l'interaction de trois composants principaux (Equation (V.1)) :

- la *pyramide* Π,
- un *réseau d'actions* noté Ω dont le fonctionnement dépend des éléments qui se trouvent dans Π,
- un moteur générique appelé NIM pour " Need Importance Manager ".

Nous expliquerons les deux premiers composants dans une même section (Section V.2) et le NIM dans une autre section (Section V. 3). La raison est d'ordre fonctionnelle : Π et Ω sont accessibles à l'utilisateur et à l'agent tandis que le NIM ne se situe qu'au niveau de l'agent uniquement. C'est lui qui pilote la proactivité de l'agent à un niveau générique, notamment dans le cadre de la sélection des motivations et des actions.

$$MASLOW =< \Pi, \Omega, NIM > \tag{V.1}$$

Le point commun de ces composants, c'est qu'ils partent tous à la base, des besoins de l'agent.

V. 2 Architecture : la pyramide et le réseau des actions

V. 2. 1 La pyramide Π

Elle a déjà été largement évoquée dans ce document. Elle est la représentation de la pyramide de Maslow et est composée d'un ensemble de besoins dits PN. Le résumé de la composition de Π est rappelé dans l'Equation (V.2).

$$\Pi = \{PN\} = \{LN\} \cup \{AN\} = \{LN\} \cup \{MN\} \cup \{HN\} \tag{V.2}$$

Nous y reviendrons sans doute lorsque nous parlerons du NIM. Pour le moment, passons à un concept qui nous semble être nouveau : le réseau d'actions Ω.

V. 2. 2 Le réseau Ω

Présentation

C'est un réseau dont les nœuds sont constitués par des actions (PR ou AC) et dont les arcs sont formés par un ensemble des connecteurs `xor`, `and`, `then` et `imp`, que nous avons introduits dans la section IV.3.2.

Le réseau, formé du couple d'un ensemble de nœuds et d'arcs est montré dans l'Equation (V.3).

$$\begin{aligned}\Omega &= (\Delta, \{connectors\}) \\ &= (\{AC\} \cup \{PR\}, \{xor\} \cup \{and\} \cup \{then\} \cup \{imp\})\end{aligned} \tag{V.3}$$

La relation de base entre Π et Ω est la suivante (Equation (V.4)) : soit `n` le nombre de AN existant dans Π, alors les actions dans Ω sont composées par toutes les `AN.action[state`$_i$`]` issues de chaque élément AN de Π. La valeur de `i` dépend du scénario de disposition choisi pour modéliser les états de chaque AN (rappel Equation (IV.1), page 89). Mais sa valeur maximale est 5, le nombre maximum d'états possibles.

$$\Delta = \cup \overset{n}{\underset{k=1}{\{}} AN_k.action[state_1], \ldots, AN_k.action[state_e]\}, 1 \leq e \leq 5 \tag{V.4}$$

Inversement, nous pouvons dire que si A=$\{$`a`$_1, \ldots,$ `a`$_n\}$ l'ensemble des actions du système, alors {`AN`}={`pn_satisf(a)`}. Autrement dit, la partie applicative de Π est formée par l'ensemble du paramètre `pn_satisf` de chaque action dans Ω.

Caractéristiques de Ω

Le réseau Ω s'inspire largement de l'architecture ANA de Maes (1991), dans laquelle les nœuds sont des modules correspondant à des actions élémentaires (voir aussi Section II.2.3). Rappelons que chaque action dans ANA est décrite par ses conditions d'application, son niveau d'activation, les liens qu'elle établit avec ses prédécesseurs, ses successeurs et les actions avec lesquelles elle se trouve en opposition. Notre modèle en suit le principe de base.

Nous reconnaissons que l'une des idées premières de ANA est celle d'éviter toute notion de hiérarchie dans le réseau, d'où justement le fait que les actions qui le composent sont toutes des actions élémentaires. Notre modèle va néanmoins introduire un *minimum* de notion de hiérarchie (d'où l'utilisation des AC) qui se justifie par notre souci d'optimiser

la création du réseau. Optimiser signifie *factoriser* certaines actions qui disposent de certains paramètres identiques :

- tout d'abord, il y a le paramètre `precondition` par rapport auquel la *factorisation* est introduite pour éviter toute *redondance* de formalisation de ces mêmes paramètres par un groupe d'actions[1].
- l'autre paramètre est le *conflit d'actions*. Tout d'abord, rappelons que lorsqu'une action `act` est mise en conflit avec une AC, alors `act` est automatiquement en conflit avec toutes les sous-actions de cette AC (rappel, la *loi du conflit* (Equation (IV.8), page 98). Il suffit alors de mettre un seul lien, celui partant de `act` vers AC, pour que le conflit s'établisse, plutôt que de mettre autant de conflits qu'il y a de sous-actions de AC.

Notre approche entraîne moins de *surcharge* du réseau, une facilité de compréhension du scénario de l'application ainsi qu'une meilleure organisation des actions. Malgré tout, de même que nous évitons une décomposition trop horizontale des données (réseau trop plat et surchargé), nous minimisons également une décomposition verticale trop poussée. L'obligation de hiérarchiser doit être motivée uniquement par la factorisation.

Notion du niveau de décomposition, des PN finaux et des PN intermédiaires

Cette hiérarchisation du réseau nous emmène à la notion de *niveau de décomposition* (noté `nd`). Ce paramètre indique le niveau sur lequel chaque action de Ω (et par conséquent chaque besoin associé à ces actions) se trouve. Le *point d'entrée* de Ω (noté `net_entry`) correspond à une action composée générique ayant le `nd` égal à 0.

Les besoins situés à `nd` =1 du réseau sont appelés *les besoins finaux*. Tout besoin qui n'est pas *final* est dit *intermédiaire*. Dans sa proactivité, l'objectif général de l'agent est de satisfaire les PN finaux.

V. 2. 3 Initialisation de Π et de Ω

Ces deux composants étant maintenant définis, voyons comment ils sont initialisés par l'utilisateur. Rappelons que ce sont les données qui y sont introduites (et qui dépendent de l'application) que le NIM va ensuite gérer à un niveau générique.

[1] En définitif, les AC servent essentiellement à la gestion de la factorisation

L'initialisation des deux composants se déroule par étapes. Cependant, il n'y a pas de limite stricte entre ces étapes. En d'autres termes, la progression ou la finition d'une étape pourrait faire revenir à une étape précédente pour la compléter. Nous en présentons ici les grandes lignes de la démarche.

La réalisation de la connexion des actions entre elles n'est pas toujours facile, notamment dans des applications contenant des données en volume importantes. En effet, il n'est pas toujours évident de trouver en une seule fois toutes les connexions possibles entre les actions. Dans ce cas, les liens ne se mettront progressivement en place qu'au fur et à mesure de l'avancée de la simulation. Cela se fait par exemple pour le cas du connector `xor`, par la détection de comportements irréalistes (ex : être à la fois au four et au moulin), obligeant l'exclusion de l'une ou l'autre des actions.

Recensement et initialisation des AN à satisfaire

Cette étape signifie :

1. la connaissance de tous les besoins AN à satisfaire dans l'application,
2. puis leur initialisation, en déterminant leur descripteur d'états, leur type (MN ou HN), leur rang et surtout les LN dont, rappelons-le, ils ne sont que la manifestation au niveau applicatif (rappel Equation (IV.6), page 95).

Le choix des AN dépend évidemment des utilisateurs et de la manière dont il interprète les besoins (surtout qu'un même besoin peut être interprété différemment par deux utilisateurs). L'essentiel est que cet utilisateur sache à quel type ces AN appartiennent et à quel(s) LN ils sont associés. Le choix de la description des états d'un PN dépend aussi de l'utilisateur.

Recensement des actions

Tout comme les AN, l'utilisateur va aussi recenser les PR et les AC en fonction de ses objectifs applicatifs. Comme nous le spécifions à la section V.2.2, il convient pour le modèle de limiter la hiérarchisation dans le cadre unique de la factorisation et de ce fait réduire au minimum le nombre de AC sur le réseau.

Recensement et initialisation de la précondition

Vient par la suite le traitement du paramètre `precondition` des actions (pour celles qui en disposent). Rappelons que c'est aussi un AN. A ce titre, il est initialisé comme un AN sauf qu'il n'est pas pris en compte lors de l'évaluation des besoins par le NIM, mais lors de la sélection d'actions. Nous pouvons noter par ailleurs que l'initialisation de `precondition` passe *après* celle des actions alors que celle du `pn_satisf` passe *avant* celle des actions.

Association des actions aux AN et aux préconditions

Ici, les besoins seront pointés vers les actions correspondantes, et vice-versa. A l'issue cette association, trois paramètres sont désormais connus : `precondition` et `pn_satisf` pour toutes les actions, et `PN.action[state]` ou `PN.defaultAction` pour tous les AN.

Connexion des actions entre elles

Enfin, installer les connecteurs entre les actions. Soient $a_1, a_2 \in \Delta$, alors la démarche à suivre par l'utilisateur pour placer les arcs dans Ω est la suivante :

Installation des connecteurs " then "

Pour créer un connecteur `then` entre a_1 et a_2, il faut spécifier `pn_satisf(`a_1`)` et `precond(`a_2`)` de façon à ce que a_2 ne puisse s'exécuter sans que `pn_satisf(`a_1`)` soit satisfait (Equation (V.5)-a), ou inversement, l'exécution de a_2 permet de déduire automatiquement que `pn_satisf(`a_1`)` est en fait déjà satisfait (Equation (V.5)-b).

$$\begin{array}{l} pn\_satisf(a_1).isUnSatisfied() \Rightarrow precond(a_2).isUnSatisfied()) \;//\; (a) \\ precond(a_2).isSatisfied() \Rightarrow pn\_satisf(a_1).isSatisfied() \;//\; (b) \end{array} \quad \text{(V.5)}$$

Corollaire : Cette affectation d'une précondition par un `pn_satisf` peut donc se produire implicitement lors de la phase précédente qui concernait l'association de toutes les AN à leur `pn_satisf` et préconditions respectives. Il n'y a pas réellement une installation explicite d'un connecteur `then`. Il s'agit plutôt de faire en sorte que l'une des spécifications de l'Equation (V.5) soit vérifiée.

Installation des connecteurs " xor "

La création d'un connecteur `xor` par l'utilisateur se traduit par la mise de a_1 dans `liste_conflits` de a_2, et vice-versa.

Cependant, il existe trois situations où le `xor` se met *implicitement* en place, sans que l'utilisateur n'intervienne[2] :

1. lorsque deux actions sont programmées pour ne pas s'exécuter en même temps,
2. lorsque la loi du conflit, énoncée dans l'Equation (IV.8), page 98, s'applique, c'est-à-dire que le `xor` s'établit entre une action a_1 et les sous-actions de a_2 avec qui elle est en conflit.
3. lorsque pour un PN donné `PN.action[state`$_i$`]` est différent de `PN.action[state`$_j$`]`, ces deux actions sont aussi implicitement en `xor`.

Installation des connecteurs " imp "

Le connecteur `imp` n'est applicable que pour les AC. Si l'utilisateur souhaite que le prédicat `isImp(`a_2`,` a_1`)` retourne vrai, alors il doit mettre a_1 dans `liste_sous_actions` de a_2

Par défaut, l'AC `net_entry` est mis en `imp` automatique avec toutes les AC et toutes les PR possibles qui sont liées aux besoins finaux.

Installation des connecteurs " and "

Si aucun autre connecteur n'est établi entre a_1 et a_2, cela signifie implicitement que `isAnd(`a_1`,` a_2`)` est automatiquement vrai.

Initialisation du temps

Cette étape ne s'effectue que lorsque l'utilisateur souhaite que son agent dispose d'une *mémoire* (notion mieux développée dans la section V.4.2) et intègre la notion du temps.

---

[2] Cela signifie que le connecteur n'est pas représenté sur le schéma du réseau Ω pour ne pas en compliquer la lecture. N'empêche que ces actions sont en `xor` car par respect de la définition de base du `xor`, elles ne s'exécutent pas en même temps.

C'est la mémoire qui stocke les paramètres temps. Lorsque tel est le cas, l'initialisation du temps se fait alors comme suit avec `timeValue` ∈`Time` :

- pour les actions qui doivent commencer à `timeValue`, faire :
  `start_time(action, timeValue)`
- pour les actions qui doivent se terminer à `timeValue`, faire :
  `end_time(action, timeValue)`

Toutes ces initialisations effectuées, le NIM de l'agent va prendre en charge la gestion de son comportement.

V. 3 Comportement : le NIM et la sélection des motivations

Le NIM est le pilote du système MASLOW. Son rôle premier est d'évaluer tous les besoins AN situés dans Π afin de dégager le(s) besoin(s) le(s) plus important(s), c'est-à-dire ceux qui doivent être traités par la suite par l'agent. C'est derrière tout ce processus du NIM que nous retrouvons la *proactivité* de l'agent : tout au long de son cycle de vie, l'agent va essayer de satisfaire ses besoins finaux en passant par la satisfaction de ses besoins intermédiaires. Le NIM va ainsi travailler d'une manière cyclique en lançant son algorithme de sélection au travers de Π et de Ω. Cet algorithme est appelé *algorithme de proactivité*.

Le déroulement de l'algorithme de proactivité du NIM comporte deux fonctions imbriquées : la première que nous appelons `testPriorityBetween(PN, PN')` permet de savoir lequel est important entre PN et PN'. La seconde, que nous appelons tout simplement `algoNIM(AC, nd)` effectue la sélection en général, en partant de la sélection des besoins à la sélection d'actions. Cette dernière fonction fait appel à `testPriorityBetween`. Au lancement de l'algorithme, `nd` vaut 1 et `AC` prend la valeur de `net_entry`.

V. 3. 1 Les filtres de sélection

Le degré de motivation est le terme désigné pour déterminer le niveau d'importance d'un besoin PN. En d'autres termes, plus un besoin est important, plus l'agent est motivé à le

satisfaire[3]. Le degré de motivation lié à un besoin PN, appelé aussi `ddm(PN)` est fonction de la combinaison de plusieurs critères. Dans `testPriorityBetween(PN, PN')`, ces critères fonctionnent comme une sorte de filtres appliqués d'une manière séquentielle aux deux besoins. Le filtre suivant est appliqué si le filtre précédent n'a pas permis de les départager. Si au dernier filtre, il n'y a pas encore eu de besoins importants, alors on choisit au hasard entre les deux besoins. Les différents critères sont résumés par l'Equation (V.6).

$$\begin{aligned} ddm(PN) &= f(< type, niveau, catégorie, rang >, < état >, < ratio >) \\ &= f(< filtre1 >, < filtre\,2 >, < filtre3 >) \end{aligned} \tag{V.6}$$

On note `PN>PN'` le fait qu'à l'issue de `testPriorityBetween`, PN est plus important que PN'.

Le filtre 1 : " type, niveau, catégorie, rang ".

Les quatre critères de ce filtre sont tous appliqués avant de savoir si un besoin est réellement plus important qu'un autre ou non. Voici comment le fonctionnement de ce filtre est formalisé :

1. le *type* : ∀PN, PN'∈ Π, si `type(PN)==MN` et `type(PN')==HN` et si `PN.isUnSatisfied()` alors `PN>PN'`, c'est-à-dire que les besoins MN, étant plus proches des LN (rappel Figure III.3, page 79), ils sont toujours plus importants.
2. si deux besoins sont de même type, alors ∀`c1, r1, c2, r2` ∈ N, appliquer :

   a-$\texttt{PN}_{n1,c1\backslash r1} > \texttt{PN}_{n2,c2\backslash r2}$ lorsque `n1<n2`. (critère *niveau*). Cela signifie que, pour deux PN de même type, celui qui est de niveau inférieur (critères d'Abraham Maslow) est le plus important s'il n'est pas satisfait.

   b-$\texttt{PN}_{n1,c1\backslash r1} > \texttt{PN}_{n1,c2\backslash r2}$ lorsque `c1<c2`. ∀`r1, r2` (critère *catégorie*) :

[3] La sélection de motivations est donc une transposition de la sélection des besoins. Mais nous orientons ici la rédaction sous l'angle " motivation " et non " besoin ".

un besoin de catégorie inférieure, pour deux PN de même niveau, est toujours plus important s'il n'est pas satisfait.

3. c-$\texttt{PN}_{n1,c1\backslash r1}$ >$\texttt{PN}_{n1,c1\backslash r2}$ lorsque `r1<r2`. (critère *rang*). Ce dernier critère est une suite des précédents en en prolongeant le principe jusqu'au paramètre *rang* d'un AN (suivi du choix d'importance défini par l'utilisateur).

Règle de l'importance : Le filtre 1 nous amène à la définition de la règle de l'importance qui est la suivante : soit `PN` le besoin considéré comme important résultant des critères de ce filtre, et `PN'` l'autre besoin en cours de comparaison avec `PN`, alors : si `PN.isUnSatisfied()` alors `PN>PN'`.

Le filtre 2 : " état "

S'il n'existe pas de PN important issu du filtre précédent (c'est-à-dire que la règle de l'importance n'a pas pu être appliquée), le NIM passe ensuite à la comparaison de l'état de PN et PN' en utilisant la relation de l'Equation (V.7).

$$insuffisant/excessif > limit\_b/limit\_h > suffisant \tag{V.7}$$

L'Equation (V.7) signifie qu'un besoin à l'état `insuffisant` ou `excessif` par exemple doit être priorisé par rapport à un besoin à l'état `limit_b`, etc.

A ce niveau de filtre, il n'existe pas de critère pouvant départager l'état `insuffisant` et l'état `excessif`.

Le filtre 3 : " ratio "

Si le critère *état* ne permet pas de trancher entre les deux besoins, on applique le filtre contenant le critère *ratio*. La notion de ratio est très proche du *niveau d'activation* que l'on trouve dans les modèles animats (Maes 1991, Drogoul 2000). Ici, le ratio correspond à la position relative du curseur dans l'axe. Plus le curseur s'approche de (ou se trouve déjà dans) l'un des états `insuffisant` et `excessif`, plus le ratio diminue.

Soit `p_extreme` une valeur déterminée par la formule suivante :

- si `ccPos`≤`maxSat` alors `p_extreme=minPoint`
- si `ccPos>maxSat` alors `p_extreme=MaxPoint`

alors, le ratio s'obtient par l'Equation (V.8).

$$ratio = \frac{|ccpos - p\_extreme|}{|MaxPoint - minPoint|} \tag{V.8}$$

Le ratio se trouve ainsi toujours entre 0 et 1 puisque |`MaxPoint-minPoint`| équivaut à la longueur maximale de l'axe.

La règle de sélection entre deux besoins `PN` et `PN'` est alors donnée par l'Equation (V.9).

$$si\, ratio(PN) < ratio(PN') \Rightarrow PN > PN' \tag{V.9}$$

Cependant, soit $\varepsilon > 0$ avec $\varepsilon <<<0.1$ et |ratio(`PN`)-ratio(`PN'`)| $< \varepsilon$, faut-il aboutir automatiquement à la conclusion de l'Equation (V.9) ? En effet, la différence est trop petite pour permettre de tirer une conclusion quelconque. La solution à cette situation est que l'utilisateur définisse une marge appelée `marge_ratio` au-delà de laquelle on peut prendre en compte la différence entre deux ratios (Equation (V.10)).

$$si\, ratio(PN) + marge\_ratio \leq ratio(PN') \Rightarrow PN > PN' \tag{V.10}$$

Si le présent filtre ne permet pas non plus de distinguer deux besoins, alors le NIM choisit définitivement au hasard entre PN et PN'.

V. 3. 2 L'algorithme de sélection : des motivations aux actions

A chaque cycle du NIM, son objectif final, via la sélection des AN de l'application, est en réalité de pouvoir établir la liste de l'ensemble des *primitives* que l'agent sera motivé à exécuter durant ce cycle (rappelons que seules les primitives sont exécutables). Dans son algorithme de sélection, le NIM ne prend pas immédiatement tous les AN qui existent dans Π. Il va obtenir ces besoins par étapes. Cette obtention progressive des listes se

fait au travers de Ω, en commençant par les besoins finaux c'est-à-dire rappelons-le, les AN qui se trouvent au `nd`=1 du réseau.

Mais avant d'entrer dans les détails des étapes que le NIM suit dans le traitement des besoins, définissions ce que sont les *besoins en cours de satisfaction*, une situation très liée à l'algorithme du NIM.

Notions préliminaires : les besoins en cours de satisfaction

Un besoin PN est en cours de satisfaction (noté `PN.isInSatisfaction()`) s'il figure parmi les besoins que le NIM est en train de traiter. Déjà, lorsque `PN.isUnSatisfied()` alors `PN.isInSatisfaction()` est automatiquement `vrai`. D'un autre côté, `PN.isInSatisfaction()` devient `faux` dans les deux cas suivants :

- si `PN.isFullySatisfied()` est `vrai` ;
- ou si `PN.isSatisfied()` est `vrai` et que, *au niveau du nd où PN se situe*, l'action `PN.action[currenstate]` correspondante fait partie des actions choisies par l'agent dans le dernier cycle où elle a été traitée, mais qu'elle n'en fait plus partie dans le cycle suivant[4]. Dans ce cas, le traitement du besoin PN est dit *interrompu*.

A son initialisation, un PN n'est pas en cours de satisfaction.

Cette notion de " être en cours de satisfaction " sera importante dans l'algorithme du NIM. En effet :

- pour les BFI, si `PN.isInSatisfaction()`, alors il est pris en compte tant qu'il n'est pas interrompu et ce, même s'il est déjà à l'état `suffisant`.
- si `PN.isInSatisfaction()`, est `faux`, alors il n'est pas pris en compte, même s'il est déjà à l'un des états `limite`.

Cette approche permet à un PN d'être potentiellement choisi par le NIM même si ce dernier, venant de l'état `insuffisant` par exemple, se trouve déjà dans sa zone de sa-

[4] Attention : ces cycles ne sont donc pas forcément successifs, particulièrement lorsque nd>1. En effet, seules les actions en nd=1 sont considérées à chaque cycle, celles de nd>1 dépendant de ces dernières. Pour les besoins en cours de satisfaction dont nous parlons ici, il s'agit des successions de cycle où l'action au `nd` en question est prise en compte. Cela peut ainsi être au cycle n puis n + 4 puis n + 18, etc.

tisfaction. En effet, si nous n'adoptons pas cette approche, alors le curseur de tous les PN s'approchant de la zone de satisfaction seront " bloqués " autour de `minSat` et ne pourra pas progresser vers une position meilleure.

Etapes suivies par l'algorithme

Voici maintenant les détails proprement dits de l'algorithme, c'est-à-dire `algoNim(AC, nd)`. Soient les données suivantes :

- une liste `listAN` contenant la liste de tous les AN parmi lesquels le choix va s'effectuer,
- deux AN, l'un situé à la `i` -è place (notons-le provisoirement AN_i) dans `listAN`, et l'autre, AN_j, à la `j`- è place de cette même liste,
- la liste `finalPrim` qui va accueillir les primitives finales à exécuter, avec `finalPrim`={} (vide au début principal de l'algorithme)

La fonction `algoNim(AC, nd)` va se dérouler en les étapes suivantes :

0- Génération de la liste initiale avec le AC et le nd correspondant

Cette étape concerne la génération de `listAN` (vide au début de la fonction) à partir d'une AC. Soit `act` l'AC en question, la génération consiste à :

- prendre chaque `act1` / `isImpDirect(act, act1)`
- puis faire `listAN`=`listAN`+{`pn_satisf(act1)`}

Une fois `listAN` générée :

1- Rejet des AN / AN.isFullySatisfied()

Il s'agit d'éliminer tous les besoins qui sont déjà pleinement satisfaits. Les AN concernés ne seront donc automatiquement plus en cours de satisfaction.

2- Tri de la liste

Ensuite, trier la `listAN` restante, en rangeant les besoins du plus important au moins important, c'est-à-dire que $\forall$ `ANi`, `ANj` $\in$ `listAN`, si, `testpriorityBetween(ANi, ANj)` $\Rightarrow$ `ANj`>`ANi`, alors que `i`<`j`, alors `permuter(ANi, ANj)` dans `listAN`.

3- Rejet des AN / not AN.isInSatisfaction ()

Eliminer les AN qui ne sont pas en cours de satisfaction. Si aucun des AN restant dans `listAN` n'est en cours de satisfaction, le NIM passe directement à l'étape suivante.

4- Génération des actions, à partir des AN.action[currentState]

A partir de la liste triée restante, le NIM va générer une liste d'actions `lAct` contenant les `AN.action[state]` de chaque élément de `listAN`. Autrement dit,

si `listAN`=$\{AN_1, \ldots, AN_k\}$ alors
`lAct`=$\{AN_1$`.action[currentstate]`$, \ldots, AN_k$`.action[currentstate]`$\}$

C'est dans cette étape que débute le processus de sélection d'actions. En effet, pour chaque AN_x, toutes les actions autre que `act`=AN_x`.action[currentstate]` sont de facto éliminées (sauf si, bien évidemment, `act` est l'action par défaut). Nous sommes ici dans une des mises en œuvre *implicite* du `xor`.

5-Application, le cas échéant, des préconditions et du temps

Dans `lAct`, procéder à la sélection d'actions en éliminant celles dont le paramètre `precondition` n'est pas satisfait ou dont le moment d'exécution n'est pas encore venu. Nous le formalisons comme suit :
$\forall$`act` $\in$ `lAct`, si `not preCondOK(act)` ou si `not isTimeOK(act)` $\Rightarrow$éliminer `act`.

6- Séparation PR/AC

Pour tous les éléments `act` $\in$`lAct`, faire :
- si `act` est une PR$\Rightarrow$`finalPrim`=`finalPrim`+$\{$`act`$\}$
- si `act` est une AC, `finalPrim`=`finalPrim`+$\{$`algoNIM(act, nd+1)` $\}$ et on revient à l'étape 0 pour exécuter `algoNIM(act, nd+1)`.

Fin d'étapes

Lorsqu'il n'y a plus de AC à traiter, le NIM prend ensuite le `finalPrim` résultat et va appliquer la notion de conflits sur ses éléments. Plus précisément, $\forall$ pr_i, $pr_j \in$ `finalPrim`, `i` et `j` indiquant leur rang respectif dans `finalPrim` avec `i` $<$ `j`, et si `isXor(`pr_i`,` pr_j`)`, alors éliminer pr_j de `finalPrim`. Pourquoi pr_j ? Car les primitives de `finalPrim` proviennent toutes de `listAN` qui a été préalablement triée à l'étape 3 de chaque niveau `nd` de la sélection. Le tri se faisant du plus important (c'est-à-dire de la gauche de la

liste) au moins important (à la droite de la liste), il est normal qu'au niveau des primitives, c'est celle qui se trouve à droite (c'est-à-dire ayant un rang plus élevé) qui soit éliminée.

Au final, `finalPrim` contiendra la liste des PR liées entre elles par le connecteur `and`.

Nous voyons que dans `algoNIM`, les primitives issues des besoins moins importants ne sont éliminées que lorsqu'elles sont en conflit avec celles dérivant de besoins plus importants. Cela permet à l'agent d'exécuter potentiellement des actions, même si celles-ci sont moins importantes.

Mais que se passerait-il si, à la fin de tout le processus précédent, `finalPrim` ne contenait aucune action ? autrement dit, `finalPrim` ={} ?. Dans ce cas (très rare), il existe une primitive par défaut que nous appelons `doNothing()` qui sera exécutée par l'agent, c'est-à-dire que `finalPrim` ={}⇒ `finalPrim` ={`doNothing`}. Par défaut, cette primitive ne fait rien. Mais elle peut être redéfinie par l'utilisateur au niveau applicatif.

V. 3. 3 Quelques remarques

Remarques de synthèse sur l'algorithme

Le fonctionnement de l'agent se manifeste par une exécution successive et en boucle de l'algorithme du NIM et de l'exécution de la liste de primitives qui en est issue. Aussi, il est fort possible que deux primitives exécutées consécutivement ou simultanément ne proviennent pas des mêmes motivations. Et comme le résultat de l'exécution du cycle précédent est imprévisible, les besoins pouvant évoluer tous en même temps, un contrôle total du système doit être refait au cycle suivant (exécution de `algoNIM(net_entry, 1)`).

Remarquons que le parcours complet de Ω n'est pas systématique car le passage au `nd` suivant du réseau dépend du résultat de la sélection des AN au niveau précédent, et plus particulièrement, de la structure (niveau, rang, etc.) et des états de ces AN. Ces états étant à leur tour le résultat de l'exécution des primitives du cycle précédent.

Justification de la généricité des critères

Du point de vue conceptuel, le NIM présente des caractéristiques génériques car la structure et les critères proposés ici demeurent applicables quel que soit le type d'agent et la sémantique des concepts utilisés par les applications.

En voici les points essentiels :

- à la base de cette thèse, nous considérons que les critères de Abraham Maslow (c'est-à-dire les cinq niveaux) (rappel Section III.2.3) sont applicables en toutes circonstances. En outre, la signification des besoins est abstraite (physiologique, sécurité, etc.). Chaque application peut les instancier en fonction de ses objectifs. La mise en place des besoins suivra implicitement une hiérarchie bien définie par Abraham Maslow.
- les LN qui sont la base du modèle traduisent le naturel. Ils se manifestent indépendamment de l'activité de l'agent. Par exemple, une personne peut toujours avoir faim qu'elle soit mécanicien ou banquier (question de rôle), ou encore qu'elle soit seule ou en famille (question de position sociale). Même s'il y a des contraintes sémantiques à suivre (ex : faim=niveau 1), cela ne pose pas de problème dans la mesure où la contrainte (si elle en est vraiment une) est universellement acceptée.
- notre modèle n'impose aucune contrainte sémantique au niveau des HN. L'utilisateur est libre d'interpréter la sémantique des HN. De toute façon, ces HN convergeront vers les LN qui leur sont associés.
- le critère `état` est évidemment lui aussi valable car il fait partie de la description interne d'un besoin PN.
- le critère `ratio` est obtenu à partir de formules mathématiques dont les opérandes figurent parmi les éléments de description interne du besoin.
- la hiérarchie établie par Maslow, associée aux règles de gestion des besoins, le tout intégré dans notre modèle, entraîne implicitement un balancement entre individualité et socialité. Ainsi, si le dernier besoin le plus important se trouvait au niveau 1 et si le besoin courant à traiter se situe au niveau 3, il existe *implicitement* un balancement entre le niveau individuel et social.

V. 3. 4 Résumé schématique du système global

La figure V.1 présente une vue globale du système MASLOW tandis que la figure V.2 résume l'algorithme du NIM.

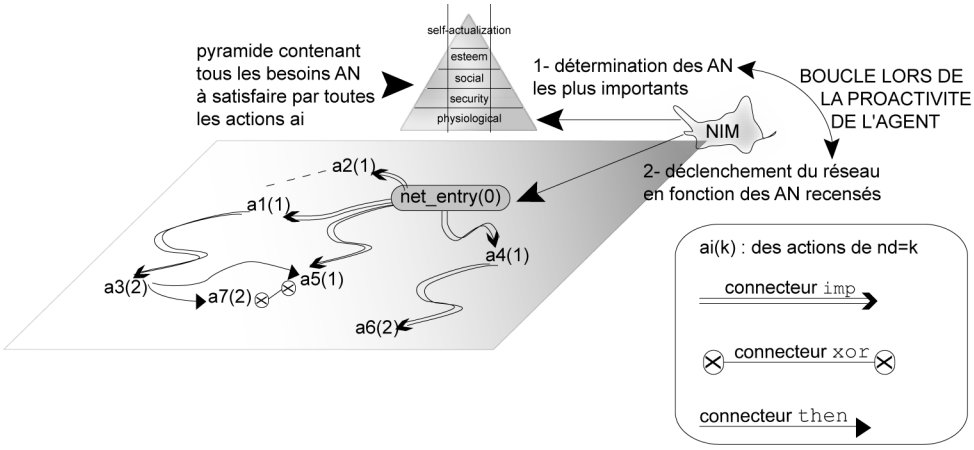


Figure V.1 : Vue globale du système Maslow dont le réseau Ω est inspiré de ANA (Maes, 1991)

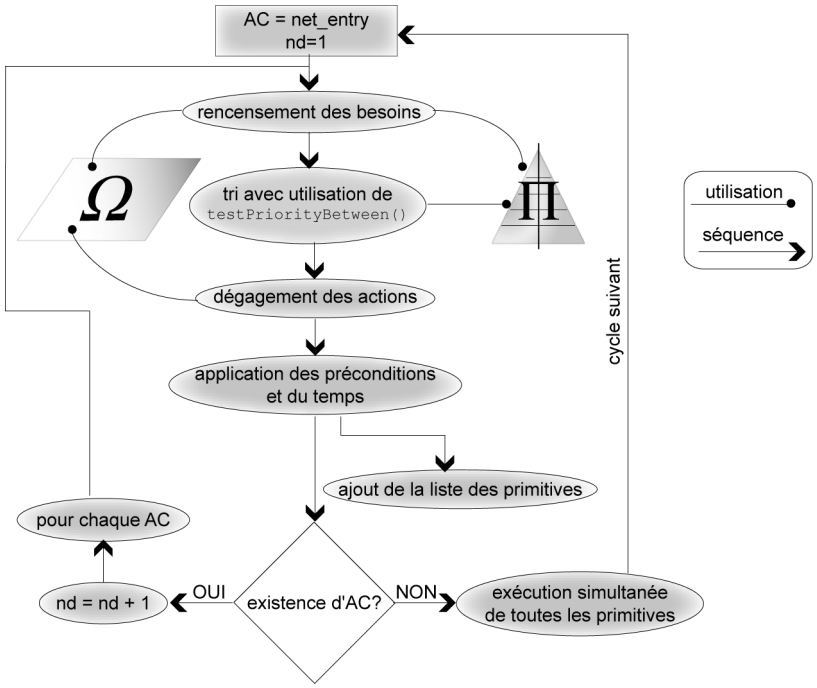


Figure V.2 : Résumé schématique de l'algorithme du NIM

V. 4 Informations complémentaires sur l'architecture

La partie que nous avons étudiée dans cette thèse, c'est-à-dire MASLOW, concerne uniquement le système conatif d'un agent, c'est-à-dire une partie seulement de son architecture. Il convient donc dans cette section de fournir davantage d'informations sur les autres parties qui, bien que n'étant pas un objectif en soi de la thèse, figurent toujours intrinsèquement dans un agent et doivent, de ce fait, être mentionnées. Nous expliquerons également ici la plate-forme multi-agent sur laquelle nous effectuerons notre simulation : il s'agit d'ADK (pour Agent Developer Kit)[5].

V. 4. 1 La plate-forme ADK

ADK est une plate-forme agent développée par Calderoni (2002). Elle a été initialement conçue pour simuler des agents réactifs dans le cadre de la Vie Artificielle. La spécialisation de ADK dans le monde de la robotique se nomme RDK (pour Robot Developer Kit). RDK utilise les modèles d'action par combinaisons tendancielles (ex : Zeghal, 1994) dans lequel l'environnement est conçu comme un espace métrique, et les déplacements des agents, modélisés sous forme de vecteurs. Nous mettons l'accent sur RDK car pour la validation expérimentale du travail, c'est la robotique collective qui est utilisée.

Les agents et l'environnement dans ADK

L'architecture d'un agent ADK est basée sur trois composants (Figure V.3) :

- des *senseurs* : ils servent à capter les informations de l'environnement, sous forme de *percepts* (c'est-à-dire d'informations obtenues à partir de la perception). Au niveau de RDK, nous pouvons donner des exemples relatifs à un robot tels les senseurs visuels, les senseurs du toucher, etc.
- des *effecteurs* que l'agent utilise pour agir dans l'environnement.
- entre les percepteurs et les senseurs se trouve le *contrôleur* (qui est le *système conatif*) de l'agent.

[5] Revoir aussi la section I. 1. 2, page 8

Chaque agent dispose en outre d'un ensemble de propriétés physiques qui le caractérisent comme par exemple son identifiant, sa position dans l'environnement, etc.

Pour un agent `a` $\in$ `A` (où `A` constitue l'ensemble des agents du système ADK), son environnement représente tout ce qui n'est pas lui, c'est-à-dire $\overline{a} = S - \{a\}$ où `S` est le système entier. La partie de l'agent liée à l'environnement, c'est-à-dire c'est ce qui reste de l'agent si on lui ôte le système conatif s'appelle *composante environnementale* de l'agent.

Outre l'agent, l'environnement contient aussi des objets situés disposant aussi d'un ensemble de propriétés dont la plus évidente est la propriété `position` (dans l'environnement).

De la perception à l'action

Lorsque l'agent souhaite agir dans l'environnement suite aux percepts reçus, le contrôleur va d'abord produire ce que l'on appelle des *commandes* qu'il va ensuite envoyer aux effecteurs[6]. A partir de ces commandes, l'agent va alors produire des *influences* (dites aussi des actes) dans l'environnement. L'environnement, en fonction de ses propres lois[7] dites aussi *lois de l'univers* va se manifester par rapport à ces influences envoyées par les agents. Le résultat de cette " rencontre " entre les influences et les lois s'appellent *action*. Au sens ADK, l'action correspond donc à la définition dans Ferber (1997) qui la considère comme la transformation d'un état global (ex : modification de propriétés d'un ensemble d'agents) en un autre état global.

La notion d'action telle qu'elle est intégrée dans MASLOW correspond dans ADK aux influences. Cependant, pour conserver la ligne d'idées que nous avons développée jusqu'à présent, nous allons conserver, pour MASLOW, le terme " action " au lieu de " influence " surtout que la notion d'actions selon ADK ne sera plus explicitement citée dans ce document. Il ne risque donc pas d'y avoir une confusion.

---

[6] Ces commandes décrivent les opérations qui doivent être exécutées par l'agent dans son environnement via les effecteurs.

[7] Ces lois (rappel page 24 dans la section sur les actions) se présentent ici sous forme de *phénomènes* qui sont intégrés dans l'environnement. Un exemple de ce phénomène est celui de la loi de la gravité. Un agent qui tente de soulever un objet est soumis à la réaction de l'environnement via cette loi. Mais cette même loi peut aussi aider l'agent dans ses objectifs, et sans son intervention. Par exemple, faire glisser un objet lourd sur une pente au lieu de le transporter soi-même.

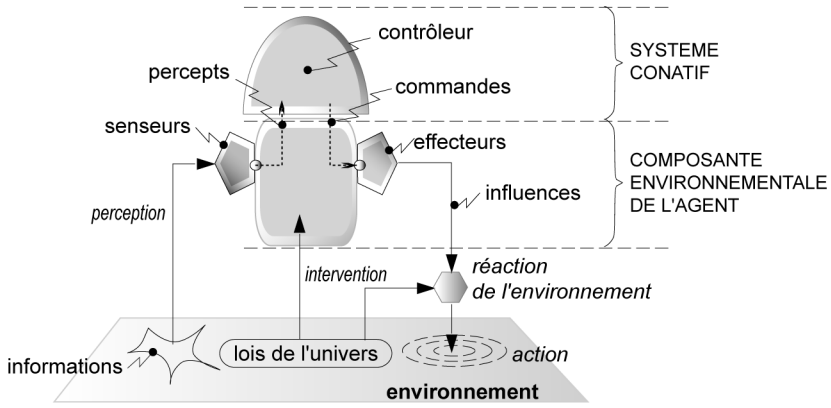


Figure V.3 : Présentation générale du système ADK

V. 4. 2 MASLOW et ADK

Généralités

La connexion du modèle MASLOW dans ADK se fait au niveau du système conatif de l'agent. En particulier, notre travail concerne l'intégration d'un mécanisme générique de sélection au sein du système motivationnel de celui-ci. Jusqu'ici, ce sous-système de ADK était plutôt vide.

Mais il existe également d'autres concepts qui sont peu (ou pas du tout) considérés dans ADK et que nous avons donc dû prendre en compte via MASLOW. C'est ce que nous expliquons ci-après.

Le mécanisme cognitif de MASLOW

Comme ADK est initialement basé sur des agents réactifs uniquement, il ne contient donc aucun composant quelconque qui gérerait le concept cognitif. Nous introduisons alors au niveau de MASLOW un minimum de mécanisme permettant la gestion des croyances (“ *belief* ”) de l'agent, ainsi que leur mémorisation. A noter cependant que l'introduction de ces concepts au niveau applicatif n'est pas systématique mais dépend de l'agent que l'utilisateur modélise.

La gestion de la croyance se fait dans MASLOW via le concept générique *d'attitude informationnelle* qui a été initialement adopté par Brazier et al. (1999) dans leur modélisation

de la gestion d'informations dans le mental d'un agent cognitif. Ce concept nous permet de modéliser la manière dont l'agent va pouvoir représenter *partiellement* son environnement et *mémoriser* les informations qui en sont issues pour des utilisations futures. La *croyance* de l'agent dérive de ce concept.

Une information `I` mémorisée par l'agent peut être classée dans l'un ou l'autre des trois types `IT` (pour Information-Type) suivants :

- $\texttt{IT}_1$ de la forme {`entité`}. Lorsque l'agent dispose d'une information de ce type (notée `agent.hasInfo(entité)`), cela stipule qu'il adopte une attitude montrant sa possession d'une information sur l'existence de `entité`. Ce paramètre `entité` peut être n'importe quel élément : agent, environnement, objet, situation, etc.
- $\texttt{IT}_2$ de la forme {`entité, <`$\texttt{prop}_1$`, `$\texttt{prop}_2$`, . . .>`}. La proposition `agent.hasInfo(`$\texttt{IT}_2$`)` signifie alors que l'agent adopte une attitude montrant qu'il dispose d'une information sur l'existence de `entité` et que cette entité possède un ensemble de propriétés $\texttt{prop}_i$.
- $\texttt{IT}_3$ de la forme {`entité, <(`$\texttt{prop}_1$`, `$\texttt{val}_1$`), . . ., <`$\texttt{prop}_n$`, `$\texttt{val}_n$`>`}. La proposition `agent.hasInfo(`$\texttt{IT}_4$`)` précise que l'agent adopte une attitude montrant qu'il dispose d'une information sur l'existence de `entité`, possédant un ensemble de propriétés $\texttt{prop}_i$ ayant respectivement pour valeur $\texttt{val}_i$.
- $\texttt{IT}_4$ de la forme {`entité, <`$\texttt{val}_1$`, . . ., `$\texttt{val}_n$`>`}. La proposition `agent.hasInfo(`$\texttt{IT}_3$`)` précise que l'agent adopte une attitude montrant qu'il dispose d'une information sur l'existence de `entité`, possédant un ensemble de valeurs $\texttt{val}_i$.

Les informations dont l'agent dispose sont ajoutées dans sa mémoire via l'instruction `memorize(`$\texttt{info}_1$`, `$\texttt{info}_2$`, . . .)` où $\texttt{info}_i$ est de type `IT`. Toutes ces informations peuvent évoluer en fonction de l'activité et du comportement de l'agent dans l'environnement.

Le prédicat `hasInfo()` est générique. Dans le cadre des croyances, il est spécialisé en `hasBelief()`.

La gestion du temps dans la simulation

Initialement, la gestion du temps dans ADK a été représentée par une simple valeur `timer` qui s'incrémente automatiquement à chaque pas de temps de la simulation. Un *pas* correspondait uniquement à un délai d'attente contenu dans une variable appelée `period` (en *millisecondes* temps machine) dont la valeur a été attribuée par l'utilisateur. Si cette approche est simple et suffisante au niveau simulateur, elle permet difficilement d'interpréter l'évolution temporelle de la simulation pour l'utilisateur.

Aussi, notre intervention à ce niveau a été de donner à la notion de temps un sens plus facilement compréhensible par l'utilisateur ordinaire. Cela se fait en introduisant une structure gérant le calendrier grégorien[8] dans le système. Ainsi, lorsque `period` atteint les 1000 millisecondes, une propriété `seconde` de cette structure est incrémentée. Si `seconde` atteint 60, une autre propriété appelée `minute` de la structure est incrémentée, etc. Le même processus se poursuit aux mois et aux années.

La valeur `timer` ci-dessus a donc été transformée en une structure plus complexe mais plus adéquate aux habitudes des utilisateurs. Cette nouvelle structure de `timer` peut être lue sous la forme " `année/mois/jour heure/minute/seconde` ". En fait, cette structure a aussi été adoptée pour se rapprocher de la notion de temps spécifiée dans MASLOW (rappel Section IV.3.3, page 99).

V. 5 Conclusion du chapitre

Ce chapitre conclut la partie théorique de cette thèse. Au travers des concepts proposés dans les chapitres précédents, il décrit tout d'abord de quelle manière les besoins, les actions et les motivations sont techniquement mis en œuvre dans le système motivationnel de l'agent hybride via notre modèle Maslow. Cela s'est fait via 3 composants importants : une pyramide notée Π, un réseau d'actions noté Ω et, à un niveau générique, le NIM (pour Need Importance Manager) qui est l'axe fondamental de la proactivité de l'agent.

En outre, comme notre travail avec MASLOW ne traite qu'une partie de l'*architecture* et du *comportement* d'un agent, nous avions donc aussi besoin de mentionner les autres

[8] Le calendrier grégorien est notre calendrier usuel, utilisant les paramètres connus tels que 1 année=365 jours (sauf année bissextile), le mois de janvier=31 jours, 1 minute=60 secondes, etc.

parties pour une compréhension globale du système et notamment, la plate-forme qui va servir de support à ce travail : ADK.

Cette explication " simultanée " de MASLOW et de ADK nous permet de mieux comprendre le niveau de contribution conceptuel de MASLOW dans ce travail, une contribution qui doit être validée au travers d'applications. C'est ce que nous essayerons de montrer au chapitre VI lors la partie expérimentation.

Chapitre VI

Expérimentation

C'est dans ce chapitre que nous détaillons les expérimentations effectuées dans le cadre de cette thèse, ainsi que les résultats obtenus. Mais si l'analyse des résultats entre dans le cadre de ce chapitre, la discussion proprement dite du modèle, suite à cette expérimentation, ne se fera que dans le chapitre VII.

Dans ce chapitre, nous parlerons

- en premier lieu, de l'introduction à l'implémentation de notre modèle (Section VI.1),
- puis par la suite le cadre de notre expérimentation (Section VI.2),
- suivis des scénarios expérimentés proprement dits (Section VI.3)
- et enfin l'analyse des résultats (Section VI.4).

VI. 1 Préambule : implémentation du contrôleur

Introduction

C'est durant cette phase d'expérimentation que l'implémentation de ce travail (et donc plus particulièrement l'implémentation du contrôleur) sera progressivement expliquée. En effet, c'est à partir de l'évolution de l'implémentation du contrôleur que nous allons (en partie) valider ce travail. Cette validation va notamment se faire en montrant la

différence de résultat entre une architecture de contrôle préprogrammé et une architecture de contrôle générique que nous allons mettre en œuvre via MASLOW.

Mentionnons déjà que ADK et MASLOW sont tous deux implémentés en JAVA. Les syntaxes utilisées dans les sections qui suivent respectent donc les spécifications de ce langage.

Le contrôleur abstrait d'ADK

Dans ADK, le contrôleur est implémenté d'une manière abstraite via une classe nommée `Controller`. L'évolution du système s'effectue à chaque pas de simulation c'est-à-dire qu'à chaque incrémentation de l'unité de temps, c'est une méthode abstraite `step()` de la classe `Controller` qui est invoquée par chaque agent du système. La démarche d'implémentation de ce mécanisme est identique à celle que nous avons adoptée dans (Andriamasinoro et al., 2001b) qui consiste pour le simulateur à lancer la méthode `step()` de chaque agent, puis une fois le dernier agent traité, le pas de temps est incrémenté et le simulateur revient au premier agent et ainsi de suite.

Au niveau de RDK, la classe `RobotController` surcharge la méthode `step()` de `Controller` pour introduire les spécificités propres à des robots : gestion des mouvements, rotation, etc.

VI. 2 Cadre d'application : les fourrageurs et les paysans

Ce travail est appliqué dans deux cadres. Le premier reprend et étend le problème de fourragement des robots, introduit par Calderoni dans son travail. Le second entre dans un cadre plus réel, à savoir le comportement de paysans vivant à Mangatany.

Le résultat recherché à partir de ces deux applications n'est pas totalement le même. Avec les fourrageurs, nous voulons montrer que notre modèle basé sur les motivations naturelles s'adapte bien dans un contexte *hybride* c'est-à-dire aussi bien aux agents réactifs qu'aux agents cognitifs. Avec l'ajout du cadre des paysans, nous cherchons à prouver que le modèle est bien générique.

VI. 2. 1 Les fourrageurs

Le principe se base ici sur le scénario dans lequel des robots fourrageurs (`forager`) explorent des minerais et les ramènent à une base. Ici, le système est alors composé d'un environnement à deux dimensions contenant les entités suivantes (Figure VI.1) :

- des robots, qui constituent aussi les agents du système,
- un ensemble d'obstacles que ces robots doivent éviter,
- des palets rouges (`red_puck`) et des palets bleus (`blue_puck`), simulant les minerais,
- et deux bases, l'une rouge (`red_base`) et l'une bleue (`blue_base`).

L'objectif des robots est de trouver des palets puis de les ramener à la base (`deliverAllPucks`). Pour cela, ils vont d'abord explorer l'environnement (`explore`) d'une manière aléatoire pour trouver des palets. Chaque robot dispose d'une pince (`gripper`) qui ne peut cependant contenir qu'un palet à la fois. Si le robot trouve un palet rouge, il le prend avec sa pince (`acquireRedpuck`) et le dépose à la base rouge (`deliverRedpuck`) tout en évitant (`avoid`) les éventuels obstacles sur sa route. Le principe est le même pour les palets bleus (`acquireBluepuck` et `deliverBluepuck`). Le nombre total de palets à transporter pour notre exemple est de 10.

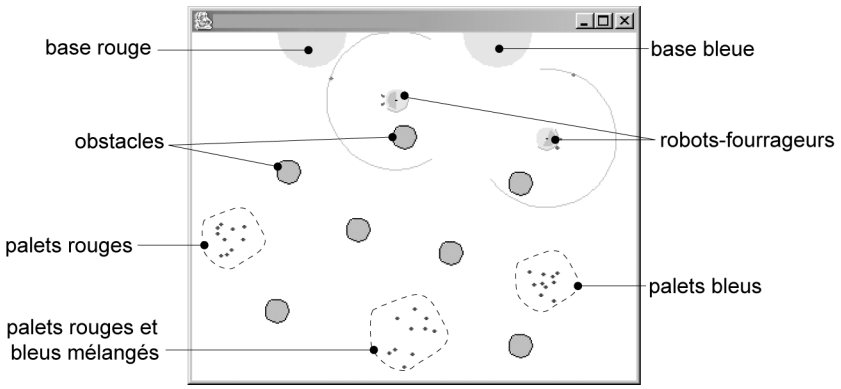


Figure VI.1 : Scénario de base des robots fourrageurs

VI. 2. 2 Les paysans

Généralités

Dans ce cadre (Figure VI.2), nous simulons l'activité quotidienne des paysans. Le travail d'un paysan consiste à aller de bonne heure à la rizière (`goToRiceField`) pour cultiver le riz (`CultivateRice`). Pour le moment, nous avons considéré uniquement la phase où il retourne la rizière (`turnRiceField`). En parallèle avec cette phase de labour, le paysan doit aussi installer des épouvantails (`installScareCrow`) pour faire fuir ce que l'on appelle à Madagascar les " fody ", des petits oiseaux qui picorent souvent les épis de riz. Le nombre d'épouvantails à installer est de 2.

Il arrive que durant son travail au champ, le paysan prenne une sorte de goûter (`snack`), ou boive (`drink`). Il peut même se reposer (`pause`), voire dormir (`sleep`).

Le midi ou le soir, il rentre chez lui (`goHome`) pour déjeuner ou pour dîner (`eatAtHome`). Après le dîner, il va garder le village, et principalement au niveau de son portail[1] (`keepPortal`). La garde du village se situe ici dans un contexte de prévision, c'est-à-dire uniquement pour des raisons générales de sécurité (donc dans un cadre HN). Il est à ne pas confondre avec le cas où des envahisseurs sont *actuellement* en train d'attaquer le village. Dans ce cas, c'est un autre besoin, de type MN cette fois, qui aurait été introduit.

Certes, ce scénario du paysan n'est pas complet. Par exemple, le fait de dormir la nuit n'est pas mentionné. La raison en est que nous ne voulons pas trop augmenter la complexité du présent scénario, tout en le rapprochant le plus possible de la réalité du village de Mangatany.

" Simulation " des besoins à cadre social

Parmi les besoins d'un paysan, il en existe quelques uns dont la modélisation demande une intervention du cadre social, c'est-à-dire celle d'autres agents[2]. Tel est le cas de

[1] L'accès à bon nombre de villages à Madagascar se faisait généralement (mais plus forcément aujourd'hui) via des portails. Nous simulons néanmoins le phénomène qui n'est finalement pas loin de la réalité.

[2] Le cadre social est à ne pas confondre avec le besoin social de la pyramide de Maslow. Le cadre social peut se situer à tous les niveaux de la pyramide, comme par exemple manger ensemble (besoin physiologique) mais à une même table.

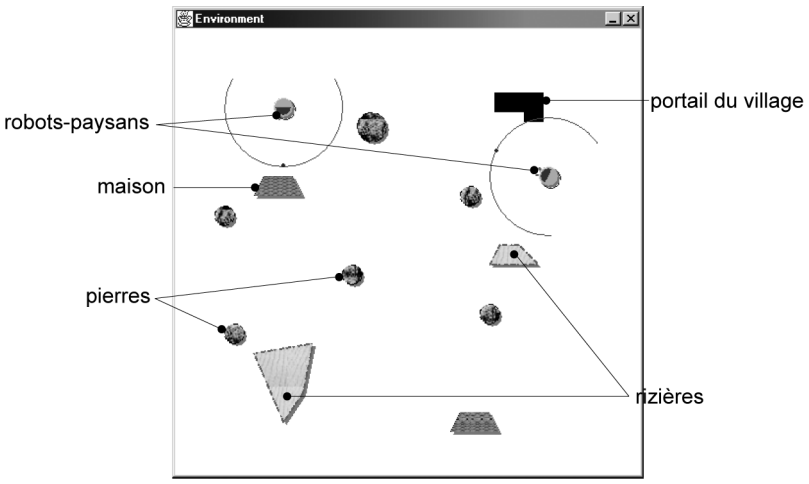


Figure VI.2 : Scénario de base des paysans

la garde du village qui, bien que situé au niveau *sécurité* (niveau 2) de Π, demande dans la réalité une organisation sociale complexe entre les villageois. Un autre cas réel (que nous n'introduisons que maintenant) concerne aussi le respect des coutumes (`respectCustom`) par les habitants du village. La modélisation de ces besoins en tant que AN demande une introduction des autres agents.

Il se trouve cependant que pour le moment, ADK ne tient pas compte de l'interaction sociale entre les agents, bien que ces derniers comme nous le savons, disposent de besoins sociaux. Il ne nous est donc pas possible ici de modéliser une organisation sociale des agents pour la garde du village, afin de montrer d'une manière réaliste que ce besoin de garder le village est satisfait ou non[(3)]. L'obtention de la valeur de `ccpos` pour ces besoins demande la présence d'autres agents. La situation est la même pour le besoin de respecter les coutumes. La solution provisoire que nous adoptons consiste à *simuler* ces besoins c'est-à-dire que nous les considérons comme étant toujours *insatisfaits*. De ce fait, l'agent va *en permanence* les prendre en compte dans son comportement. Ce sera lors de la sélection d'actions que leur action courante sera éventuellement éliminée.

Le but ici serait uniquement de savoir si ces besoins sont effectivement pris en compte en fonction des critères appliqués par l'algorithme du NIM. Une fois introduit dans ADK le

[3] C'est aussi la raison pour laquelle nous n'avons pas introduit le cas où des envahisseurs attaquent de temps en temps le village (cas qui existe pourtant dans la réalité), nécessitant ainsi un besoin MN de refouler les envahisseurs.

concept d'interaction sociale entre agents, nous pourrons progresser dans la modélisation de ces besoins qui font intervenir des mécanismes sociaux inter-agents.

VI. 3 Implémentation et comparaison de scénarios

VI. 3. 1 Présentation

Dans l'ensemble de l'expérimentation, quatre scénarios (a, b, c, d) sont mis en œuvre. Les trois premiers concernent des fourrageurs et le dernier, des paysans. En voici l'aperçu :

- Scénario (a) : il utilise la version initiale de ADK/RDK telle que Calderoni l'a implémentée. Il n'intègre donc pas encore MASLOW. La simulation porte ici sur des agents réactifs. Et pour que ces robots réactifs puissent retrouver le chemin de la base après la prise d'un palet, ces bases émettent en permanence des signaux dans l'environnement.
  Ce scénario servira de *scénario témoin* à l'évaluation de cette thèse.
- Scénario (b) : nous reprenons (a) et nous y intégrons MASLOW. Le scénario porte aussi sur des agents réactifs.
- Scénario (c) : nous expérimentons MASLOW avec des fourrageurs cognitifs qui disposent d'une représentation partielle de leur environnement.
- Scénario (d) : nous reprenons (c) mais cette fois-ci, les robots sont des paysans et non plus des fourrageurs.

Quels que soient les scénarios, notre simulation débute toujours à la date du 1^{er} janvier 2003 à 0h. Plus précisément, on a " `01/01/03 00 :00 :00` ".

Scénario (a) : des fourrageurs réactifs sans Maslow

Dans ce scénario (voir Figure VI.3), tout le mécanisme de sélection d'actions[4] du robot est directement préprogrammé par l'utilisateur au niveau applicatif (cf. classe `ForagerController` une classe dérivée de `RobotController` de RDK). Comme nous l'avons dit, c'est ce comportement préprogrammé qui servira donc de comportement *témoin* pour valider notre travail.

[4] Nous employons ici directement le terme " sélection d'actions " car dans ce scénario (a), il n'y a encore eu aucune notion de motivations intégrée dans l'agent.

Ici, les robots ressemblent quasiment à des automates à états finis dans lesquels l'état va déclencher l'exécution de l'action donnée du même nom. Par exemple, si l'agent est à l'état `EXPLORE`, alors l'agent exécute l'action `explore()`. Le passage d'un état à un autre se produit après qu'une certaine condition associée à l'état courant a été remplie. Par exemple, si l'agent aperçoit un palet rouge, alors l'état `EXPLORE` passe à l'état `ACQUIRE_RED` et l'agent va exécuter l'action `acquireRedPuck()`. Cette transition d'états est implémentée dans une classe dérivée de `ForagerController` nommée `HandCodedController`. Comme le nom de cette dernière classe l'indique, le passage d'un état à un autre est aussi préprogrammé.

A noter qu'en outre, l'évitement d'obstacle ne fait pas partie des actions à sélectionner c'est-à-dire qu'il n'y a pas d'états EVITER_OBSTACLE. Ce comportement est implicitement exécuté (si obstacle il y a) en combinaison avec les autres actions qui, comme nous l'avons vu, sont associées à des états.

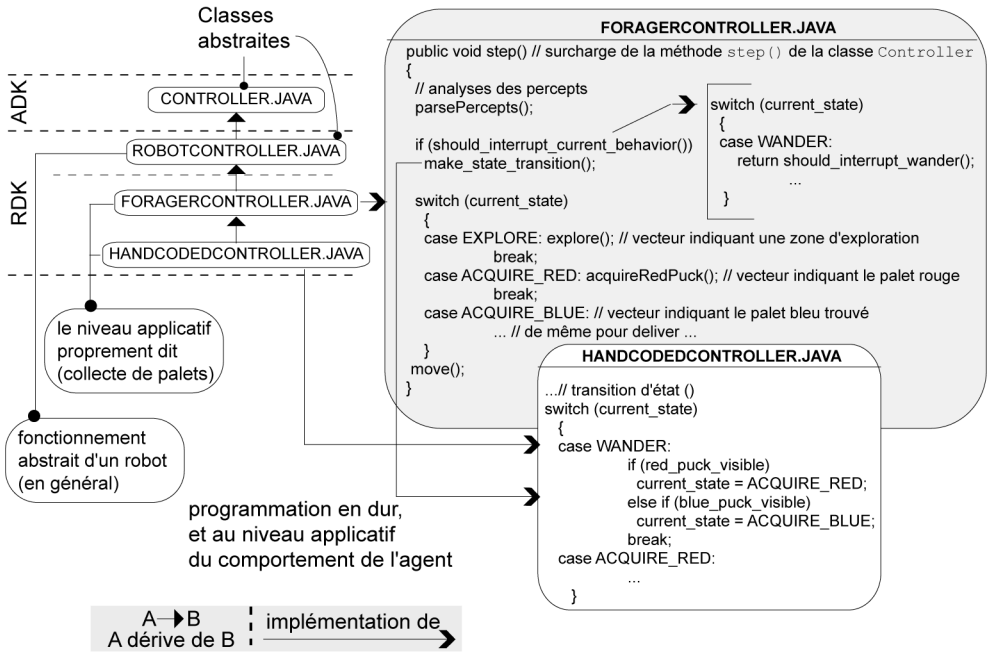


Figure VI.3 : ADK/RDK sans Maslow : un mécanisme de sélection préprogrammé.

Scénario (b) : des fourrageurs réactifs intégrant Maslow

Avec l'intégration du mécanisme de sélection des motivations de MASLOW dans le contrôleur du robot, les différences avec le scénario (a) sont les suivantes :

- la classe `HandCodedController` est éliminée car le robot n'est plus maintenant un simple automate " guidé " par l'utilisateur qui, dans (a), avait plutôt abstrait les motivations du robot. Cette fois-ci, le comportement de l'agent dépendra d'une manière générique de ses motivations, déterminées dynamiquement par le NIM.
- le mécanisme de sélection préprogrammé précédemment implémenté dans la classe `ForagerController` via la méthode `step()` est enlevé, la sélection étant gérée par le NIM. En conséquence, il est possible de faire monter cette méthode vers une classe abstraite mère que nous appelons `AnimatController`, une dérivée de `RobotController`. La classe `ForagerController` ne sert désormais plus qu'à implémenter les actions à invoquer au niveau applicatif par le NIM (Figure VI.4).

C'est dans `AnimatController` que le NIM est inséré. L'algorithme du NIM, que nous avons décrit dans la section V.3.2 y est appelé au travers d'une méthode `check` (). La connexion du NIM avec ADK se fait par l'intermédiaire d'une interface nommée `PyramidManager`. Ce " pont " permet à MASLOW d'être structurellement indépendant d'ADK c'est-à-dire que son évolution future ne dépendra pas de celle d'ADK[5] Et si nous projetons de connecter MASLOW à d'autres modèles[6], nous modifions `PyramidManager`. Le NIM reste inchangé.

L'adoption du terme " AnimatController " fait référence aux animats qui intègrent donc les motivations naturelles de base, tout comme les animaux (ou la partie animale de l'homme) dans le cadre réel. Comme notre approche stipule que le point de départ commun de tous les agents, c'est le naturel, il est donc tout à fait normal que ce soit dans `AnimatController` que le NIM est introduit.

---

[5] L'inverse est par contre être vrai lorsque ADK va définitivement intégrer MASLOW au niveau générique de son système motivationnel.

[6] A rappeler que notre étude concerne ici uniquement les modèles d'agents situés, conformément au cadre de conception de MASLOW qu'est la Vie Artificielle.

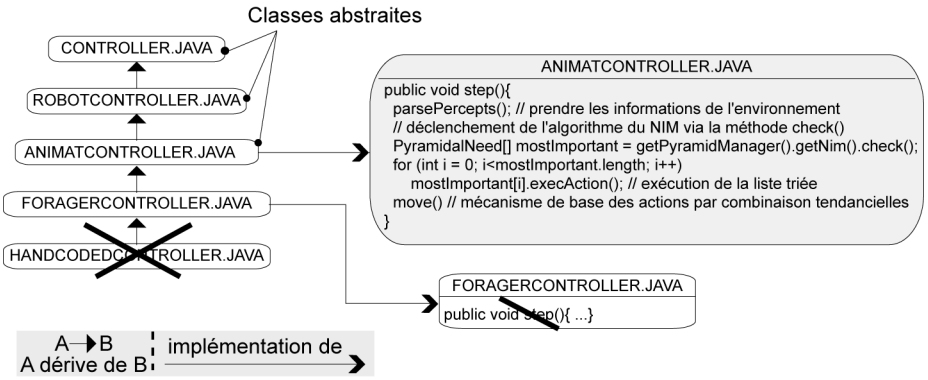


Figure VI.4 : Résultat de l'introduction de Maslow dans ADK via le scénario (b)

Quelques précisions méritent d'être apportées à partir de ce scénario :

- par rapport à (a), nous ajoutons ici le fait que les robots peuvent boire ou faire une pause. L'enjeu est de montrer que nous pouvons introduire plus de concepts sans reprogrammation du contrôleur, tout en arrivant toujours à reproduire le mécanisme de fourragement.
- la classe appelée `Obstacle` qui implémente les obstacles dans RDK (rappel Figure VI.1, page 129) a été renommée en `Stone`, c'est-à-dire que nous considérons ces objets circulaires comme des pierres. Si, à première vue, cela n'apporte rien dans l'évolution du système (puisque c'est juste un changement de nom), il est important pour sa compréhension, par un agent Maslow. En effet pour ce dernier, le besoin d'éviter un obstacle est un concept générique LN[(7)] qui se manifeste d'une manière ou d'une autre au niveau applicatif et en l'occurrence ici, par des pierres. En outre, dans de travaux futurs, nous pensons gérer ce concept d'obstacles d'une manière plus approfondie : un obstacle est en fait issu de la représentation mentale que l'agent fait d'un entité physique donnée (maison, autre agent, voiture, etc.). En d'autres termes, il n'y a pas d'entités physiques prénommées " obstacle ".

[7] Rappelons-nous du besoin *ListLN.need_obstacle_away* mentionné à la section III.3.3.

Scénario (c) : des fourrageurs cognitifs intégrant Maslow

Ces agents cognitifs disposent d'une représentation partielle de leur environnement, en ayant précisément connaissance de deux endroits : le dernier lieu où un palet a été trouvé, et la position physique des deux bases. De ce fait, pour le présent scénario, les signaux émis par ces bases sont enlevés[8].

C'est donc dans ce scénario que nous introduisons les concepts cognitifs développés dans la section V.4.2, à savoir la représentation mentale et le système de mémorisation de l'agent. Cette faculté de représentation partielle de l'environnement est gérée dans MASLOW au travers d'une classe abstraite nommée `InformationalAttitude` qui est déclarée dans une classe abstraite nommée `Memory`. La classe qui intègre les croyances de l'agent (classe appelée `Belief`) dérive de `InformationalAttitude` et est intégrée à son tour dans la classe `ForagerMemory`, une dérivée de `Memory`.

A l'initialisation, l'attitude de l'agent est représentée comme suit :

- `info1=forager.hasBelief(red_base, <(position, (-8, 13))>)`
- `info2=forager.hasBelief(blue_base, <(position, (8, 13))>)`
- `foragerMemory.memorize(info1, info2)` // ajout de `info1` et `info2` dans sa mémoire.

Au cours de son activité, l'agent va à chaque fois mémoriser la dernière position du palet. A ce moment précis, il effectuera par exemple la procédure suivante :

- `info_puck=forager.hasInfo(last_puck_position, (10, 10))`
- `foragerMemory.memorize(info_puck)`

Un problème se pose : la classe abstraite `Memory` ne peut être déclarée dans `Animat-Controller` puisque ce dernier est commun à tous les agents du système alors que les spécifications ci-dessus ne concernent que les agents cognitifs. Une autre classe dérivant de `AnimatController`, nommée `HumanoidController` est donc créée pour intégrer ces concepts[9]. La notion de temps n'est aussi gérée que dans cette dernière classe.

---

[8] Ils peuvent certes aussi être laissés tels quels mais notre but est de mettre en évidence le fonctionnement cognitif de l'agent.

[9] Le terme " humanoïde " est défini par le dictionnaire Robert (2001) comme un être vivant ou robot d'apparence humaine. Dans le cadre de la Vie Artificielle, la création d'humanoïde a pour but d'obtenir des robots qui intègrent des capacités propres à l'être humain telles que la marche bipède, la salutation,

En somme, la classe `ForagerController` dérivera de `AnimatController` pour le scénario (b) mais de `HumanoidController` pour le scénario (c).

La figure VI.5 résume toute cette nouvelle classification.

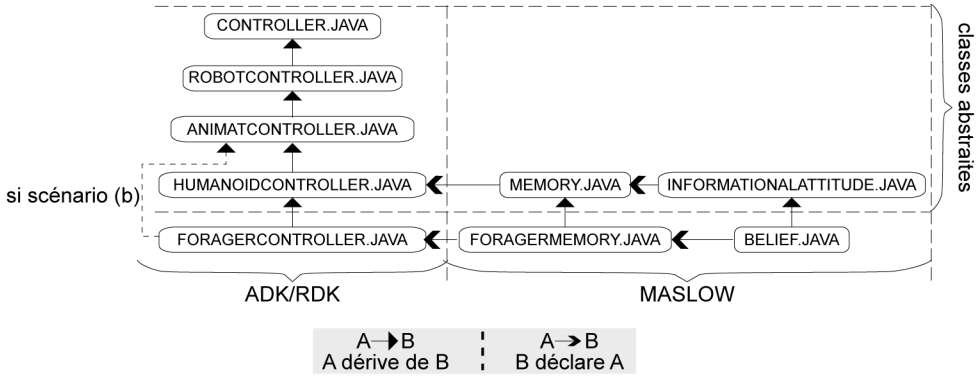


Figure VI.5 : Hiérarchie des classes dans le cadre du scénario (c)

Pour en revenir à la partie motivationnelle relative aux PN, notons que dans le présent scénario, nous avons supposé que les robots simulent des travailleurs humains qui transportent des objets, comme les dockers. Leur besoin de collecter correspond à celui de l'obtention de leur salaire (HN) qui leur permettra de satisfaire leur faim (le besoin LN de base).

Scénario (d) : des paysans intégrant Maslow

Dans ce scénario, on crée le pendant de la classe `ForagerController` précédente, à savoir la classe `PeasantController`. Il existe également la classe `PeasantMemory` pour

---

etc. Par symétrie au terme " animat " pour l'animal, nous avons choisi " humanoïde " dans le cadre humain.

gérer la mémoire d'un paysan. Cette dernière contient les positions du portail, de la maison et de la rizière, ainsi que le temps d'exécution des actions programmées.

- `info1=peasant.hasBelief(house, <(position, (-10, 9))>)` // maison
- `info2=peasant.hasBelief(ricefield, <(position, (-8, 10))>)` // rizière
- `info2=peasant.hasBelief(portal, <(position, (15, 17))>)` // portail
- `peasantMemory.memorize(info1, info2, info3)`

Quant au temps, il est initialisé avec la mémoire. Par exemple, l'initialisation de la date de début du travail (4h25 du matin) s'effectue comme suit :

1. `morning=forager.hasBelief(cultivateRice, <start_time, (4, 25, 00, "début travail matin")>)` // début du travail à 4h25 du matin.
2. `peasantMemory.memorize(morning)`, etc.

VI. 3. 2 Initialisation des applications : introduction

Etant donné que dans le scénario (a), l'initialisation s'effectue directement lors de la programmation des classes du contrôleur, la présente section décrit uniquement comment le système est initialisé dans les scénarios intégrant MASLOW.

Rappel des étapes

L'initialisation respecte la méthodologie présentée dans la section V.2.3 et que nous rappelons brièvement ici :

- recensement et initialisation des AN à satisfaire.
- recensement des actions
- recensement et initialisation des préconditions
- association des actions aux AN et aux préconditions
- connexion des actions entre elles

Quelques précisions

En vue d'une meilleure compréhension de l'expérimentation, voici les informations de base à connaître :

- dans toute l'expérimentation, nous prenons : `marge_ratio` =0.1 (cf. Equation (V.10), page 113 pour cette notion).
- l'initialisation de la variable `ccpos` sera donnée au fur et à mesure pour les BFI et les BFP. En revanche pour les BFB, la valeur initiale de `ccPos` dépend de la véracité ou non de la proposition introduite lors de la description de son état. Cela peut dépendre de beaucoup de paramètres dont le principal est celui de la position initiale des robots dans l'environnement (loin d'un obstacle ? à proximité d'un palet ?).
- sur le plan particulier de l'évitement des pierres, celui-ci a été modélisé en tant que BFB c'est-à-dire que soit l'agent est influencé par un obstacle, soit il ne l'est pas. A priori, le modéliser en BFI devrait aussi être faisable (ex : besoin progressif d'éviter un obstacle vers lequel l'agent se dirige). Cependant, nous n'avons pas encore traité la formule qui permet d'évaluer `ccpos` lorsqu'il existe plusieurs obstacles à la fois. S'il n'y avait qu'une seule pierre, cette formule serait trouvée : le calcul de la distance par rapport à cette pierre. Nous restons donc en BFB.
- autre rappel : nous utilisons ici des notations issues du langage Java. Cela s'applique donc également au cas des opérateurs booléens comme " `&&` " (et) " || ", (ou) et " `!` " (non), qui sont utilisés lors de la description des formes booléennes dans les PN.
- les notations comme `red_puck_visible` (signifiant *palet rouge visible*) parlent d'elles-mêmes.
- si un paramètre ne dispose pas de valeur, il est représenté par le mot clé `n/a` (" not available ").

Syntaxe des besoins et des actions

Pour différencier les préconditions (`precond`) des besoins à satisfaire (`pn_satisf`), nous adoptons la convention suivante :

1. les AN à satisfaire sont de la forme " satisfyXXX " où `XXX` est le libellé du besoin concerné ;
2. les préconditions sont de la forme " preCondYYY "où `YYY` est le nom de l'action dont ils sont les préconditions.

Cette convention sera utilisée pour chaque application afin de simplifier la représentation schématique, aussi bien des besoins sur Π que des actions sur Ω. Pour ce dernier, un nœud sera écrit sous la forme `XXX/YYY`. Ce qui signifie automatiquement que l'action du nœud possède `satisfyXXX` comme besoin à satisfaire et `preCondYYY` comme précondition[10].

Passons à présent à l'initialisation de chaque application.

VI. 3. 3 Initialisation du système pour les fourrageurs

Dans ce qui suit, la figure VI.6 illustre l'initialisation de Π. Les chiffres entre parenthèses près des AN indique le niveau de décomposition (`nd`) des besoins au niveau de Ω (c'est-à-dire en tant que `pn_satisf`).

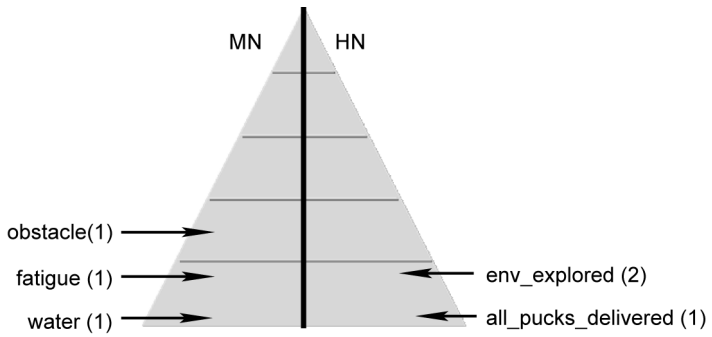


Figure VI.6 : Vue générale de Π avec les fourrageurs

La figure VI.7 montre quant à elle l'initialisation de Ω dans le cas des fourrageurs.

[10] En règle générale (mais pas toujours), nous avons fait en sorte que YYY soit la forme active (grammaticalement parlant) de XXX et XXX soit la forme passive de YYY.

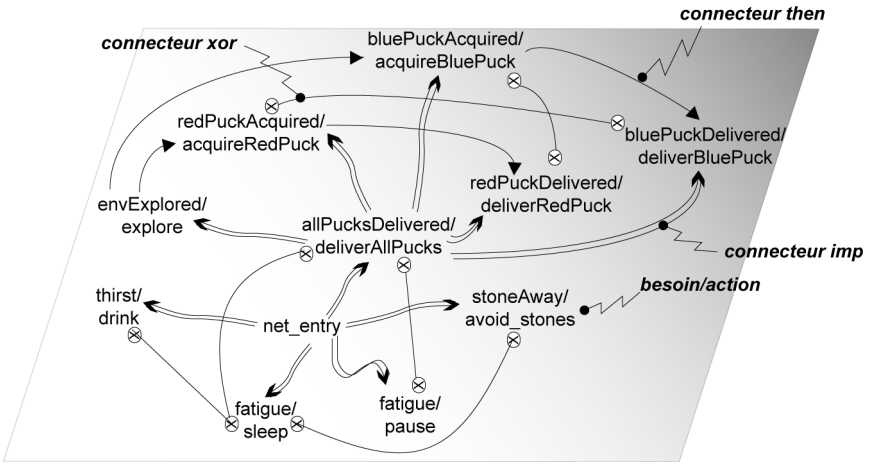


Figure VI.7 : Vue générale de Ω avec le cas des fourrageurs

Avant de présenter le recensement des besoins et des actions, voici quelques initialisations relatives à l'évolution des besoins physiologiques :

- `decWaterLevel` = 0.002 // diminution du niveau d'eau à chaque mouvement
- `incWaterLevel` = 0.2 // augmentation du niveau d'eau à chaque fois que l'agent boit
- `decFatigueLevel` = 0.2 // diminution de la fatigue à chaque fois que l'agent dort ou se repose[(11)]
- `incFatigueLevel` = 0.002 // augmentation de la fatigue à chaque mouvement

Recensement et initialisation des AN à satisfaire

Recensement MN

satisfyThirst, satisfyFatigue, satisfyStoneAway // les besoins MN finaux qui sont principalement des besoins physiologiques

Recensement HN

[11] Il est évidemment possible également de donner deux valeurs différentes de diminution pour les actions *sleep* et *pause*. C'est uniquement ici un choix de simplification dans la mesure où l'algorithme du NIM ne tient pas compte de ces valeurs. Le NIM ne fait que lancer les actions.

satisfyAllPucksDelivered, // les besoins HN finaux
satisfyEnvExplored, satisfyRedPuckAcquired, satisfyBluePuckAcquired, satisfyRedPuckDelivered, satisfyBluePuckDelivered // les besoins HN intermédiaires

Initialisation MN

state_thirst = (interval/INS_SUF, {[0, 5<, 8>, 10]}, {6})
satisfyThirst = (THIRST, ListLN.need_water, type_MN, state_ thirst) // `ccPos` initialisé à **4**

state_fatigue_convey = (interval/SUF_EXC, {[0, 2.5>, >3.5, 5]}, {2.8})
satisfyFatigue = (FATIGUE_CONVEY, ListLN.need_rest, type_MN, state_fatigue_convey) // `ccPos` initialisé à **1**

state_stone_away = (boolean, {close_to_stone})
satisfyStoneAway = (STONE_AWAY, ListLN.need_obstacle_away, type_MN, state_stone_away)

Initialisation HN

state_all_pucks_delivered = (interval/INS_SUF, {[0, 5<, 8>, 10]}, {7.5})
satisfyAllPucksDelivered = (ALL_PUCKS_DELIVERED, ListLN.need_food, type_HN, state_all_pucks_delivered) // `ccpos` initialisé à **0** i.e aucun palet transporté

state_explore = (boolean, {red_puck_visible || red_puck_in_gripper || blue_puck_visible || blue_puck_in_gripper })
satisfyEnvExplored = (EXPLORE, ListLN.need_food, type_HN, state_explore)

state_acq_red = (boolean, {red_puck_in_gripper || !red_puck_visible})
satisfyRedPuckAcquired = (ACQUIRE_RED, ListLN.need_food, type_HN, state_acq_red)

state_acq_blue = (boolean, {blue_puck_in_gripper || !blue_puck_visible})
satisfyBluePuckAcquired = (ACQUIRE_BLUE, ListLN.need_food, type_HN, state_acq_blue)

state_deliv_red = (boolean, {at_red_base && !red_puck_in_gripper})

satisfyRedPuckDelivered = (DELIVER_RED, ListLN.need_food, type_HN, state_deliv_red)

state_deliv_blue = (boolean, {at_blue_base && !blue_puck_in_gripper})
satisfyBluePuckDelivered = (DELIVER_BLUE, ListLN.need_food, type_HN, state_deliv_blue)

Recensement des actions

Les actions possibles pour satisfaire les besoins finaux

drink, sleep, pause, avoidStone, deliverAllPucks

Les actions possibles pour satisfaire les besoins intermédiaires

explore, acquireRedPuck, acquireBluePuck, deliverRedPuck, deliverBluePuck

Recensement et initialisation des préconditions

Recensement

preCondExplore, preCondAcquireRedPuck, preCondAcquireBluePuck, preCondDeliverRedPuck, preCondDeliverBluePuck

Initialisation

// dans l'initialisation, il arrive que des connecteurs `then` *se créent automatiquement. Rappelons-nous le mécanisme mentionné dans l'Equation (V.5), page 108.*

state_pr_explore = (boolean, { !puck_visible && !puck_in_gripper})
preCondExplore = (EXPLORE, ListLN.need_food, type_HN, state_pr_explore)

state_pr_acq_red = (boolean, {red_puck_visible && !red_puck_in_gripper})
preCondAcquireRedPuck = (ACQUIRE_RED, ListLN.need_food, type_HN, state_pr_acq_red)
// ci-dessus, création automatique du connecteur `then` entre `explore` et `acquireRedPuck` car satisfyEnvExplored.isUnSatisfied()⇒ preCondAcquireRedPuck.isUnSatisfied()

state_pr_acq_blue = (boolean, {blue_puck_visible && !blue_puck_in_gripper})
preCondAcquireBluePuck = (ACQUIRE_BLUE, ListLN.need_food, type_HN, state_pr_acq_blue)
// ci-dessus, création automatique du connecteur `then` entre `explore` et `acquireBluePuck` car satisfyEnvExplored.isUnSatisfied()⇒ preCondAcquireBluePuck.isUnSatisfied()

state_pr_deliv_red = (boolean, {red_puck_in_gripper})
preCondDeliverRedPuck = (DELIVER_RED, ListLN.need_food, type_HN, state_pr_deliv_red)
// ci-dessus, création automatique du connecteur `then` entre `acquireRedPuck` et `deliverRedPuck` car satisfyRedPuckAcquired.isUnSatisfied()⇒preCondDeliverRedPuck.isUnSatisfied()

state_pr_deliv_blue = (boolean, {blue_puck_in_gripper})
preCondDeliverBluePuck = (DELIVER_BLUE, ListLN.need_food, type_HN, state_pr_deliv_blue)
// ci-dessus, création automatique du connecteur `then` entre `acquireBluePuck` et `deliverBluePuck` car satisfyBluePuckAcquired.isUnSatisfied()⇒preCondDeliverBluePuck.isUnSatisfied()

Association des actions aux besoins et aux préconditions

drink = ("drink", preCondThirst, satisfyThirst}[12]
satisfyThirst.defaultAction=drink

sleep = ("sleep", n/a, satisfyFatigue)
pause = ("pause", n/a, satisfyFatigue)
satisfyFatigue.defaultAction=pause // par défaut c'est `pause` ...

[12] Dans cette initialisation, la primitive *drink* est l'action (le contexte) et le paramètre " drink " dans cette action fait implicitement référence au nom de la fonction à exécuter dont l'implémentation réelle, pour le cas des fourrageurs, se trouve dans *ForagerController*. Cette fonction, rappelons-le, peut être ainsi utilisée par plusieurs primitives.

satisfyFatigue.action[insufficient]= sleep // ...sauf pour l'état insuffisant où il faut exécuter `sleep`

avoidStone = ("avoid_stones", n/a, satisfyStoneAway)
satisfyStoneAway.defaultAction=avoidStone

deliverAllPucks = ("deliver all pucks", n/a, satisfyAllPucksDelivered)
satisfyAllPucksDelivered.defaultAction=deliverAllPucks

explore = ("explore", preCondExplore, satisfyEnvExplored)
satisfyEnvExplored.defaultAction=explore

acquireRedPuck = ("acquireRedPuck", preCondAcquireRedPuck, satisfyRedPuckAcquired)
satisfyRedPuckAcquired.defaultAction=acquireRedPuck

acquireBluePuck = ("acquireBluePuck", preCondAcquireBluePuck, satisfyBluePuckAcquired)
satisfyBluePuckAcquired.defaultAction=acquireBluePuck

deliverRedPuck = ("deliverRedPuck", preCondDeliverRedPuck, satisfyRedPuckDelivered)
satisfyRedPuckDelivered.defaultAction=deliverRedPuck

deliverBluePuck = ("deliverBluePuck", preCondDeliverBluePuck, satisfyBluePuckDelivered)
satisfyBluePuckDelivered.defaultAction=deliverBluePuck

Connexion des actions entre elles

imp(deliverAllPucks, explore)
imp(deliverAllPucks, acquireBluePuck)
imp(deliverAllPucks, acquireRedPuck)
imp(deliverAllPucks, deliverBluePuck)

imp(deliverAllPucks, deliverRedPuck)

xor(acquireRedPuck, deliverBluePuck)
xor(deliverRedPuck, acquireBluePuck)

//quand on dort, on ne fait que dormir xor(sleep, deliverAllPucks)
xor(sleep, drink)
xor(sleep, avoidStone)

// quand on fait la pause, on ne travaille pas

xor(pause, deliverAllPucks)

Initialisation du temps

Dans le cadre des fourrageurs, nous n'avons émis aucune notion du temps.

VI. 3. 4 Initialisation du système pour les paysans

Comme dans le cas des fourrageurs, la figure VI.8 illustre l'initialisation de Π pour le cas des paysans.

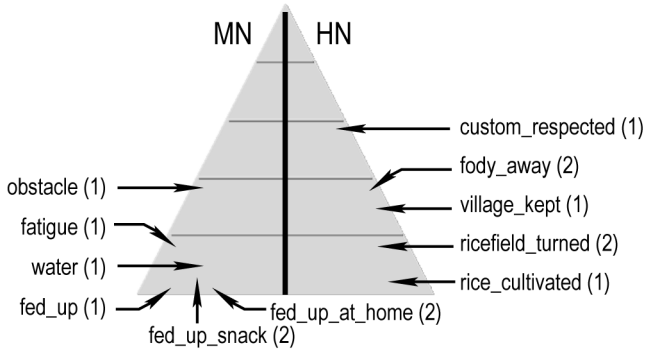


Figure VI.8 : Vue générale de Π avec le cas de paysans

Enfin, la figure VI.9 montre une initialisation de Ω avec les paysans.

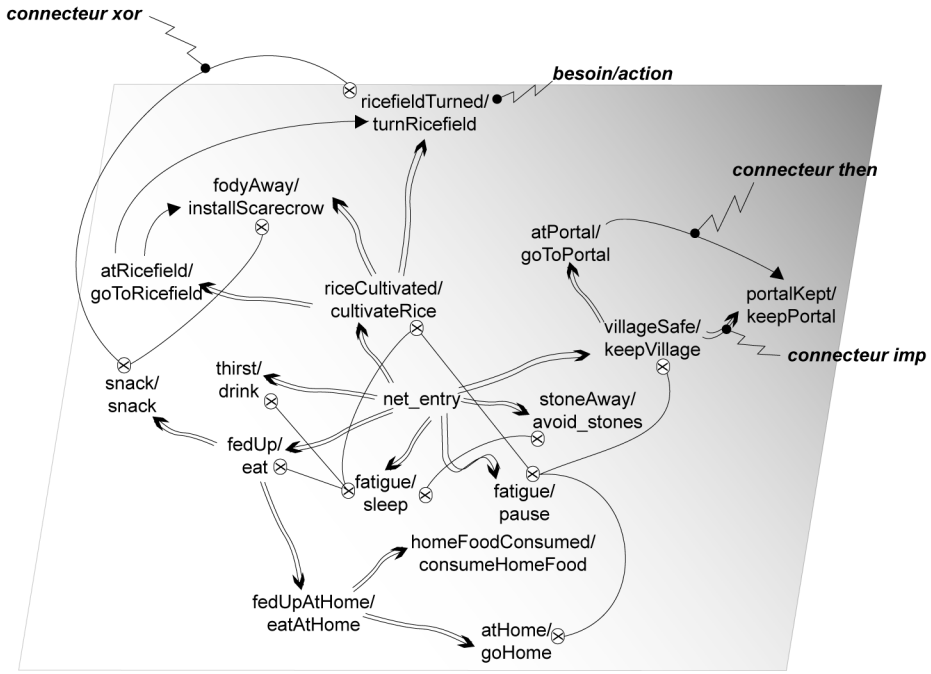


Figure VI.9 : Vue générale de Ω avec le cas des paysans

Avant de procéder au recensement des besoins et des actions, voici quelques initialisations relatives à l'évolution des besoins physiologiques :

- `decWaterLevel` = 0.002 // diminution du niveau d'eau à chaque mouvement
- `incWaterLevel` = 0.2 // augmentation du niveau d'eau à chaque fois que l'agent boit
- `decFatigueLevel` = 0.2 // diminution de la fatigue à chaque fois que l'agent dort ou se repose[13]
- `incFatigueLevel` = 0.002 // augmentation de la fatigue à chaque mouvement du robot
- `incSnackLevel` = 0.1 // augmentation du rassasiement à chaque goûter
- `decHungerLevel` = 0.03 // augmentation de la faim à chaque mouvement du robot
- `incHungerLevel` = 0.2 // augmentation du rassasiement à chaque déjeuner ou dîner

[13] Il est évidemment aussi possible de donner deux valeurs différentes de diminution pour les actions *sleep* et *pause*. C'est uniquement ici un choix de simplification dans la mesure où l'algorithme du NIM ne tient pas compte de ces valeurs. Le NIM ne fait que lancer les actions.

Recensement et initialisation des AN à satisfaire

Recensement MN

satisfyThirst, satisfyFedUp, satisfyFatigue, satisfyStoneAway // les besoins MN finaux
satisfyFedUpAtHome, satisfySnack, satisfyAtHome, satisfyHomeFoodConsumed, // les besoins MN intermédiaires

Recensement HN

satisfyRiceCultivated, satisfyVillageSafe, satisfyCustomRespected, // les besoins HN finaux
satisfyRicefieldTurned, satisfyAtRiceField,
satisfyFodyAway,
satisfyAtPortal, satisfyPortalKept // les besoins HN intermédiaires

Initialisation MN

state_water = (interval/INS_SUF, {[0, 5<, 8<, 10]}, {6})
satisfyThirst = (WATER, ListLN.need_water, type_MN, state_water) // `ccpos` initialisé à **9**

state_eat = (interval/INS_EXC, {[1, 7>, 11.5<, 12.5>, 17.5>, 20[}, {11, 15})
satisfyFedUp = (FEED_UP, ListLN.need_food, type_MN, state_eat) // `ccpos` initialisé à **10.05**

satisfyFedUpAtHome = (FEED_UP_AT_HOME, ListLN.need_food, type_MN, state_eat)
satisfySnack = (SNACK, ListLN.need_food, type_MN, state_eat)
satisfyHomeFoodConsumed = (HOME_FOOD_CONSUMED, ListLN.need_food, type_MN, state_eat)

state_fatigue_convey = (interval/SUF_EXC, {[0, 2.5>, >3.5, >5]}, {2.8})
satisfyFatigue = (FATIGUE_AWAY, ListLN.need_rest, type_MN, state_fatigue_convey) // `ccpos` initialisé à **1**

state_at_home = (boolean, {at_home})
satisfyAtHome = (AT_HOME, ListLN.need_food, type_MN, state_at_home)

state_stone_away = (boolean, {close_to_stone})
satisfyStoneAway = (STONE_AWAY, ListLN.need_obstacle_away, type_MN, state_stone_away)

Initialisation HN

state_rice_cultivated = (boolean, {rice_cultivated})
satisfyRiceCultivated = (RICE_CULTIVATED, ListLN.need_food, type_HN, state_rice_cultivated)

state_village_safe = (boolean, {false})
satisfyVillageSafe = (VILLAGE_SAFE, ListLN.need_danger_away, type_HN, state_village_safe) //cf. simulation du cadre social (rappel Section VI.2.2)

state_custom_respected = (boolean, {false})
satisfyCustomRespected = (CUSTOM_RESPECTED, ListLN.need_society_acceptation, type_HN, state_custom_respected) //cf. simulation du cadre social (rappel Section VI.2.2)

state_fody_away = (point/INS_SUF, {0, 1, 2}, {1})
satisfyFodyAway = (FODY_AWAY, ListLN.need_danger_away, type_HN, state_fody_away) // `ccpos` initialisé à **0** i.e. aucun épouvantail posé au départ

state_at_ricefield = (boolean, {at_ricefield})
satisfyAtRiceField = (AT_RICEFIELD, satisfyRiceCultivated, nsdb)

state_ricefield_turned = (interval/INS_SUF, {[0, 100>, 100<, 100[}, {100})
satisfyRicefieldTurned = (RICEFIELD_TURNED, ListLN.need_food, type_HN, state_ricefield_turned)

state_at_portal = (boolean, {at_portal})
satisfyAtPortal = (AT_PORTAL, ListLN.need_danger_away, type_HN, state_at_portal)

state_portal_kept = (boolean, {portal_kept})

satisfyPortalKept = (PORTAL_KEPT, ListLN.need_danger_away, type_HN, state_portal_kept)

Recensement des actions

Les actions possibles pour satisfaire les besoins finaux

drink, sleep, pause, eat, avoidStone, cultivateRice, keepVillage, respectCustom,

Les actions possibles pour satisfaire les besoins intermédiaires

digest, snack, eatAtHome, goHome, consumeHomeFood // actions relatives à la faim
goToRicefield, turnRicefield, installScareCrow // actions relatives au travail à la rizière

goToPortal, keepPortal // actions relatives à la sécurité du village

Recensement et initialisation des préconditions

Recensement

preCondAtPortal, preCondKeepPortal, preCondAtRicefield preCondTurnRicefield preCondInstallScarecrow, preCondAtHome, preCondConsumeHomeFood,

Initialisation

// dans l'initialisation, il arrive que des connecteurs `then` *se créent automatiquement. Rappelons-nous le mécanisme mentionné dans l'Equation (V.5), page 108.*

state_pr_at_ricefield = (boolean, {!at_ricefield})
preCondAtRicefield = (AT_RICEFIELD, type_HN, state_pr_at_ricefield)

state_pr_ricefield_turned = (boolean, {at_ricefield})
preCondTurnRicefield = (TURN_RICEFIELD, ListLN.need_food, type_HN, state_pr_ricefield_turned)
// ci-dessus, création automatique du connecteur `then` entre `goToRicefield` et `turnRicefield` car satisfyAtRiceField.isUnSatisfied()⇒preCondTurnRicefield.isUnSatisfied()

state_pr_ricefield_turned = (boolean, {at_ricefield})
preCondInstallScarecrow = (INSTALL_SCARECROW, ListLN.need_food, type_HN, state_pr_ricefield_turned)
// ci-dessus, création automatique du connecteur `then` entre `goToRicefield` et `installScareCrow` car satisfyAtRiceField.isUnSatisfied()⇒preCondInstallScarecrow.isUnSatisfied()

state_pr_at_portal = (boolean, { !at_portal})
preCondAtPortal = (AT_PORTAL, type_HN, state_pr_at_portal)

state_pr_keep_portal = (boolean, {at_portal})
preCondKeepPortal = (KEEP_PORTAL, ListLN. need_danger_away, type_HN, state_pr_keep_portal)
// ci-dessus, création automatique du connecteur `then` entre `goToRicefield` et `installScareCrow` car satisfyAtPortal.isUnSatisfied()⇒preCondKeepPortal.isUnSatisfied()

state_pr_at_home = (boolean, { !at_home})
preCondAtHome = (AT_HOME, type_MN, state_pr_at_home)

state_pr_consume_homefood = (boolean, {at_home})
preCondConsumeHomeFood = (CONSUME_HF, ListLN. need_food, type_HN, state_pr_consume_homefood)
// ci-dessus, création automatique du connecteur `then` entre `goHome` et `consumeHomeFood` car satisfyAtHome.isUnSatisfied()⇒preCondConsumeHomeFood.isUnSatisfied()

Associer les actions aux besoins à satisfaire et aux préconditions

drink = ("drink", n/a, satisfyThirst)
satisfyThirst.defaultAction=drink

eat = ("eat", n/a, satisfyFedUp)
satisfyFedUp.defaultAction=eat
satisfyFedUp.action[INSUFFICIENT]=snack // on n'attend pas l'heure du déjeuner si on a très faim

sleep = (”sleep”, n/a, satisfyFatigue)
pause = (”pause”, n/a, satisfyFatigue)
satisfyFatigue.defaultAction=pause
satisfyFatigue.action[INSUFFICIENT]=sleep

avoidStone = (”avoid_stones”, n/a, satisfyStoneAway)
satisfyStoneAway.defaultAction = avoidStone

cultivateRice = (”cultivate ricefield”, n/a, satisfyRiceCultivated)
satisfyRiceCultivated.defaultAction=cultivateRice

keepVillage = (”keep village”, n/a, satisfyVillageSafe)
satisfyVillageSafe.defaultAction=keepVillage

respectCustom = (”respectCustom”, n/a, satisfyCustomRespected)
satisfyCustomRespected.defaultAction=respectCustom

eatAtHome = (”eat at home”, n/a, satisfyFedUpAtHome)
satisfyFedUpAtHome.defaultAction=eatAtHome

snack = (”snack”, n/a, satisfySnack)
digest = (”digest”, n/a, satisfySnack)
satisfySnack.defaultAction=snack
satisfySnack.setActionEval(LIMIT_H, digest)
satisfySnack.setActionEval(EXCESSIVE, digest)

goToPortal = (”goToPortal”, preCondAtPortal, satisfyAtPortal)
satisfyAtPortal.defaultAction=goToPortal

keepPortal = (”keepPortal”, preCondKeepPortal, satisfyPortalKept)
satisfyPortalKept.defaultAction=keepPortal

goToRicefield = (”goToRicefield”, preCondAtRicefield, satisfyAtRiceField)
satisfyAtRiceField.defaultAction = goToRicefield

turnRicefield = ("turnRicefield", preCondTurnRicefield, satisfyRicefieldTurned)
satisfyRicefieldTurned.defaultAction = turnRicefield

installScareCrow = ("installScareCrow", preCondInstallScarecrow, satisfyFodyAway)
satisfyFodyAway.defaultAction = installScareCrow

goHome = ("goHome", preCondAtHome, satisfyAtHome)
satisfyAtHome.defaultAction = goHome

consumeHomeFood = ("consumeHomeFood", preCondConsumeHomeFood, satisfyHomeFoodConsumed)
satisfyHomeFoodConsumed.defaultAction = consumeHomeFood

Connexion des actions entre elles

imp(keepVillage, goToPortal)
imp(keepVillage, keepPortal)

imp(eat, snack)
imp(eat, eatAtHome)

imp(eatAtHome, goHome)
imp(eatAtHome, consumeHomeFood)

xor(turnRicefield, snack)

xor(installScareCrow, snack)

// quand on dort, on ne fait que dormir

xor(sleep, cultivateRice)
xor(sleep, drink)
xor(sleep, eat)
xor(sleep, avoidStone)

// quand on fait la pause, on ne travaille pas

xor(pause, cultivateRice)

xor(pause, goHome)

xor(pause, keepVillage)

Initialisation du temps

Tranche de temps du travail à la rizière

start_time(cultivateRice, {4, 25, "début travail matin"})
end_time(cultivateRice, {11, 50, "fin travail matin"})

start_time(cultivateRice, {14, 0, "début travail après-midi"})
end_time(cultivateRice, {18, 30, "fin travail après-midi"})

Tranche de temps pour manger

start_time(eatAtHome, {11, 50, "début déjeuner"})
end_time(eatAtHome, {13, 45, "fin déjeuner"})

start_time(eatAtHome, {18, 40, "début diner"})
end_time(eatAtHome, {22, 45, "fin diner"})

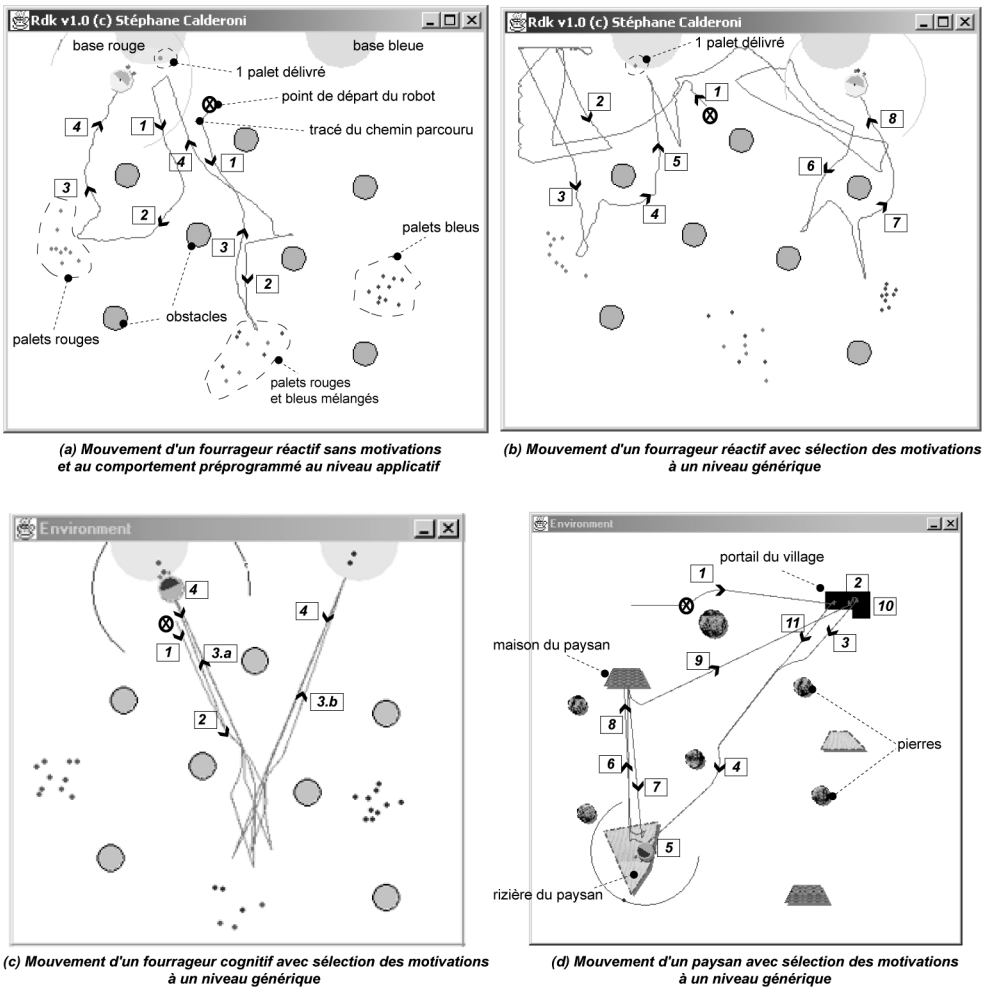
Tranche de temps pour garder le village

start_time(keepVillage, {22, 50, "début garder le village"})
end_time(keepVillage, {1, 30, "fin garder le village"})

VI. 4 Résultats et analyses

La figure VI.10 montre le résultat de tous les scénarios expérimentés. La trace du mouvement du robot dans chaque scénario permet de décrire son comportement. Chaque numéro figurant sur le chemin correspond à des actions que nous avons prélevées car considérées comme les plus significatives pour notre analyse. Les sections VI.4.1 et VI.4.2 expliquent respectivement le détail de ces numéros pour le cas des fourrageurs puis des paysans.

A noter que pour des raisons de clarté de présentation au niveau des figures (et notamment au suivi des traces en question), nous ne présentons ici que le résultat du comportement d'un seul robot pour chaque scénario, en sachant qu'à la base, tous les autres robots suivent le même principe. La seule situation introuvable dans le cas d'un unique robot, c'est l'évitement entre deux robots, ce qui, pour notre travail, peut être comblé par l'évitement d'obstacles dans la mesure où nous n'étudions pas la manière d'éviter mais la raison de l'évitement.



(a) Mouvement d'un fourrageur réactif sans motivations et au comportement préprogrammé au niveau applicatif

(b) Mouvement d'un fourrageur réactif avec sélection des motivations à un niveau générique

(c) Mouvement d'un fourrageur cognitif avec sélection des motivations à un niveau générique

(d) Mouvement d'un paysan avec sélection des motivations à un niveau générique

Figure VI.10 : Schéma comparatif du mouvement de l'agent dans les 4 scénarios étudiés

VI. 4. 1 Comportement des fourrageurs

Le scénario (a) n'est ici décrit que brièvement vu son aspect préprogrammé. Les autres scénarios sont présentés plus en détail.

Scénario (a)

`1 explore`

Dans un scénario préprogrammé de fourragement, c'est logiquement cette action qui est exécutée en premier lieu car c'est l'étape initiale du fourragement. C'est aussi automatiquement cette action qui est aussi exécutée après chaque dépôt de palet. Comme ici, le robot, n'a aucune représentation de l'environnement, il va effectuer une exploration aléatoire.

Enfin, comme il n'y a pas d'obstacles, `explore` est la seule action exécutée.

`2 acquireRedPuck`

C'est logiquement l'action suivante exécutée, suite à la perception d'un palet rouge. Comme il n'y a pas non plus d'obstacles, c'est la seule action à être exécutée.

`3 deliverRedPuck (1)`

Ici, l'agent délivre le palet acquis tout en évitant un obstacle

`4 deliverRedPuck (2)`

L'obstacle étant passé, l'agent continue sa livraison

Scénario (b) : application de algoNIM(AC, nd)

A partir de ce scénario, nous expliquerons le fonctionnement de l'algorithme du NIM par rapport aux applications proposées.

Comme à chaque début du processus dans l'algorithme du NIM, `finalPrim`={}. La variable `list¡AN` signifie uniquement `listAN` avec `nd`=i. L'analogie demeure pour `lAct`.

`1 drink & explore`

0- Génération de la liste initiale pour nd=1 avec AC=" net_entry ".

⇒list$_1$AN={satisfyThirst, satisfyFatigue, satisfyStoneAway, satisfyAllPucksDelivered }

1- rejet des AN / AN.isFullySatisfied()

satisfyThirst =4 (→insuffisant=insatisfait)
satisfyFatigue =1 (→suffisant)
satisfyStoneAway=1 (→suffisant) // pas d'obstacle⇒rejeté[14]
satisfyAllPucksDelivered=0 (→insuffisant)

⇒list$_1$AN={satisfyThirst, satisfyFatigue, satisfyAllPucksDelivered}

2- tri de la liste

Le résultat du tri est :

- satisfyThirst> satisfyFatigue // les deux besoins étant issus du même LN et ayant le même *rang*, le départage passe par l'état : insuffisant>suffisant (rappel Equation (V.7), page 112).
- satisfyThirst >satisfyAllPucksDelivered // départagé par la règle de l'importance[15].
- satisfyAllPucksDelivered> satisfyFatigue // départagé par l'état : insuffisant>suffisant. Même si MN>HN, la règle de l'importance n'a pas pu être appliquée car satisfyFatigue n'est pas à l'état insuffisant.

⇒list$_1$AN={satisfyThirst, satisfyAllPucksDelivered, satisfyFatigue}

3- Rejet des AN / not AN.isInSatisfaction ()

→rejet de satisfyFatigue car !satisfyFatigue.isInSatisfaction(). Rappelons qu'à l'initialisation, les AN sont tous non en cours de satisfaction (cf. Section V.3.2). Par contre, satisfyThirst est maintenu car le fait qu'il soit *insatisfait* le classe automatiquement comme parmi les besoins en cours de satisfaction.

---

[14] Rappelons qu'à propos des PN de la forme BFP et BFB, *PN.isSufficient()⇒ PN.isFullySatisfied()*

[15] Rappel de la règle : Soit *PN* un besoin considéré comme important résultant des critères type, niveau, catégorie ou rang, et *PN'* l'autre besoin en cours de comparaison avec PN, alors si PN est insatisfait, alors *PN>PN'*.

⇒list$_1$AN={satisfyThirst, satisfyAllPucksDelivered}

4- Génération de la liste des actions issue du AN.action[currentState] des besoins

⇒l$_1$Act={drink, deliverAllPucks}.

5-Application des préconditions (le temps n'existe pas encore ici)

→pour les deux, précondition=n/a ⇒précondition OK

⇒l$_1$Act={drink, deliverAllPucks}

6- Séparation PR/AC

Comme drink est une primitive, on a
finalPrim= {}+{drink}+{algoNIM(deliverAllPucks, 2)}

→exécution de algoNIM(deliverAllPucks, 2)

0- Génération de la liste initiale

⇒list$_2$AN={satisfyRedPuckAcquired, satisfyEnvExplored, satisfyBluePuckAcquired satisfyRedPuckDelivered, satisfyBluePuckDelivered}

1- rejet des AN / AN.isFullySatisfied()

satisfyEnvExplored =0 (→insuffisant) // ni palets visibles ni palets dans la pince
satisfyRedPuckAcquired =1 (→suffisant) // pas de palet rouge visible donc pas de besoin de prise d'un palet rouge insatisfait (c'est-à-dire que le besoin n'est insatisfait que si l'agent voit un palet rouge et qu'il ne le prend pas)
satisfyBluePuckAcquired =1 (→suffisant) // argument symétrique avec le cas du palet rouge précédent
satisfyRedPuckDelivered =1 (→suffisant) // l'agent n'est pas à la base et il n'a pas de palets dans sa pince. Ce besoin n'aurait été insatisfait que s'il avait un palet et qu'il n'était pas à la base
satisfyBluePuckDelivered =1 (→suffisant) // argument symétrique avec le cas du palet rouge précédent

⇒list$_2$AN={satisfyEnvExplored},

2- tri de la liste

Pas de tri puisqu'il n'y a qu'un seul AN

3- Rejet des AN / not AN.isInSatisfaction ()

Il n'y a aucun rejet puisque `satisfyEnvExplored` est à l'état `insuffisant` ⇒non satisfait ⇒ en cours de satisfaction.

⇒$list_2$AN={`satisfyEnvExplored`}

4- Génération des actions, à partir du AN.action[currentState]

l_2Act={`explore`}

5-Application des préconditions (le temps n'existe pas encore ici)

`preCondExplore`=1 (→`suffisant`)// pas de palets en vue et aucun palet dans la pince

⇒l_2Act={`explore` }

6- Séparation PR/AC

Nous avons : `finalPrim`= {`drink`}+{`explore`}

Comme il n'y a plus d'AC à traiter, le NIM passe à la fin d'étapes.

Application des conflits sur finalPrim

Pas de conflit entre `drink` et `explore`

⇒**finalPrim={drink, explore}**

2 explore & drink

Même processus que (1) sauf que suite à l'action `drink` précédente, `satisfyThirst` =6 (→`limit_l`) alors que `satisfyAllPucksDelivered`=0 (→`insuffisant`). Les rangs sont donc inversés. Cela signifie que les actions sélectionnées demeurent les mêmes. C'est le degré de motivations de l'agent à les exécuter qui est différent. A noter qu'entre ces deux actions, `satisfyFatigue` n'a pas encore changé d'état.

3 acquireRedPuck

Par rapport aux processus précédents, `satisfyThirst` a été entre temps pleinement satisfait (suite au `drink` exécutée continuellement[16]) et `explore` a fait place à `acquireRedPuck` au moment du test des préconditions (rappelons que la précondition de `acquireRedPuck`, c'est `satisfyEnvExplored`).

4 `avoid & deliverRedPuck`

Tout d'abord, contrairement à (3), `acquireRedPuck` a fait place à `deliverRedPuck` au moment du test des préconditions dans `algoNIM(...)` (rappelons que la précondition de `deliverRedPuck`, c'est `satisfyRedPuckAcquired`).

De plus, le besoin `satisfyStoneAway` est pris en compte : `satisfyStoneAway=0` (→`insuffisant`) dû à l'existence d'obstacles.

Ici, même si `satisfyThirst` n'est plus pleinement satisfait (tout comme `satisfyFatigue` d'ailleurs), il n'est pas encore en cours de satisfaction. Ces deux besoins ne le seront qu'une fois insatisfaits.

5 `deliverRedPuck`

Même explication que (4) sauf qu'il n'y a plus d'obstacles (l'obstacle de (4) étant passé)

6 `pause & drink`

A ce moment précis :

- `satisfyThirst`=7 (→`limite_l`)
- `satisfyAllPucksDelivered` = 1 car un palet a déjà été délivré (cf. Figure VI.10-b, page 155), mais il est toujours à l'état `insuffisant`
- `satisfyFatigue` =3 (→`limite_h` d'où le choix de `pause` au lieu de `sleep`). En outre, `satisfyFatigue` est insatisfait car `minSat` =2.8.

[16] L'action *drink* a pu être exécutée continuellement même étant à l'état *suffisant* (et donc parmi les dernières de *finalPrim*) car elle n'a été en conflit avec aucune des autres actions sélectionnées. Elle n'est normalement en conflit qu'avec *sleep*. Or *sleep* n'est pas sélectionné puisque *satisfyFatigue* n'est pas à l'état *insuffisant*.

Le tri des besoins se fait alors comme suit :

- en vertu de la règle de l'importance :
  `satisfyFatigue` > `satisfyAllPucksDelivered`.
- comme `satisfyFatigue` et `satisfyThirst` appartiennent au même type, niveau, et catégorie, rang, et puis ici, " même " état également (on ne peut pas départager entre un `limit_l` et `limit_h`), il faut passer par le ratio (rappel Equation (V.8), page 113). Nous avons donc :
  ratio(`satisfyFatigue`)=|3-5|/|5-0|=0.4
  ratio(`satisfyThirst`)=|7-0|/|10-0|=0.7 et 0.4+marge_ratio≤0.7
  ⇒ en suivant l'Equation (V.10), page 113, nous avons : `satisfyFatigue`>`satisfyThirst`.

En outre, `satisfyAllPucksDelivered`>`satisfyThirst` car `satisfyThirst` étant encore satisfait, la règle de priorité ne s'applique pas bien que ce dernier soit à l'état `limite`.

Nous avons finalement :

→`listAN`={`satisfyFatigue, satisfyAllPucksDelivered, satisfyThirst`}

Comme nous avons : `xor(pause, deliverAllPucks)`, la liste finale d'actions est : `lAct`={`pause, drink`}.

`7 avoid & deliverBluePuck`

cf. (4) avec le cas d'un palet bleu

`8 deliverBluePuck`

cf. 3 avec le cas d'un palet bleu

Scénario (c)

Dans ce scénario, les actions prises sont :
1- `drink & explore` : même départ que le (1) du scénario (b) car même initialisation que lui
2- `avoid & explore & drink` : idem avec le (2) du scénario (b) avec un obstacle en plus

3.a : `deliverRedPuck`
3.b : `deliverBluePuck`
4- `explore` mais revient au dernier endroit où un palet a été trouvé

Nous pouvons remarquer que peu d'actions ont été relevées ici par rapport au scénario (b). La raison en est que d'une manière générale, le processus de sélection des besoins dans ce scénario cognitif ne diffère pas de celui expliqué dans le scénario (b) précédent (d'autant plus qu'ils sont initialisés avec les mêmes besoins). De ce fait, la démarche ayant abouti à ces actions ne sera pas détaillée ici. Par contre, nous allons expliquer la différence de trajectoire adoptée par le robot montrée dans la figure VI.10-c, page 155, par rapport aux deux figures précédentes (a) et (b).

En fait, ce comportement (c'est-à-dire le va-et-vient à un endroit donné au lieu d'une exploration aléatoire) est dû aux facteurs cognitifs du robot, à savoir sa faculté d'avoir une représentation partielle de son environnement, etf de mémoriser la zone où le dernier palet ramassé a été trouvé. Or, nous avons vu dans la figure VI.5, page 137, que toutes les classes implémentant les facteurs cognitifs de l'agent sont connectées à l'agent à partir de la classe `HumanoidController` alors que le NIM se situe dans la classe `AnimatController` qui n'est donc pas sensé connaître les fonctionnalités de ses classes dérivées, tout en y influençant ses propres caractéristiques en tant que classe mère.

En somme, la différence notoire entre (b) et (c) se situe plutôt au niveau comportemental à savoir :

- ramener un palet à un endroit précis en fonction de sa couleur (action `3.a` si rouge et action `3.b` si bleue) en s'aidant de ses connaissances de la location des bases,
- revenir au dernier endroit exploré (action `4`) s'il a été mémorisé qu'il y existe encore au moins un palet.

Les motivations sous-jacentes sont les mêmes. Plus concrètement, nous utilisons le même algorithme du NIM, implanté génériquement au sein de la classe `AnimatController`.

VI. 4. 2 Comportement des paysans

Pour rappel, la simulation débute à 00h :00. Dans tout ce qui suit, nous mentionnons l'heure à laquelle les actions relevées se sont produites.

Comme à chaque début du processus dans l'algorithme du NIM, `finalPrim={}`

`1 snack & goToPortal & respectCustom (00 :01)`

0- Génération de la liste initiale pour nd=1 avec AC=" net_entry ".

⇒`list`$_1$`AN={satisfyThirst, satisfyFedUp, satisfyFatigue, satisfyStoneAway, satisfyRiceCultivated, satisfyVillageSafe, satisfyCustomRespected}`

1- Rejet des AN / AN.isFullySatisfied()

`satisfyThirst` =9 (→`suffisant`)
`satisfyFatigue` =1 (→`suffisant`)
`satisfyFedUp`=10.05 (→`limit_l` mais insatisfait car `minSat=11`)
`satisfyStoneAway`=1 (→`suffisant`) // pas d'obstacle⇒rejeté
`satisfyRiceCultivated` =0 (→`insuffisant`) // le riz n'est pas cultivé
`satisfyVillageSafe` =0 (→`insuffisant`) // besoin simulé
`satisfyCustomRespected` =0 (→`insuffisant`) // besoin simulé

⇒`list`$_1$`AN={satisfyThirst, satisfyFatigue, satisfyFedUp, satisfyRiceCultivated, satisfyVillageSafe, satisfyCustomRespected}`

2- tri de la liste

Le résultat du tri est :

- `satisfyFatigue`>`satisfyThirst` car les deux besoins appartenant au même type, niveau, et catégorie, rang, et ici, même état également, il faut passer par le *ratio*. Nous avons donc :
  ratio(`satisfyFatigue`)=|1-5|/|5-0|=0.8
  ratio(`satisfyThirst`)=|9-0|/|10-0|=0.9 et 0.8+marge_ratio≤0.9
- `satisfyFedUp`> `satisfyFatigue` (car `limit_l`>`suffisant`).
- `satisfyFedUp`> `satisfyThirst` (car `limit_l`>`suffisant`).
- `satisfyRiceCultivated`>`satisfyVillageSafe`>`satisfyCustomRespected` (critère de Maslow sur les niveaux)
- `satisfyFedUp` > `satisfyRiceCultivated` // règle de l'importance

⇒list$_1$AN={satisfyFedUp, satisfyFatigue, satisfyThirst, satisfyRice-Cultivated, satisfyVillageSafe, satisfyCustomRespected}

3- Rejet des AN / not AN.isInSatisfaction ()

→rejet de satisfyFatigue, satisfyThirst // à l'initialisation, les AN sont non en cours de satisfaction (sauf par la suite ceux qui sont insatisfaits)

⇒list$_1$AN={satisfyFedUp, satisfyRiceCultivated, satisfyVillageSafe, satisfyCustomRespected}

4- Génération de la liste des actions issue du AN.action[currentState] des besoins

⇒l$_1$Act ={eat, cultivateRice, keepVillage, respectCustom}.

5-Application des préconditions

→préCondition==n/a ⇒précondition OK

⇒l$_1$Act ={eat, cultivateRice, keepVillage, respectCustom}.

5bis-Application du critère temps

Seule l'action cultivateRice est éliminée car elle est programmée pour commencer à 04 :25 du matin.

⇒l$_1$Act ={eat, keepVillage, respectCustom}.

6- Séparation PR/AC

finalPrim= {}+{algoNIM(eat, 2) }+{algoNIM(keepVillage, 2)}+ respectCus-tom

→exécution de algoNIM(eat, 2)

0- Génération de la liste initiale

⇒listAN={satisfySnack, satisfyFedUpAtHome}

Ces deux besoins évoluent exactement de la même manière que satisfyFedUp vu juste précédemment. Ils suivent donc aussi les mêmes démarches que satisfyFedUp dans l'algorithme précédent. Le seul critère qui sépare satisfySnack et satisfyFedUpAtHome,

c'est leur temps d'exécution. Ainsi à cette étape, `satisfyFedUpAtHome` sera éliminée car l'action `eatAtHome` associée ne peut s'exécuter qu'à 11h50.

⇒`listAN`={`satisfySnack`}

Cependant, un problème se pose lorsqu'il sera précisément 11h50 puisque `snack` et `eatAtHome` peuvent toutes les deux être exécutées. La seule différence est que `eatAtHome` est programmée[17] alors que `snack` ne l'est pas. Pour l'instant, nous avons adopté l'hypothèse que si tel était le cas, alors c'est l'action programmée qui est choisie[18]. Si les deux actions sont toutes programmées ou toutes non programmées, alors c'est là que le choix entre elles se fait au hasard.

En définitif, nous avons donc ici :
`finalPrim`= {`snack`} +{`algoNIM(keepVillage, 2)`}+ `respectCustom`

→exécution de `algoNIM(keepVillage, 2)`

0- Génération de la liste initiale

⇒`listAN`={`satisfyAtPortal`, `satisfyPortalKept`}

1- rejet des AN / AN.isFullySatisfied()

`satisfyAtPortal` =0 (→`insuffisant`=`insatisfait`) // l'agent n'est pas encore au portail
`satisfyPortalKept` =0 (→`insuffisant`=`insatisfait`) // donc le portail n'est pas encore gardé.
⇒`listAN`={`satisfyAtPortal`, `satisfyPortalKept`}

2- tri de la liste

---

[17] Cf. Section IV.3.4 pour cette notion

[18] Ce critère de sélection est pour le moment une hypothèse propre à cette application. Nous reconnaissons que tel n'est pas forcément le cas, un agent pouvant exécuter *snack* même si l'heure de déjeuner est arrivée. Ce choix du *snack* dépend alors sans doute d'autres critères qui ne sont pas encore intégrés dans cette thèse d'où notre hypothèse ci-dessus. C'est la raison pour laquelle le critère " action programmée " n'a pas (encore) non plus été intégré comme critère générique de sélection dans l'algorithme du NIM. De toute façon, ce qui nous intéresse pour le moment dans cette thèse, ce n'est pas vraiment de savoir ce laquelle est importante entre un goûter *snack* et un déjeuner *eatAtHome*, mais plutôt le déclenchement de la motivation sous-jacente. Le choix de ces deux actions dans le scénario ne visant uniquement qu'a nous rapprocher de la réalité.

Comme ils ont le même niveau, catégorie, rang, état et ratio (avec ratio=0), les critères du tri ne pourront pas les départager.

⇒listAN={satisfyAtPortal, satisfyPortalKept }

3- Rejet des AN / not AN.isInSatisfaction ()

Ils sont tous les deux en cours de satisfaction car insatisfaits.

4- Génération des actions,, à partir du AN.action[currentState]

⇒l_1Act={goToPortal, keepVillage}

5- Application des préconditions

preCondAtPortal=1 (→suffisant) // l'agent n'est pas au portail
preCondKeepPortal=0 (→insuffisant) // car justement l'agent n'est pas encore au portail→rejeté

⇒l_1Act={goToPortal}

5bis- Application du critère temps

Cette action n'est pas programmée et donc exécutable. Plus précisément, cette action est programmée mais le temps d'exécution se trouve dans keepVillage (mécanisme de factorisation).

⇒l_1Act={goToPortal}

6- Séparation PR/AC

Comme goToPortal est une primitive, on a
finalPrim= {snack} + {goToPortal}+ {respectCustom}

⇒finalPrim= {snack, goToPortal, respectCustom}

Fin d'étapes : application des conflits

Pas de conflit entre ces 3 actions

⇒**finalPrim= {snack, goToPortal, respectCustom**}

2 keepPortal & respectCustom (02 :15)

Il s'agit du même processus que (1) sauf que tous les besoins physiologiques sont, soit satisfaits, soit ne sont pas en cours de satisfaction. Il n'y a pas non plus d'obstacles. En procédant de la même manière que précédemment, nous aboutissons donc automatiquement à ces deux actions ci-dessus.

`3 goToRiceField & respectCustom (04 :25)`

Ces actions peuvent être expliquées par la combinaison de l'explication de (1) et de (2). Tout d'abord, si nous suivons bien (1), l'action `cultivateRice` a été rejetée parce qu'il n'est programmé qu'à 04 :25, alors que `keepVillage` a été maintenue. Cette fois-ci, c'est le cas inverse qui se produit.

Ensuite, nous ramenons ici l'explication dans (2), sauf que l'action `keepPortal` est remplacée par `goToRiceField`. C'est cette dernière qui est choisie et non `turnRiceField`, située pourtant au même `nd` qu'elle dans `algoNIM(cultivateRice, 2)`. En effet, dans la mesure où l'agent n'est pas encore à la rizière, `turnRiceField` sera éliminée par le critère *précondition*, au profit de `goToRiceField`.

`4 avoidStone & goToRiceField & respectCustom (06 :40)`

C'est le même scénario que (3) sauf qu'il y a une pierre à contourner. Et en vertu de la règle de l'importance (car `satisfyStoneAway` n'est pas satisfait), l'évitement passe en premier et on obtient les actions ci-dessus qui, en outre, ne sont pas en conflit entre elles.

`5 turnRicefield & installScareCrow & respectCustom (11 :30)`

Ici, nous avons le même processus que dans (4) mais un peu plus tard dans le temps (près de 5 heures plus tard) et avec les différences suivantes :

- comme il n'y a pas d'obstacles, `satisfyStoneAway` est satisfait et est éliminé au moment de l'étape 1 ;
- lors de l'exécution de `algoNIM(cultivateRice, 2)`, l'action `goToRiceField` est remplacée par `turnRicefield` et `installScareCrow`. Ces dernières sont exécutées simultanément dans la mesure où elles ne sont pas en conflit.

Voici les détails de `algoNIM(cultivateRice, 2)`.

0- Génération de la liste initiale.

⇒list$_1$AN={satisfyAtRiceField, satisfyRicefieldTurned, satisfyFodyAway}

1- Rejet des AN / AN.isFullySatisfied()

satisfyAtRiceField =1 (→suffisant) // l'agent est à la rizière ⇒rejeté
satisfyRicefieldTurned = 1.192 (→insuffisant) // la surface retournée n'est que de 1.19% de la surface totale
satisfyFodyAway =1 (→limit_l) // l'agent a déjà installé un épouvantail mais il en faut 2. En outre, ce besoin de forme point est satisfait car nous avons initialisé son point minSat à 1.
⇒list$_1$AN={satisfyRicefieldTurned, satisfyFodyAway}

2- Tri de la liste

Le résultat du tri est : satisfyRicefieldTurned>satisfyFodyAway suite au critère *état*. La règle de l'importance ne s'applique pas ici car satisfyFodyAway est déjà satisfait.

⇒list$_1$AN={satisfyRicefieldTurned, satisfyFodyAway}

3- Rejet des AN / not AN.isInSatisfaction ()

Les deux besoins sont tous en cours de satisfaction. Pour satisfyRicefieldTurned, cela est clair car il est dans un état insuffisant. Pour satisfyFodyAway, il est aussi en cours de satisfaction bien qu'il soit déjà à l'état limite car jusqu'ici, à chaque fois que l'on exécute algoNIM(cultivateRice, 2), cette action a toujours été prise en compte. En d'autres termes, satisfyFodyAway n'a jamais été *interrompu* (cf. Section V.3.2 pour cette notion).

⇒list$_1$AN={satisfyRicefieldTurned, satisfyFodyAway}

4- Génération de la liste des actions issue du AN.action[currentState] des besoins

⇒l$_1$Act={turnRicefield, installScareCrow }.

5-Application des préconditions

→les deux préconditions sont satisfaites car elles vérifient toutes que l'agent est bien à la rizière

⇒l_1Act={turnRicefield, installScareCrow }.

5bis-Application du critère temps

Ces deux actions ne sont pas préprogrammées donc peuvent être exécutées[19]

⇒l_1Act={turnRicefield, installScareCrow }.

6- Séparation PR/AC

Comme les deux actions sont toutes des primitives, alors :

finalPrim= finalPrim +{turnRicefield }+{installScareCrow} + ... // complétée par le traitement de satisfyCustomRespected

⇒finalPrim= finalPrim +{turnRicefield, installScareCrow} + ...

Note : Le numéro (5) montré dans la figure VI.10-d, page 155, correspond en fait à plusieurs séquences d'actions effectuées par le paysan durant tout le temps où il travaille sa rizière. Cependant, nous ne pouvions pas toutes les représenter pour des raisons d'espace sur le schéma. En voici alors quelques unes :

- à 08 :07, le paysan exécute snack & respectCustom. L'action snack étant en xor avec le travail, l'agent s'arrête momentanément de travailler.
- à 11 :31, il exécute drink & turnRicefield & installScareCrow. Contrairement à snack, nous n'avons pas mis drink en xor avec le travail d'où la possibilité de continuer à travailler[20].
- etc.

6 goHome & respectCustom (11 :51)

Comme il est 11h51 et que l'agent a faim, il va exécuter cette séquence d'actions plutôt que snack & respectCustom. Il va donc rentrer. A noter cependant qu'à 12 :30, au moment où il rentre chez lui, son état passe à insuffisant (car le fait de marcher fait aussi augmenter la faim). Dans ce cas, il exécute snack & respectCustom car c'est snack qui est à exécuter lorsque satisfyFedUp se trouve dans cet état.

[19] Plus précisément, ces deux actions sont implicitement préprogrammées mais leur temps d'exécution est factorisé au sein de *cultivateRice*.

[20] C'est juste un scénario choisi pour tester l'algorithme. Nous aurions pu aussi bien mettre ces actions en *xor* tout comme avec *snack*.

`7 goToRiceField & respectCustom (14 :00)`

Nous sommes ici dans un cas similaire à (3) mais avec un point de départ différent : la maison au lieu du portail

Remarque : Outre la différence de point de départ, il est également possible que certains besoins de Π aient changé à ce moment-là leur `ccpos` respectif et donc différent de celui qu'il a été durant (3), comme avec `satisfyThirst` ou `satisfyFatigue` par exemple.

`8 goHome & respectCustom (18 :30)`

Nous sommes ici dans un cas similaire à (6) mais avec un horaire différent et avec la remarque générale de (7) sur la modification des besoins.

`9 goToPortal & respectCustom & snack (20 :50)`

Nous sommes ici dans un cas similaire à (1) sauf que la valeur de `satisfyFedUp` n'est pas la même que dans (1). Ici, `ccpos`= 11.59. (état `limite_l` mais besoin *satisfait*) alors que dans (1), `ccpos=10.05` (état `limite_l` et besoin *insatisfait*).

`10 keepPortal & respectCustom (23 :42)`

Nous sommes ici dans un cas similaire à (2) mais avec actions relevées à un horaire un peu différent de (2). La remarque de (7) sur la modification des besoins reste valable.

`11 goToRiceField & respectCustom (04 :25)`

Nous sommes ici dans un cas similaire à (3) mais le jour suivant (02/01/03), et avec un point de départ qui n'est pas tout à fait identique à celui de (3). Ce point dépend du mouvement effectué par le robot durant la garde du village[21]. La remarque générale de (7) reste aussi valable.

VI. 4. 3 Evolution graphique des besoins

Pour avoir une vue plus claire de l'évolution des besoins des agents, une fenêtre graphique contenant des courbes a été mise en œuvre (Figure VI.11). Chaque courbe correspond à un couple *agent/besoin* et est associée à une couleur donnée. Il appartient à l'utilisateur d'associer les couleurs d'une courbe à un couple donné (via la partie supérieure de la

[21] Nous voulons uniquement montrer ici la dynamique du système, même pour un événement s'effectuant périodiquement.

fenêtre) et de l'identifier et de le suivre durant la simulation (via la partie inférieure de la fenêtre).

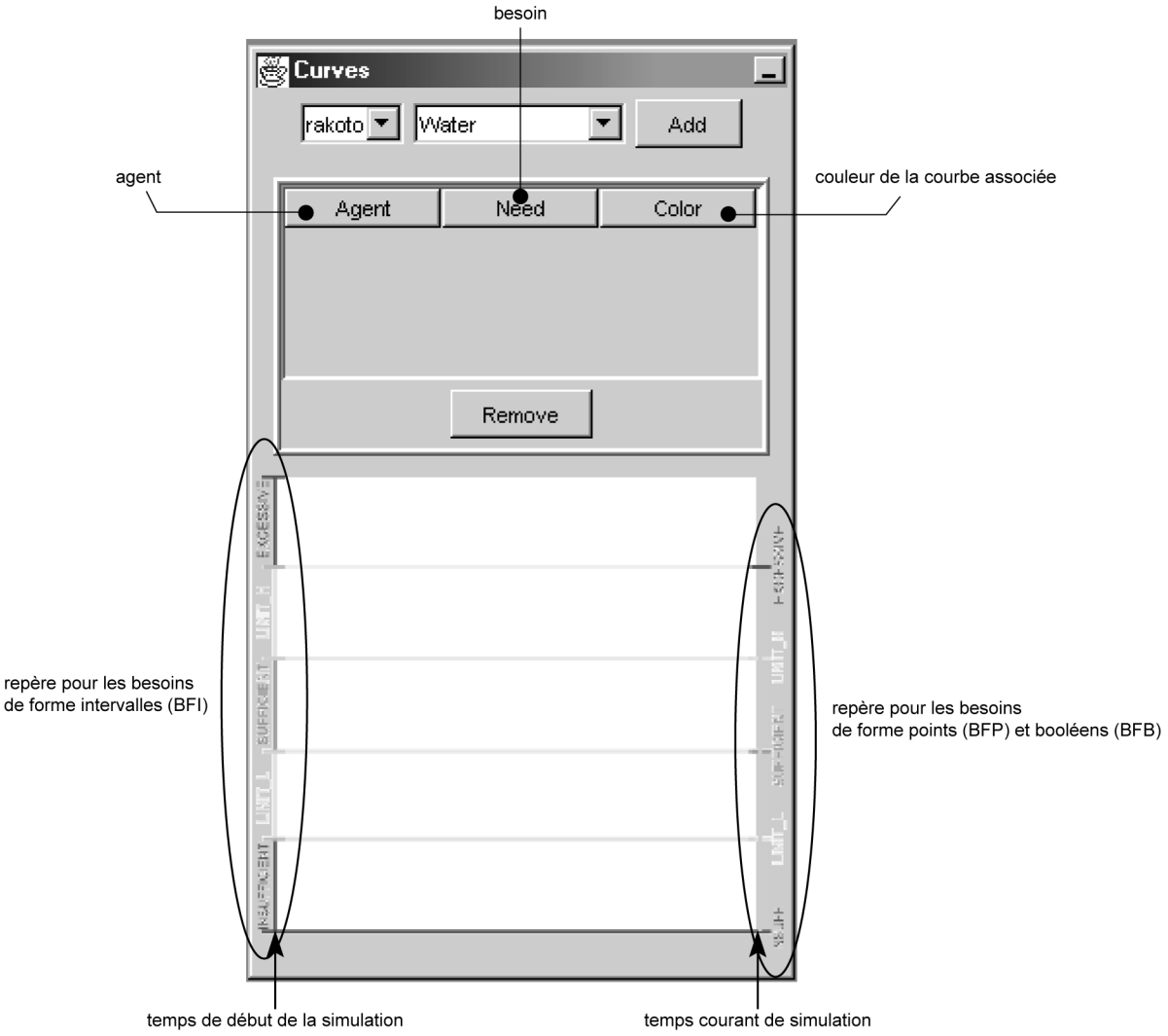


Figure VI.11 : Présentation générale de la fenêtre de suivi de l'évolution des besoins

La partie inférieure de la fenêtre est constituée de deux repères :

- un repère horizontal qui correspond au temps. Il varie entre le temps du *début* de la simulation et le temps *courant* de celle-ci.
- deux repères verticaux. Le repère de gauche correspond à celui des besoins de forme intervalle (BFI) et celui de droite celui des besoins de forme points (BFP) ou booléens (BFB).

En tout, notre fenêtre permet de visualiser, pas seulement l'état d'un besoin mais aussi de plusieurs besoins simultanément. Cependant, ce choix entraîne une contrainte pour les BFI : comme la longueur d'un intervalle associé à un état est souvent différente d'un besoin à un autre, il faut obligatoirement représenter tous les besoins sur une même échelle. La représentation donnée ici indique donc une position relative du curseur par rapport à l'intervalle. Ainsi, il est pour le moment impossible de visualiser si l'énergie du robot est, par exemple, égale à 5. Par contre, il est globalement possible de savoir à chaque instant, quel ensemble de besoins est satisfait et quel autre ensemble ne l'est pas.

Il est clair qu'une autre fenêtre associée à l'évolution graphique de chaque besoin (sans mise à l'échelle) doit aussi être envisagée pour représenter la valeur absolue des curseurs.

La figure VI.12, nous montrons l'exemple d'évolution de quelques besoins, suite au scénario (b)[22]. En particulier, il s'agit de 4 PN : `satisfyThirst`, `satisfyFatigue`, `satisfyStoneAway` et `satisfyAllPucksDelivered`.

Voici à présent comment nous expliquons cette évolution aux échantillons de temps t_1 à t_6.

Instant t_1 :

A l'instant t_1, la courbe de `satisfyFatigue` (qui est, rappelons-le, de scénario `SUF_EXC`) évolue en dent de scie autour du point correspondant à son `minSat`. Cela signifie qu'à partir du moment où l'agent entre dans sa zone de satisfaction, il va commencer à travailler. Or, le fait de travailler implique une augmentation de la fatigue. Et comme les actions `sleep` ou `pause` associées sont en conflit avec toutes les autres actions, la courbe va, soit augmenter à chaque fois que l'agent travaille, soit diminuer lorsque celui-ci se repose. A rappeler que le repos est prioritaire (physiologique) sauf qu'ici, il est exécuté le moins possible étant donné qu'il y a encore du travail à faire (c'est-à-dire que le besoin de travailler `satisfyAllPucksDelivered` n'est pas encore satisfait).

D'un autre côté, la courbe de `satisfyThirst` (de scénario `INS_SUF`) diminue en continu car le besoin en question n'a pas encore atteint sa zone de satisfaction. Le besoin

[22] Les autres scénarios ne sont pas présentés. Cependant, ils suivent la même logique d'explication que celle que nous fournissons ici. Ce qui nous intéresse dans cette section, c'est de montrer le fonctionnement de la simulation graphique de l'évolution des besoins, réalisé durant cette thèse.

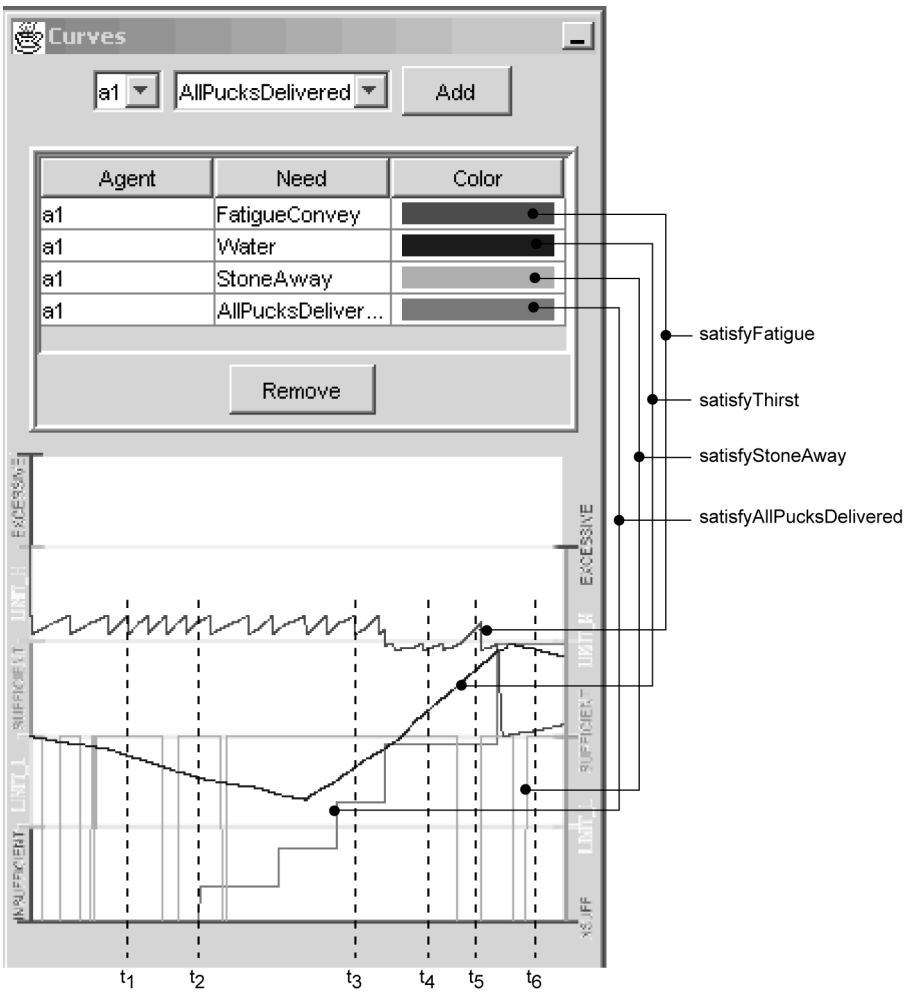


Figure VI.12 : Exemple de suivi graphique de l'évolution des besoins du scénario (b)

`satisfyStoneAway`, quant à lui ne passe dans l'état `insuffisant` que lorsque l'agent se trouve près d'un obstacle. Autrement, ce besoin est à l'état `suffisant`.

Instant t_2 :

Ce moment correspond au dépôt du premier palet à la base, correspondant à une montée (ici en escalier) de la courbe associée au besoin `satisfyAllPucksDelivered`, l'éloignant ainsi progressivement de l'état `insuffisant`. Les autres AN suivent l'explication que nous avons donnée dans l'instant t_1.

Instant t_3 :

Après avoir récemment atteint le point `minSat`, le besoin `satisfyThirst` revient vers un état meilleur, particulièrement lorsque l'agent adopte l'action `drink`. Cependant, la progression est continue et non en dent de scie car l'action `drink` n'est en conflit avec aucune autre action, c'est-à-dire que nous avons supposé qu'il est possible de boire, tout en travaillant.

Instant t_4 :

A cet instant précis, `satisfyFatigue` entre enfin dans l'état `suffisant`, c'est-à-dire que le besoin de se reposer est de plus en plus satisfait. La raison en est que `satisfyAll-PucksDelivered` entre également dans sa zone de satisfaction et que l'agent travaille donc de moins en moins. Il existe ici une complémentarité entre le fait de travailler et de se reposer : plus le travail tend vers sa finition, plus l'agent peut se reposer. Cette complémentarité est confirmée par l'évolution des courbes, déjà mentionnée à l'instant t_1 : lorsqu'il y a encore du travail à faire, l'agent ne se repose qu'au minimum.

Instant t_5 :

Dans cette phase, `satisfyFatigue` sort provisoirement de l'état `suffisant` en raison de l'évitement d'obstacle, associé au travail de transport qui lui, n'est pas encore (tout à fait) terminé.

Instant t_6 :

A présent, le travail de transport est totalement fini. Les besoins `satisfyFatigue` et `satisfyThirst` sont tous à l'état `suffisant` mais en parallèle, ils commencent à évoluer vers leur état respectif d'insatisfaction car l'activité physiologique d'un agent évolue toujours même s'il ne travaille pas. A cela s'ajoute le dernier évitement d'obstacle montré sur la figure, juste avant t_6.

VI. 5 Conclusion du chapitre

Ce chapitre s'est proposé de valider notre modèle au travers de deux cadres d'expérimentations. Le premier cadre concerne les robots fourrageurs, un domaine de validation bien courant dans le monde des SMA. Il est ensuite suivi d'un cadre un peu plus complexe : celui des paysans. Les quatre scénarios proposés dans cette phase

démontrent que non seulement le modèle est bien générique, mais il est aussi applicable dans un cadre hybride, c'est-à-dire réactif et/ou cognitif (tel qu'il est défini à la page 11). Une analyse et une interprétation de simulations ont donc été réalisées pour montrer l'utilisation pratique des concepts théoriques proposés dans les chapitres précédents.

Une fois le modèle MASLOW mis en œuvre et les expérimentations faites, le chapitre VII va discuter l'apport de cette recherche dans le domaine des SMA.

Chapitre VII

Discussion du modèle

Le présent travail va être discuté par rapport à deux aspects complémentaires :

- au résultat expérimental, présenté dans le chapitre VI,
- au cadre plus général lié aux modèles agents existants actuellement dans le champ de recherche SMA.

VII. 1 Discussion par rapport au résultat expérimental

VII. 1. 1 L'aspect hybride du modèle

Au travers de l'expérimentation, nous avons pu constater que c'est le même algorithme du NIM qui a été appliqué pour les scénarios (b), (c) et (d). Cet algorithme a été implémenté dans la méthode `step()` implémentée dans la classe abstraite `AnimatController`. Cette *même* classe a été le pilote des trois scénarios. Or comme nous le savons, le scénario (b) présente un agent réactif et les scénarios (c) et (d), des agents cognitifs. En outre, tous les besoins AN sont *toujours* motivés par au moins un LN (qui représente le concept générique du naturel). Le présent modèle basé sur les motivations naturelles s'intègre donc aussi bien dans des agents artificiels réactifs, que dans ceux cognitifs (et donc dans ceux hybrides).

En fait, notre idée à la base est que le NIM choisit et déclenche les actions sans tenir compte ni de leur origine (plan d'un agent cognitif, comportement inné d'un agent

réactif, etc.) ni de leur contenu (qui peut inclure des informations venant du percept, des informations provenant des connaissances, etc.). Il n'agit que du point de vue externe c'est-à-dire que son rôle se concentre sur le déclenchement d'un ensemble d'actions associées à un ensemble de motivations préalablement sélectionnées, tout en ignorant le détail du contenu interne de l'action. Certes, nous avons introduit la notion du temps qui intervient dans le paramètre des actions. L'initialisation de ce paramètre n'est cependant pas obligatoire pour que l'algorithme du NIM fonctionne. Il nous sert pour sélectionner une action et non à modifier son contenu.

Sur une vue plus globale de l'architecture du travail, cette distinction entre MASLOW et l'implémentation des actions est plus visible dans la connexion ADK-MASLOW. Alors que le système motivationnel est implémenté dans MASLOW, les actions sont toujours implémentées dans ADK via les contrôleurs. Or, MASLOW ne connaît pas ADK[1]. C'est plutôt l'inverse qui est vrai, c'est-à-dire que c'est ADK qui exploite les concepts de MASLOW. Ainsi, que ADK simule des agents réactifs ou cognitifs, MASLOW fonctionnera toujours. C'est au niveau de l'implémentation des actions que se dégage la différence de comportement du fourrageur dans les scénarios (b) de (c) : le robot cognitif effectue, pour sa collecte, un va-et-vient sur une zone précise jusqu'à ce qu'il n'y trouve plus de palets (comportement résultant de sa connaissance mentale du milieu), alors que le robot réactif doit, après chaque livraison d'un palet, repartir vers une nouvelle exploration. Mais ces deux robots utilisent le même modèle MASLOW.

VII. 1. 2 MASLOW, un modèle agent générique

Déjà, le fait que nous utilisions la même version du NIM pour tous les scénarios intégrant MASLOW fait de ce modèle un modèle générique. Mais cette analyse a été faite *a posteriori* c'est-à-dire à partir de résultats. Dans cette section, nous menons notre explication *a priori*, en nous basant sur les détails conceptuels même du modèle, en faisant intervenir notre rôle dans le système en tant que modélisateur et en tant qu'utilisateur.

Dans le scénario (a), le robot agit effectivement en respectant correctement l'objectif qu'il s'était fixé initialement, à savoir le transport des palets à la bonne base. Cependant sur le plan conceptuel, l'ajout d'un nouveau comportement à effectuer par ce robot doit repasser par la reprogrammation informatique de la classe `ForagerController`

[1] Il suffit de voir la figure VI.4 (page 135) pour s'en rendre compte : le NIM est appelé par le contrôleur, mais le contrôleur n'est pas connu du NIM.

(telle qu'elle est présentée dans la figure VI.3, page 133). Plus précisément, cette programmation concerne la mise à jour de la méthode `step()` qui implémente les règles de comportement (via le sélecteur `case` contenue dans cette méthode) établissant le lien entre situations[2] et actions. Le problème est que ici, tout le mécanisme est codé au niveau applicatif. Ainsi, si nous passons d'une application à une autre (par exemple des fourrageurs aux paysans) en utilisant le même modèle, il faut de nouveau reprogrammer toutes les règles de comportement des paysans, et par analogie au fourrageur, dans la classe `PeasantController` qui se trouve elle aussi au niveau applicatif. De la même manière, il faut aussi recoder la transition d'états telle qu'elle a été présentée dans la classe `HandCodedController`.

Dans notre modèle à base de motivations MASLOW, le mécanisme de gestion du comportement comme nous le savons, se situe à un niveau générique. Pourtant, alors que les besoins de la pyramide évoluent en permanence[3], la figure VI.10, page 155, montre que dans les scénarios (b) et (c), le robot adopte un comportement aussi cohérent que dans le scénario (a) préprogrammé par l'utilisateur : évitement d'obstacle s'il y a lieu, livraison des palets à la bonne base, etc. Il en est de même pour le scénario (d) avec le paysan. Mais l'avantage de notre approche par rapport à celle de (a) est que en outre, nous avons pu, lors de la création des trois derniers scénarios, enrichir le comportement du robot d'une manière significative sans avoir à reprogrammer le mécanisme de sélection[4]. Ainsi dans (b) et (c), des besoins tels la satisfaction de la soif et de la fatigue ont été introduits par rapport à (a). Dans le scénario (c), il y a eu en sus, la satisfaction de la faim, la réalisation de la culture du riz, la garde du village et le respect de la coutume. Or, malgré l'ajout de ces besoins, et quel que soit le scénario expérimental, la cohérence de l'ensemble des actions du robot est maintenue. Ainsi, le NIM, qui rappelons-le, est le même pour les trois scénarios, fonctionne indépendamment du nombre, de la sémantique et de l'évolution des besoins situés dans la pyramide.

Certes, il faut implémenter les actions (sinon comment l'agent pourra-t-il s'activer au niveau de l'application ?) mais elles ne sont ici que des éléments " passifs " qui n'attendent qu'à être déclenchés en fonction des motivations de l'agent. Dans la version actuelle d'ADK avec MASLOW, les classes `ForagerController` et `PeasantController`

[2] Pour le scénario préprogrammé, la condition de déclenchement d'une action, c'est l'état dans lequel l'agent se trouve.

[3] Cette évolution peut être simultanée ou non pour tous les besoins, en fonction des actions effectuées dans les cycles précédents.

[4] D'autant plus que les scénarios peuvent concerner deux applications différentes.

contiennent uniquement l'implémentation de chaque primitive c'est-à-dire des actions de base exécutables. A ce niveau, il n'existe aucune description de l'enchaînement de l'exécution des actions puisque la méthode `step()` est déjà remontée vers une classe abstraite.

VII. 2 Discussion générale

VII. 2. 1 MASLOW et les modèles animats

Modélisation du comportement

Divers modèles ont déjà été proposés dans la modélisation du comportement chez les animats qui, rappelons-le, intègrent le plus la notion de motivations naturelles. Parmi ces modèles, les plus connus sont les modèles hiérarchiques, les modèles ascendants et les modèles par combinaison d'action tendancielle (voir aussi Section II. 2. 3 du document).

Cependant, tous ces modèles ont actuellement leur limite respective. Comme le dit Ferber, les modèles *hiérarchiques* sont critiqués par l'existence d'une certaine forme de dominance entre les actions. L'ordre est donné par en haut et les actions qui se trouvent à un niveau inférieur peuvent être ainsi inhibées par des actions de niveau supérieur. Dans les modèles *ascendants* incluant ceux à tâches compétitives (Maes 1991, Tyrrell 1993, Drogoul et Ferber 1994), le choix dépend des conditions environnementales et des liens que les actions ont institués entre elles. Mais dans tous les cas, une action, et une seule, est sélectionnée. Les autres sont de facto mises en sommeil. C'est dans le cadre des actions par *combinaison tendancielle*, utilisant des champs potentiels qu'il est possible d'exécuter plusieurs actions à la fois.

Dans notre approche, MASLOW peut combiner ces trois types de modèles, en en tirant les avantages tout en essayant d'en réduire les inconvénients. En effet :

- notre modèle, de par la forme même de Π, ainsi que des spécifications de Abraham Maslow, est hiérarchique. Cependant, contrairement aux critiques formulées ci-dessus, les PN de niveaux inférieurs de la pyramide ne bloquent pas de facto ceux des niveaux supérieurs. Comme nous le remarquons dans l'algorithme du NIM (Section V.3.2) à l'étape 2, l'évaluation de l'importance entre deux PN sert uniquement à les trier dans un ordre précis et non à

éliminer le moins important. Pour être plus précis, ce n'est pas au niveau de Π (et donc de l'aspect hiérarchique) qu'il existe un quelconque blocage entre les besoins.

– notre modèle, au travers de Ω est un modèle ascendant. La mise en relation des actions se fait via des connecteurs et les conditions environnementales sont implémentées par le paramètre `precondition`. Rappelons que ce modèle est inspiré de l'architecture ANA de Maes qui est lui-même un modèle ascendant. La différence avec ANA se situe cependant au niveau des critères déterminant les motivations qui déclenchent les actions du réseau. MASLOW offre des critères plus riches que ceux de ANA, notamment avec les notions issues de Π telles que le *niveau*, et la *catégorie* dont les valeurs sont issues de l'étude du monde réel. A cela s'ajoutent les notions de *type*, de *rang* et *d'états*, toutes étant des critères *génériques*. L'état permet en outre à un besoin d'être associé non pas forcément à une mais plusieurs actions en fonction du contenu de `pn.action[state]`.
– notre modèle peut s'intégrer dans des systèmes à combinaisons tendancielles (Simonin et Ferber 2001, Calderoni 2002). Preuve en est faite avec l'application du modèle sur ADK. En fait, le but général de l'algorithme du NIM lui-même à la base (c'est-à-dire indépendamment du système qui l'utilise), c'est de dégager un ensemble d'actions qui peuvent s'exécuter simultanément. C'est cette fonctionnalité de simultanéité qui est exploitée dans ADK, de façon à aboutir à un ensemble de tendances à satisfaire via l'exécution de l'action `move`. Et comme cette dernière action est totalement inconnue de MASLOW mais est propre à ADK et aux systèmes à combinaisons tendancielles en général, MASLOW s'offre à ces systèmes tout en n'en étant pas dépendant. Cette indépendance est possible car comme nous l'avons spécifié dans la section VII. 1. 1, MASLOW ne doit pas tenir compte de l'implémentation des actions. Dans ADK, MASLOW ignore qu'il est actuellement en train de manipuler des tendances. Il ne fait que lancer les actions.

L'indépendance de MASLOW par rapport à un système donné se manifeste aussi par la possibilité de l'intégrer dans des modèles où l'agent exécute soit une, soit plusieurs actions. Pour cela, il suffit de mettre toutes les actions du système en conflit entre elles.

Automatiquement, seule la première action, issue du besoin le plus important dans Π (c'est-à-dire `finalPrim[0]`), sera exécutée.

MASLOW et la modélisation des concepts naturels

Dans la psychologie animale, le comportement des animaux se base sur des phénomènes naturels de stimulus/action et d'attraction/répulsion (Bonabeau et al. 1994). La modélisation des actions de ces agents passe donc entre autres par l'utilisation des vecteurs et des champs potentiels utilisant des vecteurs. En ce sens, nous pensons que les modèles animats progressent vers une " conformité " avec cet aspect naturel du monde animal. La modélisation se base cependant sur des exemples *ponctuels* dont les principaux sont la faim ou la soif (traduit en niveau d'énergie pour les robots), l'évitement d'obstacle ou encore dans certains cas, l'étude du transport de nourriture (ou des minerais pour les robots). L'avantage de ce procédé est que ces exemples simples aident rapidement à comprendre le comportement étudié. Cependant, limiter les exemples d'étude revient à rester à un niveau local de la conception. En effet, les besoins naturels innés dans l'espèce animale sont a priori très variés et cela doit être conceptuellement pris en compte. En conséquence, il est préférable de proposer une structure abstraite englobant ces besoins. C'est par la suite que des spécialisations via des études de cas devraient être envisagées à un niveau applicatif. L'introduction de LN à un niveau générique du modèle constitue un effort dans ce sens.

Voilà pour les agents réactifs. Passons à présent aux agents cognitifs et hybrides.

VII. 2. 2 MASLOW et les modèles hybrides et cognitifs

Suite à l'intégration du naturel à un niveau générique, il devient de ce fait exploitable dans le cadre des agents hybrides et cognitifs. Jusqu'ici dans les modèles hybrides, le naturel a été :

- soit complètement abstrait comme c'est le cas des modèles que nous avons vu dans la section II. 4 de ce document,
- soit pris au cas par cas uniquement en fonction des besoins de l'application, le plus souvent étant l'évitement d'obstacles (Ferguson, 1992, Zeghal 1994, Tambe 1996, Au et al. 1998, ...). Avec MASLOW, le mécanisme est

généralisé, et non plus pris au cas par cas. Par exemple, l'évitement d'avion dans le modèle organisationnel de Tambe (1996) correspond dans la structure générique de MASLOW à la remise à l'état `satisfait` d'une instance du besoin de sécurité, précédemment modifié par l'approche d'un autre avion.

Le scénario hybride peut désormais être plus enrichi davantage. En effet, les aspects liés au naturel dans le monde hybride ne se limitent plus à des notions de sécurité comme l'évitement d'obstacle. Il existe aussi d'autres notions physiologiques. En effet, nous avons vu avec MASLOW une large connaissance des activités de l'agent, allant du plus général (travailler au champ tous les jours) au plus détaillé (un petit goûter à 08h07, c'est-à-dire un besoin naturel autre que sécurité). L'idée est que même si ces notions de naturel n'intéressent pas toujours forcément une application hybride donnée, d'autres applications hybrides qui en auront besoin ne devraient pas être limités en raison de l'absence de ces notions. Notre objectif est de fournir des spécifications, que celles-ci soient utilisées par les applications ou non.

Un dernier aspect lié à l'hybridisme dans le modèle concerne cette dualité MN/HN, et plus particulièrement le basculement entre la satisfaction de deux besoins : une satisfaction en temps réel (immédiatement) des besoins naturels et une satisfaction prévisionnelle de ces mêmes besoins. Bien que la limite entre le réactif et le cognitif soit floue, ce dont on est sûr pour ce modèle, c'est que les agents cognitifs possèdent ici *de facto* le côté HN de la pyramide vu leur capacité à anticiper. D'un autre côté, les agents réactifs dont la vie est davantage basée sur les instincts possèdent *de facto* le niveau MN. Ensuite, l'existence ou non des HN au niveau réactif dépend des agents à modéliser. Dans tous les cas, l'hybridisme est présent au niveau des besoins. C'est ce qui distingue MASLOW des modèles hybrides comme INTERRAP (Müller, 1994) dans lesquels le basculement réactif/cognitif se fait plutôt au niveau des actions. En fait, ces modèles sont complémentaires au nôtre. Nous pouvons par exemple envisager que pour un PN donné, l'action `PN.action[limit_l]` sera gérée par la partie cognitive de INTERRAP alors que `PN.action[insuffisant]` sera gérée par sa partie réactive.

Enfin, par rapport à BDI, MASLOW peut être analysé de la manière suivante :

- Le “ Belief ” correspond à une instanciation particulière de l'attitude informationnelle de l'agent.
- Le “ Desire ” correspond aux HN, motivé donc à la base par la satisfaction des MN.
- Seule l'intention n'est pas encore explicitement intégrée par MASLOW. Elle existe dans ADK lorsque l'agent souhaite envoyer des influences dans l'environnement mais que cet envoi se trouve en contradiction avec une instance de la loi de l'univers.

VII. 2. 3 La tâche des utilisateurs de MASLOW : facilité et complexité

Alors que la programmation informatique du mécanisme de la sélection de motivations ou d'actions est éliminée pour l'utilisateur de MASLOW, celui-ci peut néanmoins se trouver face à une situation assez complexe[5] inhérente au modèle lors de l'initialisation de Π et de Ω. Comme nous le savons, l'essentiel de cette initialisation consiste, non seulement à établir les paramètres des besoins et des actions, mais à connecter ces éléments entre eux, et plus précisément :

- la connexion des besoins à un ensemble d'actions,
- la connexion des actions à un ensemble de préconditions,
- la connexion des actions à un ensemble d'actions,
- mais aussi la connexion des besoins AN à des besoins LN.

Dans l'initialisation de chaque AN, il faut donner le nom des LN associés, le rang ainsi que les états (revoir Equation (IV.6), page 95). En sus, un état contient aussi des paramètres comme la forme, le scénario, la description de la forme, ainsi que le(s) point(s) `minSat`. Maintenons, si nous imaginons le nombre de besoins à satisfaire et le nombre de préconditions associées aux actions pour une application donnée, nous pouvons avoir une idée de l'ampleur des tâches à réaliser. Déjà, un premier effort a été effectué en n'ayant pas besoin de déclarer les critères *type*, *niveau* et *catégorie* au sein des AN. Néanmoins, davantage de réductions sont encore préférables.

[5] Un aperçu de cette complexité se trouve dans la section VI.3.3 lorsque nous initialisons les applications. Voir aussi par exemple la figure VI.7, page 141.

Les actions par contre sont plus faciles à initialiser une fois que les besoins sont établis. Cependant, les problèmes liés aux actions concernent la liaison des actions entre elles. Il est plus facile de retrouver les actions qui sont à établir par les connecteurs `imp` et `then` puisque ce sont généralement des successions verticale (pour le `imp`) et horizontale (pour le `then`). En revanche, retrouver les actions à connecter par le `xor` est plus ardu, notamment lorsque le nombre de données à manipuler au niveau applicatif augmente. Parfois, c'est la simulation qui aide à établir les connexions, en constatant progressivement des comportements incohérents qui sont constatés au sein de l'agent. Par exemple, l'agent va à la fois à la rizière et à la maison (en prenant le milieu des deux destinations).

Malgré ses difficultés, nous pouvons synthétiser notre apport au niveau utilisateur comme suit :

- réduction de la tâche de l'utilisateur, en éliminant les aspects " programmation informatique " du comportement des agents et proposition par la suite d'une méthodologie visant à bien structurer cette tâche lors de l'initialisation de l'application.
- liberté de l'utilisateur sur la définition des AN en fonction de ses propres interprétations d'un besoin. L'essentiel est qu'il sache à quel(s) LN ce AN correspond.
- possibilité d'offrir à des psychologues, des biologistes ou des sociologues des moyens d'étendre le modèle pyramidal (éventuellement à un niveau générique via les LN) en enrichissant potentiellement celui-ci avec des règles ou des fonctions qui sont propres à ces disciplines.

Connecter Maslow à un modèle générique de planification serait aussi envisagé pour faciliter au mieux cette gestion de l'initialisation du modèle. Cela nous permettrait en parallèle d'améliorer la manière dont le temps est géré.

VII. 3 Conclusion du chapitre

L'objectif principal de ce chapitre est de situer l'apport de ce travail de thèse dans la modélisation du comportement d'agents : montrer que le modèle proposé, basé sur la motivation naturelle est effectivement hybride et générique. Il s'applique, non seulement aux agents réactifs (les animats) comme tel a été le cas jusqu'ici dans le domaine SMA,

mais aussi aux agents cognitifs (et donc hybrides). Différentes comparaisons par rapport à ces autres modèles ont été discutées.

Nous présentons également ici une analyse critique de notre travail. Celle-ci concerne principalement la complexité de l'initialisation du modèle par l'utilisateur. C'est un aspect du travail sur lequel il nous faudra pencher car l'idéal étant effectivement d'aboutir à un modèle, certes, générique, mais aussi facile d'utilisation.

Conclusion générale

Dans les études menées dans le cadre des SMA, les motivations naturelles (c'est-à-dire celles qui sont liées à la satisfaction des besoins naturels) ne sont souvent considérées que dans les modèles d'agents réactifs et généralement ignorées dans un contexte cognitif et hybride. Pour ce dernier, la partie " réactive " n'est gérée que d'une manière élémentaire (règle de gestion simple fournie par l'utilisateur et non l'agent). Dans l'ensemble, le concept *d'agents hybrides* et celui de *motivations naturelles* sont encore faiblement couplés. Le but de cette thèse était de réduire cette faiblesse, en proposant plus particulièrement une structure générique qui tient compte en permanence de cette motivation, indépendamment du type d'agent et du contexte d'application. En résumé, quels que soient les objectifs, les agents ont pour " ambition " de satisfaire un LN. L'évolution du système ne dépendra ensuite que de l'activité des agents qu'il contient, associé à divers paramètres (rôles, règles d'organisation, etc.).

Pour atteindre notre objectif, nous avons considéré, outre l'aspect SMA, certains comportements des êtres humains mais aussi des animaux. En conséquence, la partie conceptuelle du travail a été basée sur l'adaptation dans une perspective agent de la pyramide des besoins du psychologue Abraham Maslow. Il nous semble logique que la construction d'agents artificiels parte d'un modèle psychologique existant afin de nous approcher davantage de la réalité. Dans notre vie quotidienne, il est presque devenu une habitude pour l'auteur de cette thèse de se demander pour chaque situation qui arrive : " *à quel niveau de la pyramide cet événement peut- il correspondre ?* "

Ce qui a été réalisé

Malgré les divers problèmes que nous avons rencontrés dans la réalisation de cette thèse, le bilan provisoire que nous pouvons tirer de notre travail est plutôt positif. Nous avons ainsi progressé à la fois dans la modélisation de motivations ou de buts que dans la conception d'architectures hybrides. En effet, nous sommes cette fois-ci allé au-delà de ces architectures où les couches ne sont composées que de décisions ou de comportements, aboutissant au fait que ces architectures hybrides font souvent abstraction de toute la première partie du fonctionnement du système conatif : le système motivationnel.

Par ailleurs, en associant le fonctionnement de l'instinct et de l'homéostasie à un modèle conceptuellement générique et validé dans le monde réel, nous avons pu progresser dans la représentation de l'autonomie décisionnelle et comportementale du concepteur sur l'agent. En effet, ce dernier agit maintenant " en toute connaissance de cause ". Au travers de ce travail, nous avons tenté de représenter la pyramide d'Abraham Maslow vers les agents. Ainsi, ce sont ces derniers qui " sentent " et qui agissent et non plus le concepteur.

Ce qui n'a pas été réalisé

Le présent travail n'est évidemment pas un aboutissement. Aussi, à la question de savoir ce qui n'a pas été réalisé, nous serions tenté de dire : à peu près tout ! Ce qui serait quand même un petit peu injuste. Mais il est vrai que tant de choses restent à faire et que tant de concepts nous font encore défaut ; des concepts qu'il est probable que nous n'ayons abordés dans ce travail qu'une parcelle de la partie visible de l'hybridisme.

L'apprentissage

L'introduction du mécanisme d'apprentissage dans la sélection des motivations et des actions figure parmi nos priorités. En effet, l'apprentissage est un des facteurs importants dans l'autonomie de l'agent. Ainsi, au fur et à mesure de l'évolution de l'activité d'un agent, son choix dans la sélection des PN va progressivement converger vers un ensemble donné qu'il juge comme le plus fréquemment à satisfaire. En conséquence, deux agents disposant initialement des mêmes données (croyances, besoins, etc.) au niveau du contrôleur peuvent ensuite évoluer dans deux directions totalement différentes.

Une idée de modélisation serait par exemple d'étendre le modèle d'apprentissage de Calderoni (2002) qui emploie les algorithmes génétiques. Un travail dans ce sens n'est pas à écarter dans la mesure où MASLOW est déjà connecté à la plate-forme ADK de Calderoni.

L'adaptation

Il devrait aussi y avoir la faculté de l'agent à s'adapter aux contraintes imposées par son environnement. Nous savons que l'adaptation est quasiment une caractéristique innée d'un agent et d'ailleurs, influence la décision de celui-ci lors de la sélection d'actions. Dans le modèle MASLOW par exemple, les délimiteurs d'états des PN c'est-à-dire les points `IL`, `LS`, etc. (cf. Section IV. 1. 3) devraient aussi, tout comme le curseur, pouvoir " glisser " le long de l'axe[6] pour se stabiliser par la suite respectivement sur une valeur donnée. Une personne ayant vécu dans un endroit chaud et qui migre au pôle nord verra par exemple ses délimiteurs d'états changer progressivement de valeur pour se stabiliser petit à petit sur un point donné en fonction de la température de son nouvel environnement.

La partie sociale

Jusqu'ici, les besoins sociaux introduits dans la pyramide ne concernent que l'agent lui-même. Ainsi, la pyramide de Maslow a été suivie quasiment à la lettre (individuel>social). Or, l'on sait bien que pour un agent social, l'état des besoins de l'autre, est aussi inclus (éventuellement en partie) dans ses propres besoins, et les influence. D'ailleurs, selon la théorie de Durkheim (1995), il existe des situations dans lesquelles une personne sacrifie sa vie au bénéfice d'êtres chers. Les études de Durkheim ont montré qu'en certaines circonstances, l'instinct de survie pouvait être annulé par le pouvoir de l'instinct relationnel. En conséquence, les critères initiaux de Abraham Maslow (*besoins primaires>besoins sociaux*) ne sont plus les seuls critères à prendre en compte. Il faut aussi intégrer dans le modèle le degré de rapport social entre deux ou plusieurs agents.

Globalement, l'extension de la partie sociale part de la relation suivante :

$$a_1 \overset{agit\ sur}{\longrightarrow} a_x \overset{pour\ satisfaire\ les\ besoins\ de}{\longrightarrow} a_y$$

[6] Ce coulissement des délimiteurs d'état se fait évidemment sans que les états eux-mêmes ne se chevauchent. En outre, cette variation ne se fait que sur un déplacement bien déterminé en fonction de la sémantique du besoin et du degré d'adaptation.

En faisant varier les valeurs de `x` et de `y`, le modèle peut être enrichi d'une manière significative.

Le cognitif

Comme nous le savons, ce travail est plutôt parti du bas niveau (instinct, etc.). Même si des concepts cognitifs sont aussi introduits (page 122), il faut encore l'enrichir au travers de modèles génériques de raisonnement, de manipulation de symboles et de planification afin de mieux gérer cet aspect de haut-niveau. Cela nous permettra aussi d'améliorer la manière dont nous gérons le temps dans notre modèle.

Un agent non sincère ?

Enfin, nous prévoyons d'inclure dans notre agent (en tout cas dans sa partie cognitive) l'aspect de *non sincérité*[7]. Ce terme signifie que nous devrons maintenant traiter le cas où un agent cache délibérément ses vraies motivations et les substitue par d'autres, en justifiant la raison de cette attitude. Par exemple, pour une raison pas forcément connue extérieurement, un agent va consommer de la nourriture alors qu'il n'a pas forcément faim.

Méthodologie de l'utilisateur

Nous avons pu remarquer au travers de cette thèse combien l'utilisateur peut encore éprouver des difficultés à initialiser le système. Cette situation a d'ailleurs été rappelée à la section VII. 2. 3, page 184. Il convient donc pour nous dans un futur proche de réduire encore aussi loin que possible, les tâches à effectuer par l'utilisateur tout en lui offrant le maximum de possibilités dans l'initialisation de l'application.

Ce qui devra être réalisé un jour

Dans un cadre assez lointain, il faudra penser à bâtir la pyramide d'un groupe. Effectivement, un groupe possède aussi sa pyramide : il a besoin de vivre, d'être en sécurité (financière par exemple) et aussi d'être en contact avec ou d'être reconnu par d'autres groupes. Ces groupes qui prennent forme sous diverses appellations (associations, coopératives, etc.) existent déjà de par le monde. Cette pyramide de groupe sera

[7] Voir aussi la section I.2.4, page 16.

ensuite associée par exemple à des modèles de coopération pour aboutir à la modélisation de la *motivation de groupe*.

Certes, le niveau d'études à ce moment-là deviendra plus complexe. Nous ne parlerons plus d'instinct ni d'homéostasie dans le sens où nous le connaissons actuellement. Mais il est clair que l'instinct (réel) de l'agent intervient constamment, même pour faire fonctionner une association. Comment par exemple seront établies les relations fonctionnelles entre les deux niveaux ? Voilà des questions qui méritent d'être résolues dans l'avenir.

Epilogue

Le travail présenté dans le cadre de cette thèse constitue une ouverture vers un certain nombre de *possibilités* qui, nous l'espérons, constitueront une part non négligeable de la recherche de demain en Vie Artificielle et en Intelligence Artificielle. La clé de cette ouverture passe sans doute par un très important travail expérimental, et, de ce point de vue, les Systèmes Multi-Agents ont tout à gagner sur des collaborations interdisciplinaires comme celles que nous avons pu mener sur le projet MASLOW, avec des géographes, des géophysiciens, des psychologues.

Nous souhaitons, par l'écriture de cet ouvrage, avoir progressé et avoir permis à d'autres chercheurs de progresser ne serait-ce que de quelques pas dans la compréhension de la modélisation du comportement naturel et de l'hybridisme. En tout cas, s'il ne fallait retenir qu'une chose de cet ouvrage, c'est qu'il nous reste encore beaucoup de choses à apprendre du vivant avant de pouvoir nous lancer dans l'aventure de l'artificiel.

Nous souhaitons également que le point suivi dans cette thèse, à savoir l'explicitation des motivations dans le domaine hybride amène de nombreux lecteurs à explorer cette voie, très certainement riche en surprises et découvertes de toutes sortes. C'était là toute l'ambition de ce travail, et nous espérons qu'elle aura été perçue comme telle.

Références

Andriamasinoro Fenintsoa (1999). *Gestion de la Biomasse Agricole Par Système Multi-Agents.* Rapport de stage de DESS Réseaux-Multimédia-Internet à l'IREMIA-Université de La Réunion, 45 pages.

Andriamasinoro Fenintsoa, Courdier Rémy (2000). *A model of virtual competition between remote agents.* In proceedings of International ICSC Symposium on Multi-Agents and Mobile Agents in Virtual Organizations and E-Commerce (MAMA'2000), december 11^{th}-15^{th}, 2000, Wollongong, Australia, CD-ROM and also visible at *http ://www.wbmt.tudelft.nl/pto/research/conferences/Proceedings/icsc/012.html.*

Andriamasinoro Fenintsoa, Courdier Rémy (2001a). *Un modèle dynamique de comportement agent à base de besoins.* In Journées Francophones sur l'Intelligence Artificielle Distribuée et les Systèmes Multi-Agents (JFIADSMA'01), 12-14 novembre, Montréal, Québec, Canada (Hermès Eds.), pp-351-353.

Andriamasinoro Fenintsoa, Courdier Rémy (2002a). *The Basic Instinct of Autonomous Cognitive Agents.* In Proceedings of 1^{st} International Congress on Autonomous Intelligent System (ICAIS'2002), February 12^{th}-15^{th}, Geelong, Australia, in CD-Rom, and Abstract in World Scientific and Engineering Academy and Society Press, (ISBN 3-906454-30-4), p. 61.

Andriamasinoro Fenintsoa, Courdier Rémy (2002b). *Integration of Generic Motivations in Social Hybrid Agent.* In International Workshop on Regulated Agent-Based Social Systems : Theories & Application (RASTA'02) at the 1^{st} International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS' 2002) July 15-19, 2002, Bologna, Italy, pp 71-91.

Andriamasinoro Fenintsoa, Courdier Rémy, Piquet Eric (2001b). *Enhancing a Multi-agent System's Performance : From Implementation to Simulation Analysis.* In Proceedings of the 2001 IEEE International Symposium on Cluster Computing and the Grid (CCGrid'01), May 16^{th} -18^{th}, Brisbane, Australia, pp 464-469.

Au Sherlock, Liang Jiasen, Parameswaran N (1998). *Progressive plan execution in a dynamic world* in Dynamic and Uncertain Environments Workshop in Artificial Intelligent Planning, Eds. *Ralph Bergmann, Alexdrader Kott,* Pittsburgh USA, AAAI, pp 136-143.

Barbato Antonio (2001). *Instincts, centres et sous-types.* In Enneagram Monthly Journal, #77 - vol. 7, n ° 10, November.

Behnke Sven, Raul Rojas, Wagner Gerd (2000). *A Hierarchy of Reactive Behaviors Handles Complexity.* In Workshop on Balancing Reactivity and Social Deliberation in Multi-Agent Systems at the 14th European Conference on Artificial Intelligence (ECAI), Berlin, Germany, Lecture Notes in AI, volume 2103, Springer, pp. 125-136.

Bonabeau Eric, Dorigo Marco, Theraulaz Guy (1999). *L'intelligence en essaim.* In Journées Francophones sur l'Intelligence Artificielle Distribuée et les Systèmes Multi-Agents (JFIADSMA'99), 7-9 novembre, Ile de La Réunion, France, pp. 13-25.

Bonabeau Eric, Theraulaz Guy (1994), *Intelligence collective*, Edition Hermes.

Bond Alan et Gasser Les (1988). *Readings In Distributed Artificial Intelligence,* sous la direction de Bond A. H. et Gasser L., Morgan Kaufman.

Bratman Michael, Israel David, Pollack Martha E (1988), *Plans and resource-bounded practical reasoning.* Comput. Intelligence 4, pp. 349-355.

Brazier Frances, Dunin-Keplicz Barbara, Treur Jan, Verbrugge Rineke (1999). *Beliefs, Intentions and DESIRE.* Modeling internal Dynamic behavior of BDI agents. In JJ Meyer, PY Schobbens (eds.) Formal Models of Agents : ESPRIT project Modelage final workshop, Lecture Notes in AI, volume 1760, Springer, pp 36-56.

Brooks Rodney (1986), *A Robust Layered Control System For a Mobile Robot*, IEEE Journal of Robotics and Automation, Vol RA-2, No. 1, pp. 14-23.

Burloud (1936). *Principe des psychologies des tendances.* Paris.

Buytendijk Frederic J. J. (1952). *Traité de psychologie animale.* Presses Universitaire de France. 362 pages.

Calderoni Stéphane (2002). *Ethologie Artificielle et Contrôle Auto-Adaptatif dans les Systèmes d'Agents Réactifs : de la Modélisation à la Simulation.* Thèse à l'Université de La Réunion, 175 pages.

Camalot Jean-Pierre (2000). *Aide à la décision et à la coopération en gestion du temps et des ressources.* Thèse à l'Institut National des Sciences Appliquées (INSA) de Toulouse, France.

Chevrier Vincent (2002). *Contributions au domaine des systèmes multi-agents.* Mémoire d'Habilitation à Diriger des Recherches. Université Henri Poincaré, Nancy, 71 pages.

Courdier Rémy, Guerrin François, Andriamasinoro Fenintsoa, Paillat Jean-Marie (2002). *Agent-based simulation of complex systems : application to collective management of animal wastes.* In Journal of Artificial Societies and Social Simulation vol. 5, no. 3, 30 juin. Visible sur *http ://jasss.soc.surrey.ac.uk/5/3/4.html*

Daco Pierre (1965). *Les triomphes de la psychanalyse*, Marabout Service, MS 29

De Jong Edwin (1997). *Multi-agent Coordination by Communication of Evaluations* in proceedings of the 8^{th} European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW'97 Ronneby, Sweden, pp. 63-78.

Drogoul Alexis (1993). *De la simulation multi-agent à la résolution collective de problème. Une étude de l'émergence de structures d'organisation dans les systèmes multi-agents.* Thèse à l'Université Paris VI, 214 pages.

Drogoul Alexis (2000). *Systèmes multi-agents situés.* Mémoire d'Habilitation à Diriger des Recherches, Université Paris VI, 117 pages.

Drogoul Alexis, Ferber Jacques (1994). *Multi-agent simulation as a tool for modeling societies : application to social differentiation in ant colonies.* Artificial Social Systems, vol. 830, n ° 8, pp. 3-23.

Drogoul Alexis, Picault Sébastien (1999). *Microbes : vers des robots socialement situés.* In Journées Francophones sur l'Intelligence Artificielle Distribuée et les Systèmes Multi-Agents (JFIADSMA'99), Ile de la Réunion, France, pp. 265-278.

Durkheim, Emile (1995). *The Teaching of Morality in Primary Schools*. Journal of Moral Education, 24, #1

Elfassy Solange (1999). *La Pauvreté et la Motivation Humaine*. Rapport sur le Programme MAG 97/008 du PNUD sur la Réduction de la Pauvreté et Promotion de Modes d'Existence Durables : Cellule de Coordination et d'Exécution, Octobre, env. 40 pages.

Erceau Jean, Chaudron Laurent, Ferber Jacques, Bouron Thierry (1994). *Systèmes Personne(s)-Machine(s) : Patrimoines cognitifs distribués et mondes multi-agents, coopération et prise de décision collectives.* In Systèmes coopératifs : de la modélisation à la conception. (Eds. Bernard Pavard) Octares Editions, pp. 119-152.

Ferber Jacques (1996). *Reactive Distributed Artificial Intelligence : Principles and Applications*, in Foundations of Distributed Artificial Intelligence (eds. G. M. P. O'Hare and N. R. Jennings), Wiley, pp. 287-314.

Ferber Jacques (1997). *Les systèmes Agents, Vers une intelligence collective*, Inter-Editions, 521 pages.

Ferguson Innes (1992a). *TouringMachines : Autonomous Agents with Attitudes*. In IEEE Computer, 25 (5), may.

Ferguson Innes (1992b). *TouringMachines : An Architecture for Dynamic, Rational, Mobile Agents*. PhD thesis, Clare Hall, University of Cambridge, UK. (Also available as Technical Report No. 273, University of Cambridge Computer Laboratory).

Feyreisen Pierre, De Lannoy Jacques-Dominique (1987). *L'Éthologie humaine*. Paris : Presses universitaires de France. 125 pages.

Freud Sigmund (1923). *Le moi et le ça* in Essais de psychanalyse, Payot, Paris, 1981.

Georgeff Michael, Ingrand Francois Felix (1989). *Decision-making in an Embedded Reasoning System*. In Proceedings of the 11^{th} International Joint Conference on Artificial Intelligence (ICJAI), Detroit, MI, pp. 972-978.

Gershenson Carlos, González Pedro Pablo, Negrete Jose Martinez (2000). *Action Selection Properties in a Software Simulated Agent.*, in Cairó et. al. (Eds.) MICAI 2000 : Advances in Artificial Intelligence. Lecture Notes in Artificial Intelligence 1793, Springer-Verlag, pp. 634-648.

Glaser Norbert, Morignot Philippe (1997). *The Reorganization of Societies of Autonomous Agents* in proceedings of the 8^{th} European Workshop on Modelling Autonomous Agents in a Multi-Agent World, (MAAMAW'97) Ronneby, Sweden, pp. 98-111.

Grosz Barbara J, Kraus Sarit (1996). *Collaborative plans for complex group actions* in Artificial Intelligence 86, pp.269-357.

Guerrin François, Courdier Remy, Calderoni Stéphane, Paillat Jean-Marie, Soulié Jean-Christophe, Vally Jean-Dany (1998). *Biomas : un modèle multi-agents pour aider à la gestion négociée d'effluents d'élevage.* In Modèles et systèmes multi-agents pour la gestion de l'environnement et des territoires (N.Ferrand, Ed.), Actes de colloques, Cemagref éditions, Clermont-Ferrand (F), pp. 359-378.

Guillot Agnès, Meyer Jean Arcady (1998). *Synthetic Animals in Synthetic Worlds.* In Kunii et Luciani (Eds.) *Cyber Worlds.* Springer Verlag, pp.111-123.

Haddadi Afsaneh, Sundermeyer Kurt (1996). *Belief-Desire-Intention Agent Architectures* in Foundations of Distributed Artificial Intelligence (eds. G. M. P. O'Hare and N. R. Jennings), Wiley, pp. 169-185.

Hill David (1993). *Analyse Orientée Objet et Modélisation par Simulation.* Addison-Wesley, 1993.

Hoffman Edward (1996). *Future Visions : The unpublished papers of Abraham Maslow* Thousand Oaks : Sage Publications. International Educational and Professional Publisher, Thousand Oaks London, New-Dehli, 221 pages.

Immelmann Klaus (1990). *Dictionnaire de l'éthologie.* Traduit de l'allemand par Anne Ruwet, Bruxelles ; Liège : Pierre Mardaga, 296 pages.

Jennings Nicholas R (1993) : *Specification and Implementation of a Belief-Desire-Joint-Intention Architecture for Collaborative Problem Solving*, Int. Journal of Intelligent and Cooperative Information Systems, 2 (3), 1993, 289-318.

Jennings Nicholas R (1996). *Coordination Techniques for Distributed Artificial Intelligence*, in Foundations of Distributed Artificial Intelligence (eds. G. M. P. O'Hare and N. R. Jennings), Wiley, pp. 187-210.

Kiss George (1992). *Variable coupling of agents to their environment : combining situated and symbolic automata*. In Decentralized Artificial Intelligence 3 (E. Werner and Y. Demazeau, eds.), Elsevier/North-Holland, Amsterdam, pp. 231-248.

Kitano Hiroaki, Tadokoro Satoshi, Noda Itsuki, Matsubara Hitoshi, Takahashi Tomoichi, Shinjou Atsuhi and Shimada Susumu (1999). *Robocup Rescue : Search and Rescue in Large-Scale Disasters as a Domain for Autonomous Agents Research*. In Proceedings of IEEE International Conf. on Systems, Man and Cybernetics (SMC-99), Vol. VI, october, Tokyo, pp. 739-743, 1999.

Kitano Hiroaki., Tambe Milind, Stone Peter, Coradeschi Silvia, Matsubara Hitoshi, Veloso Manuela, Noda Itsuki, Osawa Eiichi, Asada Minoru (1997). *The robocup synthetic agents' challenge*. In International Joint Conference on Artificial Intelligence (IJCAI'97).

Kodjabachian Jérôme, Christophe Corne, Meyer Jean-Arcady (2000). *Evolution of a Robust Obstacle-Avoidance Behavior in Khepera : A comparison of Incremental and Direct Strategies*. Robotics and Autonomous Systems. In Press.

Kurihara Satoshi, Aoyagi Shimegi, Onai Rikio (1997). *Adaptive Selection of Reactive/Deliberate Planning for the Dynamic Environment, - A Proposal and Evaluation of MRR-planning* -. In proceedings of the 8^{th} European Workshop on Modelling Autonomous Agents in a Multi-Agent World, (MAAMAW'97) Ronneby, Sweden, pp. 112-127.

Lahaie Franz, Grasso Jean-Robert, Marcenac Pierre, Giroux Sylvain (1996). *Modélisation de la dynamique auto-organisée des éruptions volcaniques : application au comportement de la fournaise, Réunion.* Comptes-rendus de l'Académie des Sciences de Paris. Géophysique interne, France, vol. 830, II.a, pp. 569-574

Larousse (1998). *Le Petit Larousse Illustrée*. Paris, France.

Larousse (2000). *Dictionnaire analogique*. Sous la Direction de Niobey Georges, Paris, France.

Lhuillier Marc (1998). *Une approche à base de composants logiciels pour la conception d'agents : Principes et mise en œuvre à travers la plate-forme Maleva*. Thèse de doctorat de l'université Paris 6.

Maes Pattie (1991). *A Bottom-Up Mechanism for Action Selection in an Artificial Creature.* From Animals to Animats : Proceedings of the Adaptive Behavior Conference '91, edited by S. Wilson and J. Arcady-Meyer, MIT Press, February, pp. 238-246.

Malec Jacek (2000). *On Augmenting Reactivity with Deliberation in a Controlled Manner.* Workshop on Balancing Reactivity and Social Deliberation in Multi-Agent Systems at the 14th European Conference on Artificial Intelligence (ECAI), Berlin, Germany, Lecture Notes in AI, volume 2103, Springer, pp. 76-91.

Marcenac Pierre (1997). *Modélisation et Simulation par Agents : Application aux Systèmes Complexes*, Habilitation à Diriger des Recherches, Université de La Réunion.

Marcenac Pierre, Giroux Sylvain (1998). *GEAMAS : A Generic Architecture for Agent-Oriented Simulations of Complex Processes.* Applied Intelligence, Vol 8, N ° 3, may, pp. 247-267.

Maslow Abraham (1998). *Toward a Psychology of Being, 3rd Edition.* Edited by Lowry Richard, J.Wiley & Sons, Inc, November 1998, 320 Pages.

Maslow Abraham H. (2000). *The Maslow business Reader*, edited by Collins Deborah. S, J.Wiley & Sons, Inc. 320 pages.

Mavromichalis Vangelis Kourakos, Vouros George (2000). *ICAGENT : Balancing between Reactivity and Deliberation.* In Workshop on Balancing Reactivity and Social Deliberation in Multi-Agent Systems at the 14th European Conference on Artificial Intelligence (ECAI), Berlin, Germany, Lecture Notes in AI, volume 2103, Springer, pp. 53-75.

Meyer Jean Arcady, Guillot Agnès (1991). *Simulation of Adaptive Behavior in animats : Review and Prospect.* In : From animals to animats. Proceedings of the First Conference on Simulation of Adaptive Behavior. Meyer J.A., Wilson S.W. (eds). MIT Press, USA ; pp. 2-14

Moulin Bernard, Chaib-Draa Brahim (1996). *An overview of Distributed Artificial Intelligence*, in Foundations of Distributed Artificial Intelligence (eds. G. M. P. O'Hare and N. R. Jennings), Wiley, pp. 3-55.

Moulin Bernard, Sahli Nabil, Maamar Zakaria (2001). *Utilisation d'agents mobiles et stationnaires pour la planification d'itinéraires dans un environnement dynamique.* In

proceedings of Journées Francophones sur l'Intelligence Artificielle Distribuée et les Systèmes Multi-Agents (JFIADSMA'01), 12-14 novembre, Montréal, Québec, Canada (Hermès Eds.).

Müller Jörg P. (1994). *A conceptual model of agent interaction.* In Deen, S. M., editor, Draft proceedings of the Second International Working Conference on Cooperating Knowledge Based Systems (CKBS-94), DAKE Centre, University of Keele, UK, pp. 389-404.

Newell Allen (1982). *The knowledge level.* Artificial Intelligence. 18(1), 87-127.

Nuttin Joseph (1985). *Théorie de la motivation humaine : du besoin au projet d'action.* Presses Universitaires de France PUF (Coll. Psychologie d'aujourd'hui), 383 pages.

Panzarasa Pietro, Norman Timothy, Jennings Nicholas R. (1999). *Modeling Sociality in the BDI Framework*, in J. Liu and N. Zhong (Eds.), Intelligent Agent Technology : Systems, Methodologies, and Tools. Proceedings of the 1st Asia-Pacific Conference on Intelligent Agent Technology, Hong-Kong World Scientific Publishing, pp. 202-206.

Picard Dominique (1998). *Politesse, Savoir-vivre et Relations sociales,* Que sais-je ? Presses Universitaires de France (puf), 122 pages.

Pollack Martha E., Ringuette Marc (1990). *Introducing the Tileworld : Experimentally Evaluating Agent Architectures.* In AAAI 90, pp. 183-189.

Quinqueton Joël, Hamadi Youssef (1999). *Communication et émergence : une épidémie chez les termites.* In Journées Francophones sur l'Intelligence Artificielle Distribuée et les Systèmes Multi-Agents (JFIADSMA'99), 8-10 novembre 1999, Saint-Gilles, Ile de la Réunion, (Hermès Eds.), pp. 225-235.

Rakotondraompiana Solofo, Randrianarison Tahina, Collet Claude (2001). *Caractérisation des paramètres géométriques et spatiaux des zones soumises A l'érosion hydrique. Cas d'un bassin versant des hautes terres de Madagascar.* In IXé Journées scientifiques de la télédétection : Yaoundé – Cameroun (30 Novembre – 4 Décembre).

Rao Anand, Georgeff Michael (1992). *An abstract architecture for rational agents.* In Proceedings of 3^{rd} International Conference on Principles of Knowledge Representation and Reasoning (KR & R-92), (Nebel B., Rich C., Swartout W., eds.) Morgan Kaufmann Publishers, Inc. pp. 439-449.

Renck Jean-Luc, Servais Véronique (2002). *L'éthologie : Histoire naturelle du comportement*. Edition du Seuil, Janvier, 337 pages.

Robert (2001). *La Robert pour tous*. Paris, France

Sadek David (1994). *Attitudes mentales et fondements du comportement coopératif.* In Systèmes coopératifs : de la modélisation à la conception. (Eds. Bernard Pavard) Octares Editions. pp 93-117

Searle John (1983). *Intentionnality : an Essay in the Philosophy of Mind*, New York, Cambridge University Press.

Seow Kiam Tian, How Khee Yin (2002). *Collaborative Assignment : A Multiagent Negotiation Approach Using BDI Concepts*. In proceedings of the International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS' 2002) July 15-19, Bologna, Italy, pp. 256-263.

Simonin Olivier, Ferber Jacques (2001). *Modélisation des satisfactions personnelle et interactive d'agents situés coopératifs*. In 9eme journées Francophones d'Intelligence Artificielle Distribuée et Systèmes Multi-Agents (JFIADSMA'2001), 12-14 novembre, Montréal, pp. 215-226.

Simonin Olivier, Michel Fabien, Chapelle Jérôme, Ferber Jacques (2002). *Un simulateur de systèmes multi-robots dans MadKit*. In 10eme journées Francophones d'Intelligence Artificielle Distribuée et Systèmes Multi-Agents (JFIADSMA'2002), 28-30 octobre, Lille, pp. 167-170.

Soulié Jean Christophe (2001). *Vers un modèle multi-environnements pour les agents*. Thèse à l'Université de La Réunion.

Sullivan David, Grosz Barbara, Kraus Sarit (2000). *Intention Reconciliation by Collaborative Agents.* In 4^{th} International Conference on Multi-Agent Systems (ICMAS 2000), Boston USA, IEEE Computer Society Press, pp. 293-300.

Tambe Milind (1996). *Executing Team Plans in Dynamic Multi-agent environments.* AAAI Fall Symposium on Plan Execution. Boston USA.

Tambe Milind, Adibi Jafar, Al-Onaizan Yaser, Erdem Ali, Kaminka Gal A., Marsella Stacy C., Muslea Ion (1999). *Building Agent Teams Using an Explicit Teamwork Model and Learning* in Artificial Intelligence, volume 110, pp 215-240

Tinbergen Nikolaas (1951). *The study of Instinct*. Claredon Press. Trad. Fr. *L'Etude de l 'instinct*. Payot, 1971

Todes Daniel (2001). *Pavlov's physiology factory : Experiment, Interpretation, Laboratory Enterprise*. The Johns Hopkins University Press, november 2001, 512 pages.

Tyrrell Toby (1993). *The Use of Hierarchies for Action Selection*. Adaptive Behavior, 1(4), pp. 387-420.

Wehrle Thomas (1994). *New Fungus eater experiments*. In P. Gaussier, & J.-D Nicoud (Eds.), From Perception to action. Los Alamitos : IEEE Computer Society Press, pp. 400-403.

Werner Eric (1996). *Logical foundations of Distributed Artificial Intelligence*, in Foundations of Distributed Artificial Intelligence (eds. G. M. P. O'Hare and N. R. Jennings), Wiley, pp. 57-117.

Wooldridge Michael (2002). *An Introduction To Multiagent System*, Published by John Wiley & Sons, March, 340 pages.

Wooldridge Michael, Jennings Nicholas R. (1995). *Agent Theories, Architectures, and Languages : A Survey*. In Intelligent Agents, Workshop on Agent Theories, Architectures, and Languages, ATAL-94, M. Wooldridge & N. R. Jennings eds, Springer-Verlag, Lecture Notes in Artificial Intelligence (LNAI), Vol. 890, Heidelberg, Germany, pp. 1-39.

Zeghal Karim (1994). *Vers une théorie de la coordination d'actions – Application à la navigation aérienne*. Thèse de Doctorat, Université Pierre Marie Curie (Paris VI) France.