Thèse de doctorat de l'Université de Paris VI
Université Pierre et Marie Curie

Spécialité Informatique

Mickaël Camus

pour obtenir le titre de Docteur de l'Université de Paris VI

Système auto-adaptatif générique pour le contrôle de robots ou d'entités logicielles

7 septempbre 2007

Composition du Jury :
Pr. Joël Colloc : Rapporteur
Pr. Khaled Ghedira : Rapporteur
Pr. Jean-François Perrot : Examinateur
Pr. Peter Van Roy : Examinateur
Mme Zahia Guessoum MdC HDR : Examinateur
M. Nabil El Kadhi : Examinateur
Pr. Alain Cardon : Directeur de thèse

Situation et Remerciement

Cette thèse a été réalisée grâce à un contrat à durée indéterminée que j'ai obtenu à l'Epitech, une école d'expertise informatique formant des experts en cinq ans après le bac. Durant ma thèse, en ma qualité d'enseignant chercheur, j'ai enseigné la programmation Kernel (BSD, Linux), l'intelligence artificielle, l'administration système (Système V, BSD), la sécurité informatique (pour les premières années) et le C/UNIX (pour les premières années). J'ai eu aussi la possibilité d'enseigner le Oz/Mozart et de faire publier des étudiants de cinquième année dans une conférence internationale.

Mes recherches ont eu cour au Laboratoire Informatique de Paris 6 (LIP6 UMR CNRS 7606), ainsi qu'au laboratoire de recherche de l'Epitech, le L.E.R.I.A. (Laboratoire Epitech de Recherche en Informatique Appliquée), dont je suis un des cofondateurs. Dans ce laboratoire, j'ai effectué mes recherches en qualité de directeur adjoint.

Je remercie sincèrement le Professeur Alain Cardon de m'avoir permis de travailler à ses côtés. Ce fut très enrichissant et passionnant. J'espère de tout cœur que notre travail ouvrira des perspectives importantes sur les recherches avenirs.

Je remercie les rapporteurs de mon mémoire de thèse M. Joël Colloc et M. Khaled Ghedira pour leurs commentaires constructifs et leurs connaissances dans le domaine de mes recherches.

Je remercie le jury pour toutes ses questions constructives, montrant le grand intérêt porté à mon sujet de recherche, Mme Zahia Guessoum, M. Jean-François Perrot, M. Peter Van Roy et M. Nabil El Kadhi.

Je remercie aussi Monsieur Nicolas Sadirac, le directeur de l'Epitech, qui m'a fait confiance toutes ces années en me permettant de former des étudiants et en finançant tous les déplacements concernant les conférences scientifiques.

Je remercie également Monsieur Fabrice Bardèche, directeur général de Ionis Education Group, un groupe privé d'enseignement supérieur dont l'Epitech fait partie. Son intérêt pour nos recherches montre son ouverture d'esprit et l'importance qu'il donne aux évolutions concernant l'informatique.

Un grand merci aussi à tous les étudiants qui ont contribué à ce projet. J'ai travaillé avec eux avec grand plaisir. Leurs compétences leur permettront un avenir professionnel fort prometteur.

Merci à ma famille et à mes amis qui ont toujours cru en ma valeur et qui ont respecté tous les choix que j'ai faits. Merci à tous.

Table des matières

I Les systèmes complexes sous le paradigme multi-agent 11

1 Mise en place de la problématique et système proposé 13
1.1 Introduction 13
1.2 Hypothèses à valider 14
1.3 Architecture systémique 14

2 Une science de la complexité 17
2.1 Les systèmes complexes 17
2.1.1 Définitions 17
2.1.2 Comportements 17
2.2 La théorie du chaos 18
2.2.1 Les systèmes dynamiques 18
2.2.2 Système chaotique 19
2.3 Recherches actuelles 19
2.3.1 Les protocells 19
2.3.2 Réseau pensant 20
2.3.3 Le complexe : la relation des sciences 21

3 Les systèmes autonomes 25
3.1 Théorie Génerale 25
3.2 Cybernétique : La synergie de la robotique et de l'IA 27
3.2.1 Apprentissage pour le comportement 27
3.3 Systèmes embarqués 28
3.3.1 Architecture d'un système embarqué 29
3.3.2 Les besoins d'autonomie des systèmes spatiaux 29
3.3.3 Design et architecture 30
3.4 Autonomie et neuroscience 36
3.4.1 Mémoire et associativité 37

4 Les systèmes adaptatifs et auto-adaptatifs 39
4.1 Définitions 39
4.2 Propriétés 39
4.3 Systèmes existants représentatifs 40

4.3.1 Projet SMARTHERM 40
4.3.2 Projet Lumière 42
4.3.3 Projet Astrolab 43
4.3.4 Bilan de ces recherches 43

II Modélisation et fonctionnement du système auto-adaptatif générique 45

5 La boucle systémique 47
5.1 Fonctionnement général 47
5.2 Activation des connaissances 47
5.3 Génération et représentation de scènes 47
5.4 Vers un résultat non optimal mais satisfaisant 49

6 Modèle proposé 53
6.1 Les ontologies 53
6.1.1 Description ontologique 54
6.1.2 Traitement dynamique de l'ontologie 54
6.1.3 Activités et Intentions 55
6.2 Emotions et représentation des émotions 56
6.2.1 Description des émotions 58
6.2.2 Traitement dynamique des émotions 59
6.3 Approche SMA : création et organisation d'agents 60
6.3.1 L'agent et sa proactivité 60
6.3.2 Mise en action d'un agent dans le système 63
6.3.2.1 Liste de threads 64
6.3.2.2 Création de threads et envoi de messages 64
6.3.2.3 Design pattern 66
6.4 Les organisations du Système Multi-Agent 66
6.4.1 Organisation générique agent 68
6.4.2 Organisation structurante 69
6.4.3 Architecture générale du système 70
6.4.3.1 Approche 70
6.4.3.2 Architecture 71
6.4.4 Organisation de morphologie 74
6.4.5 Organisation d'analyse 76
6.4.6 Synergie des Types 77
6.4.7 Synergie des Rôles 78
6.5 Morphologie et formes géométriques 78
6.5.1 Notion de morphologie et stabilité structurelle 79
6.5.2 Définition de la morphologie par R. Thom 79
6.5.3 Utilisation de la morphologie pour le contrôle du SMA . . 80
6.5.3.1 Notion de projection 80

6.5.3.2 Formes géométriques et métriques 81
6.6 Le contrôle du système : l'organisation des objectifs 82

7 Implémentation du système 85
7.1 Introduction 85
7.2 Matériel utilisé : l'Aibo de Sony 85
7.2.1 Capteurs et effecteurs 86
7.2.2 Vers une autonomie adaptative 86
7.2.3 Vision réaliste de l'autonomie à court terme 86
7.3 Langage utilisé 87
7.4 Réalisation du système 87
7.4.1 Modèle conceptuel de donnée pour l'ontologie 87
7.4.2 Classes et messages 89
7.4.2.1 Gestion des senseurs et des effecteurs 89
7.4.2.2 Création des entités pour groupes de robots . . . 92
7.4.2.3 Distribution de l'ontologie et paramétrage des rôles 96
7.4.2.4 Communication interne et types de messages . . 97

8 Utilisation du système 103
8.1 Système et options d'utilisation 103
8.2 Les ontologies 104
8.3 Monitoring 105

III Réalisations et validation des hypothèses 113

9 Les différentes expériences et leurs résultats 115
9.1 Composition de l'ontologie pour les expérimentations 116
9.1.1 La classification 116
9.1.2 Les connaissances 117
9.1.3 Les expériences mémorisées 117
9.1.4 Les accointances 117
9.2 Les différentes expériences et leur utilité 117
9.3 Première expérience : activité émergente 118
9.3.1 Objectifs et hypothèse 118
9.3.2 Méthodes d'application 118
9.3.3 Monitoring et analyse 119
9.4 Deuxième expérience : processus émotionnel 129
9.4.1 Objectif et hypothèse 129
9.4.2 Approche 129
9.4.3 Objectifs du scénario 130
9.4.4 Caractéristiques émotionnelles 130
9.4.5 Accointances 130

9.4.6 Monitoring et analyse 130
9.5 Troisième expérience : fusion de deux systèmes auto-adaptatifs . . 132
9.5.1 Objectif et hypothèse 132
9.5.2 Approche 133
9.5.3 Monitoring et analyse 134
9.6 Quatrième expérience : modification des paramètres en temps-réel 136
9.6.1 Objectif et hypothèse 136
9.6.2 Approche 137
9.6.3 Monitoring et analyse 137
9.7 Cinquième expérience : génération de code html 141
9.7.1 Objectif et hypothèse 141
9.7.2 Approche 142
9.7.3 Analyse du résultat 144

10 Développements en cours : acquisition on-line de connaissances nouvelles 147
10.1 Connaissances et apprentissage 147
10.1.1 Constat d'un élément non connu 148
10.1.2 Situation spatiale et temporelle d'un élément non connu . 149
10.1.3 Rapprochement d'un élément non connu à des éléments connus 150
10.1.4 Application d'expériences à un élément non connu 150
10.1.5 Test de besoin sur un élément non connu pour atteindre un objectif 151
10.1.6 Identification temporaire 151
10.1.7 Nommage 152
10.1.8 Action et vérification du bon déroulement de l'action entreprise 152
10.2 Développements 152

11 Conclusion et perspectives 155
Table des figures 158
Bibliographie 163

IV Annexes 169

A Définitions de l'autonomie 171

B Définitions de la morphologie 175

C Les logiciels développés 179
C.1 Le noyau de transcription ontologique 179
C.1.1 Présentation 179
C.1.2 Description 179

C.2 Morphoviewer 181
C.2.1 Présentation 181
C.2.2 Description 181

D Les ontologies utilisées 183

E Le Oz/Mozart 189
E.1 Présentation 189
E.2 Caractéristiques 189
E.3 L'environnement de développement 190
E.4 Le Oz Kernel Language 190
E.5 Les types 192
E.5.1 Initialisation 193
E.6 Les différents paradigmes supportés 193
E.6.1 Type de développement 193
E.6.2 Les spécificités du langage 193
E.6.3 Scripting 194
E.6.4 Les modules 195
E.7 Le processing 197
E.8 La programmation orienté objet 198
E.8.1 Les classes 198
E.8.2 Les objets 199
E.8.3 La syntaxe 200
E.8.4 L'héritage 201
E.9 La programmation logique 201
E.9.1 Les disjonctions 201
E.9.2 Les conditions 202
E.9.3 Les programmes non déterministe 202
E.10 La programmation par contraintes 203
E.11 Les contraintes et les domaines finis 204
E.12 La propagation de contraintes 204
E.13 Application en Oz 204
E.14 Oz Explorer 205

Résumé

Français

Les besoins concernant l'autonomie et la flexibilité sont grandissants. Les systèmes de contrôle et d'information se doivent d'être de plus en plus génériques, adaptables et autonomes afin d'augmenter au mieux la valeur ajoutée d'un groupe travaillant dans différents secteurs comme l'aéronautique, l'aérospatiale, l'automobile, le ferroviaire, la robotique, la gestion ou l'aide à la décision. Cette thèse propose un modèle de système auto-adaptatif et défend le fait qu'une entité autonome adaptative fonctionne selon une boucle systémique à organisation complexe, en évoluant dans un environnement non stable. Ces recherches se basent sur l'informatique avec notamment les systèmes multi-agents et la robotique, les neurosciences et la composition du système nerveux, les systèmes complexes, ainsi que la psychanalyse pour toute la modélisation de l'appareil psychique humain. La solution proposée est un système auto-adaptatif pour le contrôle de groupes d'entités hétérogènes. Ce système est un planificateur dynamique de processus qui permet la programmation par morphologie en se basant sur des liaisons entre des connaissances et des objectifs en général complètement intégrés dans ces liaisons. Les expérimentations ont montré que le système s'organise de bonne manière afin de respecter les objectifs par entités ou par groupes. Elles ont aussi montré que le système est totalement générique dans le cas où l'ontologie et les objectifs sont modifiés. Ce système auto-adaptatif permet de prendre en compte de manière différente la plupart des problématiques du traitement de l'information en ce qui concerne les logiciels de prise de décision, les systèmes d'exploitation, la robotique, mais aussi d'amener de nouveaux outils pour l'étude du comportement humain dans différents milieux. Un point essentiel de ce système est qu'il constitue une base computationnelle de la conscience artificielle qui permettra d'évoluer vers un système informationnel homogène générant des représentations pour son compte et complètement adapté à l'environnement de communication humain.

Anglais

There is an increasing need for autonomy and flexibility in industry. Information systems and control systems need to be more generic, adaptable, and autono-

mic, to better support industries working in multiple sectors (such as aeronautics, aerospace, automobiles, railways, robotics, management, and decision support). This dissertation proposes a model for self-adaptive systems using global feedback loops that can operate in hostile environments. This research draws on many domains including computer science, multi-agent systems, robotics, neuroscience, complex systems, as well as psychanalysis for modeling the human psyche. The proposed solution is a self-adaptive system for controlling groups of heterogeneous entities. The system is a dynamic process planner that uses morphology programming based on links between knowledge items and the goals that are implicit in these links. Experiments show that the system self-organizes in a satisfactory manner to respect the objectives of entities or groups. Experiments also show that the system is completely generic when the ontology or the goals are modified. This self-adaptive system allows to take into account in a novel way many of the problems encountered in information processing regarding decision support, operating systems, and robotics. It also proposes new tools for the study of human behavior in different environments. An essential aspect of the system is that it is a computational foundation for an artificial "consciousness" that can be evolved into a global information system that is completely adapted to the human environment.

Première partie

Les systèmes complexes sous le paradigme multi-agent

Chapitre 1

Mise en place de la problématique et système proposé

1.1 Introduction

Les systèmes informatiques gèrent de multiples évènements totalement asynchrones, du fait de l'instabilité du monde réel qu'ils appréhendent, aboutissant à une action précise à réaliser. Cette gestion d'évènements est possible grâce à la composition du processeur d'un ordinateur qui est scalable et qui possède une horloge interne. Ces caractéristiques permettent de dire qu'un programme informatique s'exécute dans un environnement qui n'est pas statique, et qui est même en constante évolution. Il en est de même pour les systèmes industriels qui héritent de ces caractéristiques. Un système d'information qui pourrait s'auto-adapter à une situation permettrait une réutilisation du logiciel pour différentes problématiques très économiques. Un tel système serait capable de générer des algorithmes en temps-réel en fonction de différents objectifs passés en paramètre. Ceci permettrait de travailler au niveau de la connaissance pour les exécutifs, en laissant de coté la partie technique de la programmation ainsi que les problèmes de compatibilité de systèmes.

Cette thèse présente un modèle de système auto-adaptatif complexe permettant de planifier des processus de manière dynamique afin de s'adapter en temps réel à la situation courante précise. Celui-ci permet de travailler au niveau connaissance défini par A. Newell [New82] sans être obligé de reprogrammer chaque activité comportementale à chaque fois. Pour ce faire, nous ferons un état de l'art des systèmes autonomes ainsi que des systèmes adaptatifs complexes développés dans l'industrie ou dans la communauté scientifique. Ensuite, nous détaillerons la mise en place du modèle que nous proposons, son implémentation et l'utilisation du logiciel en découlant. Pour terminer, nous exposerons une conclusion ouvrant sur les possibilités que permet l'utilisation d'un tel système.

1.2 Hypothèses à valider

Il s'agit ici d'un problème d'architecture et de modélisation. La plupart des applications de l'intelligence artificielle classique utilisent des moteurs d'inférences et une base de règles avec modification dynamique afin d'accélérer les traitements en cours et futurs. Le problème avec ce genre de technique est que le temps de la recherche augmente avec la taille de la base de règles. L'idée générale de cette thèse est de permettre à la connaissance de se manifester de manière autonome sans avoir à parcourir une base de connaissances. De ce fait, le traitement de l'information se fait instantanément.

Voici les hypothèses soutenues dans cette thèse :

- Nous posons qu'un système complexe peut être réalisé grâce à un système multi-agent possédant une organisation interne spécifique.
- Nous posons qu'une entité autonome ne peut pas être représentée par un simple agent possédant une base de connaissances et un moteur d'inférence, une multitude d'agents est nécessaire.
- Nous posons qu'il est possible de contrôler un nombre massif d'agents autonomes ayant des rôles différents.
- Nous posons qu'un système pourra s'adapter à un environnement seulement avec une analyse morphologique de son organisation interne.
- Nous posons qu'un système multi-agent est capable de s'auto-modifier de manière continue.
- Nous posons qu'il est possible de planifier une nuée de processus en temps-réel.

L'étude des systèmes autonomes permettra à la société d'évoluer intelligemment vers une disparition progressive des problèmes techniques et des problèmes de compatibilité, laissant place à la réflexion et à la gestion de la connaissance qui sont des activités plus constructives pour l'homme.

1.3 Architecture systémique

L'architecture globale du système fonctionne de manière systémique c'est à dire que les données d'un environnement sont interprétées pour le modifier. Une fois l'environnement modifié, les données sont réinterprétées par le système (figure 1.1). Chaque décision prise et action appliquée modifie l'environnement qui est réinterprété de manière continue pour prendre de nouvelles décisions et effectuer de nouvelles actions.

La figure 1.2 montre les types d'agents présents dans l'architecture. Ceux-ci seront traités et expliqués dans le partie II concernant la modélisation et le traitement du système. Voici ces différents types d'agents :

- Les agents structurants qui contiennent toutes les connaissances de l'onto-

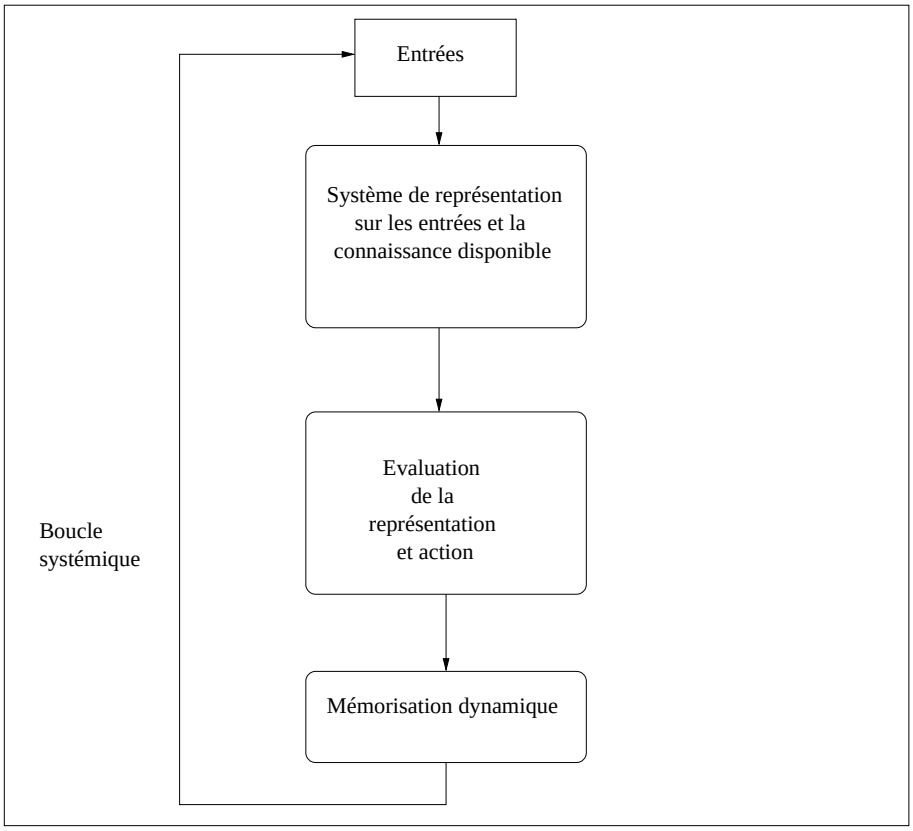

FIG. 1.1 – Boucle systémique

logie et génèrent les activités internes pour produire la représentation de la situation.

- Les agents de morphologie qui génèrent des statistique sur les communications des agents structurants.
- Les agents d'analyse qui représentent sous forme de graphe toutes les statistiques des agents de morphologie.
- Le Pattern d'Evènement Spécifique (PES), un composant qui choisit un ou des nœuds du graphe envoyé par les agents d'analyse en fonction des objectifs donnés au système et permet de réaliser l'émergence.

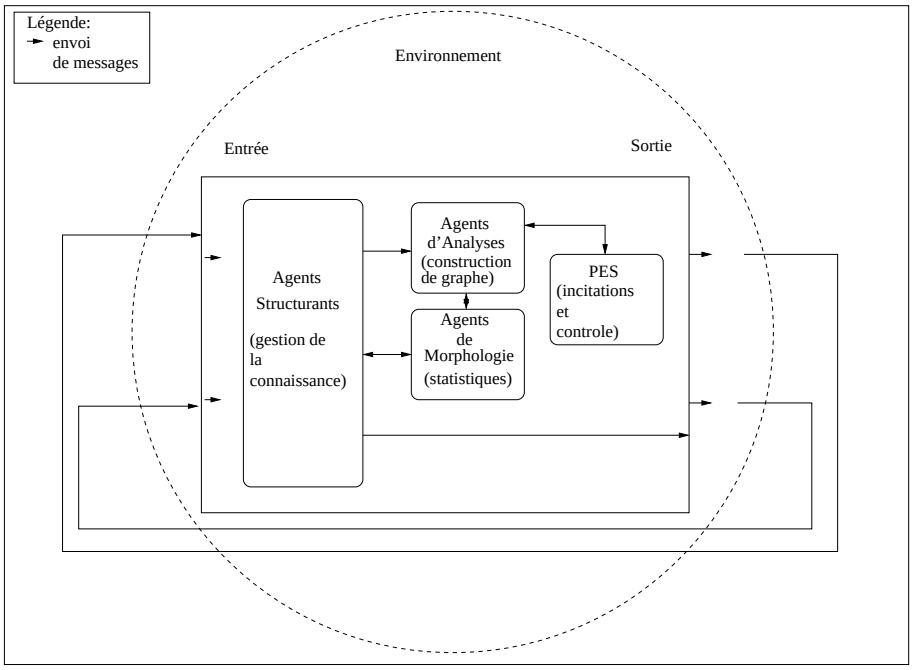

FIG. 1.2 – Architecture systémique

Chapitre 2

Une science de la complexité

2.1 Les systèmes complexes

Un système est dit complexe (définition de M. Gell-Mann dans [GM95]) lorsqu'il est constitué d'un grand nombre d'entités en interaction et qu'un observateur ne peut pas prévoir son comportement ni son évolution par un raccourci de calcul [Rit06] [otUoC06]. Un système complexe est composé "d'un grand nombre de parties qui interagissent de manière qui n'est pas simple" [Sim01]. Certaines réactions chimiques sont décomposables en quelques équations et le résultat sera donc connu à l'avance. Ceci n'est pas le cas concernant les cellules nerveuses dans le cerveau et leur activation en fonction des phénomènes se déroulant dans l'environnement de l'organisme qui l'héberge. Ce dernier point est très important pour la suite de cette thèse car il lie la complexité à l'être humain, qui lui sera lié par la suite à la machine computationnelle [Wol02].

2.1.1 Définitions

Un système complexe est un système composé d'un grand nombre d'entités en interaction locale et simultanée dont le but est d'exprimer et de contrôler les phénomènes émergents [aES03]. Il n'existe pas de définition formelle d'un système complexe, mais un tel système aura les caractéristiques suivantes :

- Un graphe d'interaction non trivial, c'est à dire qui n'est pas forcément fortement connecté.
- Des interactions locales, tout comme les informations transitant dans le système.
- Le système est en contact continuel avec un environnement extérieur. Ce contact est délimité et le système ne peut pas se fondre dans l'environnement.

2.1.2 Comportements

Un système complexe possède des comportements spécifiques le caractérisant. Le comportement est crucial dans ce type de système car la constitution des en-

tités du système n'est pas forcément connue [Car04] [Gol99] [See99]. Voici les caractéristiques comportementales :

- Auto-organisation et émergence des propriétés ou des structures cohérentes et apparition de motifs représentant une certaine forme de connaissance.
- Robustesse locale et fragilité à moyenne échelle. Un nombre très important de liens sont créés et modifiés en temps-réel par le système lui-même. Si une entité est affectée par un évènement extérieur, alors les entités ayant un lien avec cette entité seront aussi affectées. Plus le système possède de liens, moins il est robuste et seulement une faible perturbation permettra de changer son orientation.
- Brisure de symétrie, c'est à dire la connaissance d'une partie du système ne permet pas d'affirmer que le reste du système est en moyenne dans le même état.
- Plusieurs comportements sont en compétition dans le système, certains sont simples, d'autre chaotiques. Le système est souvent à la frontière entre les deux et alterne de manière non planifiée ces deux types de comportements.
- Plusieurs échelles temporelles et spatiales sont présentes, il y a donc des hiérarchies de structures.

2.2 La théorie du chaos

La théorie du chaos est une théorie physique qui traite des systèmes dynamiques déterministes présentant un phénomène fondamental d'instabilité appelé "sensibilité aux conditions initiales" découvert par Poincaré à la fin du 19ème siècle. On peut trouver une explication de cette théorie dans [Gut90] ou encore [Ott02] et [Moo90]. En associant ce phénomène à une propriété de récurrence, les systèmes ne sont pas prédictibles sur le long terme. Voici le théorème du retour de Poincaré : *pour presque toutes les "conditions initiales", un système dynamique conservatif dont l'espace des phases est de "volume" fini va repasser au cours du temps aussi près que l'on veut de sa condition initiale, et ce de façon répétée*.

2.2.1 Les systèmes dynamiques

Que ce soit en mathématique, en physique théorique ou en ingénierie, un système dynamique est un système classique, c'est à dire qui utilise les théories de la physique classique [All97] qui évoluent au cours du temps de manière :

- causale : l'avenir du système ne dépend que des phénomènes du passé ou du présent,
- déterministe : pour une condition initiale donnée à un instant présent va correspondre à chaque moment ultérieur un et un seul état futur possible.

Une système dynamique évolue donc dans le temps [Spr03]. Cette évolution déterministe peut se modéliser de deux façons distinctes :

1. Une évolution continue dans le temps.

2. Une évolution discontinue dans le temps, qui est beaucoup plus simple à représenter mathématiquement car chaque élément représentant la discontinuité est représentable.

La première manière de faire est la plus réaliste physiquement parlant car pour nous, être humain, le temps ne s'arrête pas et tout se passe de manière parallèle, c'est à dire qu'à chaque instant t, est associé une suite de phénomènes ayant des définitions relatives au contexte.

2.2.2 Système chaotique

Une système chaotique est en fait un système dynamique qui est chaotique [Dev03]. Un tel système est dit chaotique si une portion significative de son espace des phases[1] présentes simultanément les deux caractéristique suivantes :

- le phénomène de sensibilité aux conditions initiales,
- une forte récurrence.

Ces deux propriétés provoquent un comportement extrêmement désordonné qualifié de "chaotique".

2.3 Recherches actuelles

De nombreux organismes de recherche en science informatique s'orientent vers les systèmes complexes et la théorie du chaos afin de trouver une logique de fonctionnement à l'être humain et à la vie en général. Le Santa Fe Institute se démarque un peu des autres organismes de recherche car il axe ses recherches depuis plus de vingt ans maintenant sur la science de la complexité. D'autres laboratoires prestigieux comme le CSAIL du MIT et l'article de Rodney Brooks concernant le relation entre la matière et la vie [Bro01] ou le Complex System Lab de l'Université de Pompeu Fabra en Espagne, prennent cet axe de recherche très au sérieux. Ils remettent en cause certaines affirmations et démonstrations concernant la biologie humaine [Bro01] et étudient la logique de fonctionnement des organismes, humains [SMRM06] ou végétaux.

2.3.1 Les protocells

Depuis de longues années, certains chercheurs essayent de trouver le moyen de construire un système pouvant s'adapter à plusieurs situations, devenir hétérogène, spatialement organisé et pouvant atteindre des états homogènes permettant une évolution constante dans un environnement. C'est le cas de A. Turing, qui a publié en 1952 un article concernant la base chimique de la morphogénèse. Cet article explique de manière mathématique l'organisation dans le temps des différents

[1] L'espace des phases est un espace abstrait dont les coordonnées sont les variables dynamiques du système étudié.

éléments permettant la morphogénèse [Tur52].

L'article de Turing est une base de travail pour Solé et Macia du Santa Fe Institute et du Complex System Lab pour construire des cellules synthétiques ayant un comportement similaire aux cellules d'origine en fonction de certaines réactions chimiques [MS06]. Ces chercheurs ont travaillé sur plusieurs constantes afin de simuler les réactions chimiques, ainsi que les conséquences de celles-ci sur les cellules. Voici un extrait des paramètres utilisés :

- La constante Kinétique.
- La perméabilité.
- La conductivité hydraulique.
- Le coefficient de diffusion.
- La constant de déplacement.
- Le coefficient d'élasticité.
- Le taux de dégradation.

Au cours des simulations, les chercheurs ont réussi à définir un modèle cellulaire minimal proposant un métabolisme interne permettant de créer une instabilité dans le sens où Turing l'entendait sur son étude de la morphogénèse [Tur52]. Les figures 2.1 et 2.2, reprises de l'article de Macia et Solé, montrent en premier lieu le modèle de base de la cellule synthétique avec ses deux types de molécules, $g1$ et $g2$ pour ensuite montrer l'évolution du profil de concentration de ces mêmes molécules. A chaque séparation, il y a création d'une autre concentration. Une concentration est liée à un type de molécules. Ces recherches montrent que la complexité de niveau cellulaire peut être automatisable à un certain niveau.

2.3.2 Réseau pensant

M. Mitchell, de l'Université de Portland, aborde un sujet pluridisciplinaire très important dans [Mit06]. Le "réseau", de par sa définition générale, est un outil commun à toutes les sciences. Seulement les connaissances et les traitements spécifiques diffèrent selon le contexte d'utilisation. Pour Barabàsi, une nouvelle forme de science voit le jour, elle est basée sur la connexion et l'interaction d'éléments entre eux [Bar02].

La théorie des graphes joue dans ce domaine un rôle très important. Un réseau est un ensemble de noeuds et de liens et la méthode de création de ces noeuds et de ces liens fait toute la différence au niveau de l'organisation générale du système complexe. Les liens permettent d'accéder à des éléments pour accéder à d'autres éléments et ainsi de suite. L'application d'une organisation spécifique à un réseau entraîne un métabolisme interne particulier. Il existe plusieurs organisations pouvant être compatibles au fonctionnement d'un réseau :

- L'automate cellulaire [Wol02].
- Le système immunitaire [SC01].
- Le système nerveux [SK02].

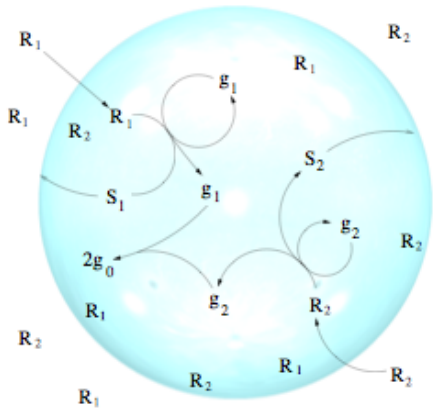

FIG. 2 The basic protocell model considered in this paper. It involves the presence of a membrane γ together with precursors ($\mathcal{R}_1, \mathcal{R}_2$) and two basic types of molecules (the morphogens g_1 and g_2) which interact and diffuse through the (two-dimensional) space, defining a minimal, spatially-extended metabolic network.

FIG. 2.1 – Modèle de base d'une cellule synthétique

- Les colonies de fourmis [BDT99].
- Le métabolisme des cellules biologiques [SC01].

La synthèse de ces articles mène à la superposition des structures et des traitements en commun sans tenir compte de l'information elle-même, de la connaissance, le constat étant que la transposition est possible en travaillant au niveau Méta. Comme le précise M. Mitchell, tous les évènements et les différents problèmes pouvant apparaître sont des problèmes liés à un réseau, une liaison de connaissances pouvant amener à une solution.

2.3.3 Le complexe : la relation des sciences

R. Brooks se pose certaines questions dans [Bro01] concernant l'intelligence artificielle (IA) et la vie artificielle (VA). Il existe certainement une relation entre eux comme la matière et la vie, mais certains paramètres manquent pour mettre en place une théorie globale. Le but de l'IA de la VA est de construire un système computationnel en s'inspirant des différents aspects de la vie. Un grand nombre de paramètres venant de connaissances scientifiques multiples doivent être pris en compte. Cela rejoint le points de vu de S. Wolfram dans son livre "A New Kind of Science" [Wol02] dans lequel il aborde un grand nombre de science telles les mathématiques, la physique, la biologie, les sciences sociales, la science informatique, la philosophie, l'art, la technologie, l'intelligence artificielle, la vie artificielle, la théorie des catastrophes, la théorie du chaos, la théorie de la complexité, la cybernétique, la théorie de l'évolution et bien d'autres, cette liste n'étant pas exhaustive. Mais un autre chercheur s'est intéressé à cette relation, du moins du coté des ma-

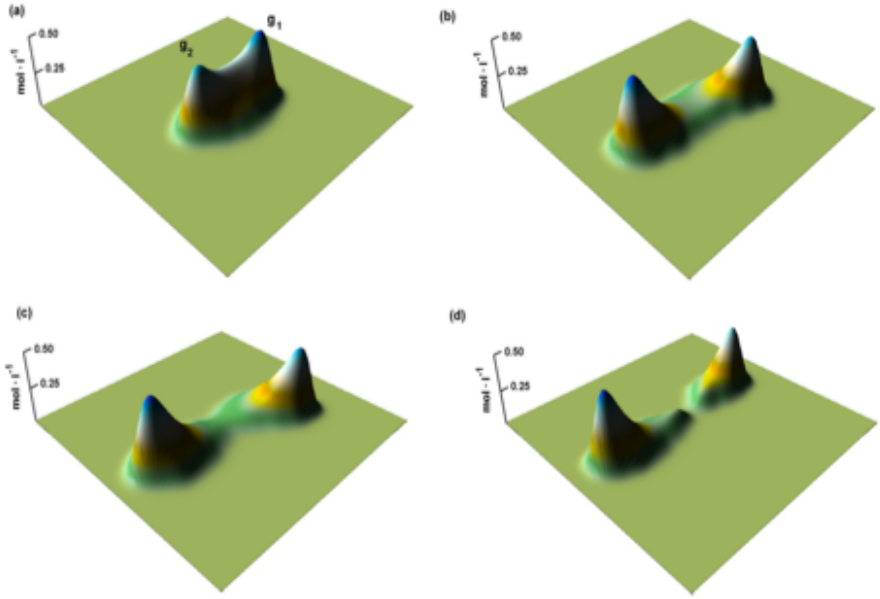

FIG. 5 Evolution of the concentrations profiles of g_1 and g_2 coupled with the membrane expansion process. Simulation parameters from table I. Here we can see that after a transient, two peaks emerge (a) indicating two maximal concentrations of g_1 and g_2. As the simulation proceeds, the peaks separate (b-c) as the membrane (not shown) gets deformed. In (d) we shaw the two concentration profiles right after cell splitting.

FIG. 2.2 – Evolution du profil de concentration des deux types de molécules (Source : Santa Fe Institute).

thématiques et de la biologie, il a découvert la théorie des catastrophes, R. Thom [Tho99]. Ce chercheur pensait que les biologistes n'utilisaient pas les bonnes méthodes de travail pour comprendre le fonctionnement de la vie. Il a mis beaucoup de temps pour mettre en place les outils mathématiques permettant la création de liens entre certaines expérimentations. Il apparaît ici que la pluridisciplinarité est cruciale pour la compréhension de la vie en général. A. Turing, dans les années cinquante, avait compris cela en abordant une théorie de base concernant la morphogénèse [Tur52]. Ce dernier point nous ramène aux études faites au Santa Fe Institute au Nouveau Mexique concernant des cellules synthétiques [SMRM06] [MS06]. La fin des années soixante-dix a aussi marqué l'avancer déjà existante dans la synergie des sciences avec la publication en 1979 du livre de D. R. Hofstadter [Hof99]. Cet ouvrage tisse un lien entre la logique mathématique, l'intelligence artificielle, la musique de Bach et les dessins de Maurits Cornelis Escher [Esc71]. Les déductions amenées dans ce livre proposent aux lecteurs une vision synergique entre la manière de faire de l'homme et sa manière de fonctionner liée à sa composition (matière, cellulaire) en faisant abstraction du sujet auquel appartient la chose créée par celui-ci.

R. Brooks amène quatre propositions dans [Bro01] concernant l'inexistence actuelle de système computationnel copiant le fonctionnement de l'être humain :

1. Certains paramètres, considérés comme bons, sont peut être erronés.
2. Des modèles basés sur les sciences de la complexité doivent être construits.
3. Il existe peut être un problème de puissance au niveau des micro-ordinateurs.
4. Quelque chose de fondamentale a peut être été oublié, ou bien n'est pas encore imaginé, dans le modèle biologique existant.

La suite de cette thèse montrera que la troisième proposition n'est pas un problème. Les systèmes actuels sont assez puissants pour copier la méthode de fonctionnement de l'être humain. Les deuxième et quatrième propositions sont vraies, cette thèse vise à éclairer ces points avec la proposition d'un modèle. La première proposition peut être fondue avec la quatrième, la mauvaise utilisation de certains paramètres mène à de mauvaises pistes. A. R. Damasio [Dam95] [Dam05] et J. C. Eccles [Ecc93] amènent beaucoup d'arguments et de déductions scientifiques concernant le fonctionnement de l'être humain de manière neurobiologique, ce qui fournit des éléments de réponses pour la quatrième proposition.

Chapitre 3

Les systèmes autonomes

Les professionnels demandent aux systèmes industriels une autonomie de plus en plus importante. C'est le cas de l'ESA - European Space Agency, et plus précisément de l'ESTEC - European Space research and Technology Center, en ce qui concerne l'autonomie des satellites et des sondes spatiales. Ce besoin d'autonomie est apparu en même temps que le projet de *myriades* [Tho01], se composant de cinq micro-satellites ayant une capacité d'autonomie et de partage d'informations grandement supérieures à celles d'un satellite standard. Il a engendré une demande de plusieurs études concernant les algorithmes nécéssaires pour effectuer une telle tâche. La société d'ingénierie *Axlog* passe une partie du temps consacré à sa recherche et développement sur ce probléme. Plusieurs axes ont déjà été abordés comme la programmation par contraintes pour la planification [GPB99] [CP01] et les systémes multi-agents en ce qui concerne l'aspect social.

L'ESA n'est pas le seul organisme à avoir de plus en plus besoin d'autonomie dans ses systèmes. La Défense Européenne a commandé une étude pour la gestion de l'autonomie des drônes, MISURE - MIssion management System for Uninhabited aiR vEhicles and new techniques [Pon02]. Le but de ce logiciel est d'automatiser toute la gestion des missions des drônes et de donner à chacun d'eux une méthode de prise de décision pour la suppression du facteur humain.

L'intéret croissant que portent ces organismes internationaux à l'autonomie des systèmes donne une importance encore plus grande à la science informatique et à l'électronique. Actuellement, l'autonomie en informatique et en électronique n'est pas définie, chaque industrie et laboratoire ont leur propre définition et essayent de se rapprocher le plus possible de celle-ci au niveau de la technique.

3.1 Théorie Génerale

Bien que l'autonomie ne possède pas de définition claire au niveau de la science informatique, ce n'est pas le cas dans d'autres sciences. L'encyclopédie "Le Trésor

de la Langue Française" [CAdN02] donne une définition de l'*autonomie* que l'on peut voir en Annexe A.

Le mot "autonomie " est utilisé dans différents contextes et différents environnements, il est partagé par de nombreuses disciplines. Dans le cas de la science informatique, l'autonomie est plutôt orientée vers l'autonomie d'une personne. Celle-ci pense, décide, agit, s'adapte, crée, tant de choses que la science informatique aimerait reproduire en excluant les points négatifs comme la destruction des peuples, de l'environnment, la soumission des pays les plus démunis. Bien entendu, tous les êtres de notre monde ne sont pas sans scrupules et effectuent des tâches pour combattre ce système comme par exemple la formation des pays pauvres aux nouvelles technologies pour qu'ils se construisent une valeur ajoutée. Cette bonne conscience peut elle être reproduite ? Différents chercheurs comme Alain Cardon ont déjà abordé très largement le problème [Car00]. Faire penser une machine n'est pas une utopie selon Alain Cardon [Car04]. Marvin Minsky, chercheur renommé en intelligence artificielle, travaille sur le fonctionnement de l'esprit humain pour essayer de le reproduire. Pour Minsky, l'esprit prend la forme d'une société complexe et adaptative [Min88]. Les recherches de ces deux chercheurs sont complémetaires bien que la méthode d'approche soit différente. L'autonomie se fera aussi en étudiant le fonctionnement du cerveau et de l'esprit humain. Il faut savoir maintenant comment mesurer l'autonomie d'un système et dans quelle mesure on peut le considérer comme "Autonome".

La définition de l'autonomie est très éphémère, elle change en fonction du secteur et des besoins. Cependant, deux types d'autonomie peuvent être catalogués : l'autonomie **contrôlée** et l'autonomie **adaptative**. Le premier type d'autonomie est énormément utilisé dans l'industrie d'aujourd'hui. Elle permet de profiter des aspects techniques de l'intelligence artificielle comme les bases de règles et les moteurs d'inférences, les réseaux de neurones, la planification [1] et communication sociale d'une entité à une autre [2]. Toutes ces techniques aident des secteurs comme l'aéronautique, l'aérospatiale, le ferrovière, la défense à orienter leurs technologies vers l'autonomie. L'autonomie adaptative est une tout autre voie, la philosophie la concernant est à l'opposé de l'autonomie contrôlée. Il n'existe pas d'objet spécifique dans le programme, c'est à dire que pour une problématique donnée, aucun composant logiciel strictement fonctionnel n'est développé. L'ensemble du programme doit s'adapter à n'importe quelle problématique. De cette manière, une entité sera capable d'étendre ses capacités en peu de temps et sans gros investissement du coté du développement de logiciel.

[1] Celle-ci a été énormément simplifiée et optimisée grâce à la programmation par contraintes.
[2] Cette communication reste très basique comparé à une discusion humaine.

3.2 Cybernétique : La synergie de la robotique et de l'IA

Cela fait plusieurs années maintenant que les scientifiques, les écrivains et les cinéastes s'interressent aux robots [Asi72]. L'industrie travaille énormément avec la robotique, qu'elle soit automobile, pharmaceutique, agro-alimentaire, elle lui permet d'augmenter sa valeur ajouté et sa flexibilité. L'industrie de très haute technologie du type de l'aérospatiale demande de plus en plus d'utilisation spécifique et d'autonomie à la robotique [BJ04] [Dav04]. Cela permet un aménagement du temps et des conditions de travail plus souples, sans compter la capacité physique des employés ainsi que leurs horaires. La création d'un laboratoire commun entre la France et le Japon en novembre 2001 [CNR01b] prouve que la robotique se modernise de plus en plus et commence à devenir une haute technologie très demandée. La vision future de certains auteurs [Ald69] et cinéastes [KS01], en relation étroite avec certains scientifiques, montre une évolution exponentielle de la robotique et de l'intelligence artificielle mettant en scène des entités copiant l'apparence physique et le comportement social des êtres humains. La recherche en cybernétique, liant la robotique avec des algorithmes de très haut niveau utilisés en intelligence artificielle, permettra d'arriver à de telles performances comportementales et émotionnelles.

3.2.1 Apprentissage pour le comportement

Depuis plusieurs années les chercheurs essayent de construire un robot imitant le comportement humain. On entend par comportement humain, une capacité de communication sociale avec un veritable être humain. Pour réaliser cela, la synergie entre la robotique et l'intelligence artificielle formant la cybernétique, a développé différentes méthodes afin de copier mentalement et physiquement l'être humain. C'est le cas du robot Leonardo du Media Lab au MIT [BBG+03].

Ce robot est aussi une vision artistique et cinématographique de la cybernétique. Celui-ci est capable d'écouter, d'analyser et d'agir en fonction du contexte. Ce robot effectue une suite d'actions complexes programmées pour agir dans un contexte particulier. Sorti de son environnement d'apprentissage, celui-ci n'est pas capable de s'adapter, un nouvel apprentissage doit être effectué. En restant dans son environnement, Leonardo est capable de réaliser des choses surprenantes comme écouter une personne en la regardant, et faire ce que celle-ci lui demande [1]. Ainsi, quand un être humain demande à Leonardo d'appuyer sur le buzzer numero cinq se trouvant devant lui, il regarde attentivement la personne lorsqu'elle pose la question, et ensuite, il appuie sur le bon buzzer. Un gros travail a été fait sur le comportement et les mimiques du robot pour le rendre le plus naturel possible.

[1] Ceci limite beaucoup le robot car il doit prendre connaissance du contexte avant d'agir. Cela ne suit pas la philosophie du "Etre Soi" du Professeur Alain Cardon et du Professeur Marvin Minsky qui ne permet pas une représentation statique de l'esprit.

En mettant l'accent sur le comportement du robot, les chercheurs mettent en évidence ce que pourrait donner la synergie entre le travail d'un robot et celui d'un humain. Dans le futur, certaines actions industrielles seront très dangereuses pour qu'un être humain puisse s'y risquer. C'est le cas atuellement pour les missions spatiales et le danger encouru par les astronautes pour des tâches à effectuer qui sont répétitives. Dans cette optique, la NASA [1], en collaboration avec la DARPA [2], travaille sur un robot pouvant agir en même temps que les astronautes : le Robonaute [BAD+03]. De la même manière que Leonardo, le robonaute a des tâches bien précises à effectuer. Certaines tâches demanderont un apprentissage pour une reconnaissance d'images, en analysant le tableau de bord de la navette spaciale par exemple, pour ensuite décider d'actionner telle ou telle fonctionnalité. L'action du robonaute, comme celui de Leonardo, est limitée à ce qu'il a appris. L'aspect adaptatif n'est pas abordé, on peut donc parler d'autonomie controlée. Le rôle du robot restera très technique, sa conception peut paraître rudimentaire comparée à Leonardo, mais le but n'est pas du tout le même et peu importe, actuellement le coté artistique pour la NASA.

Dans le secteur de la robotique, la recherche Japonaise possède une avance technologique significative. Techniquement, les robots peuvent marcher, sauter, courir. La seule chose manquante se trouve dans la communication sociale et la "construction du Soi" pour qu'il y ait survie, et donc, adaptation. Il est possible à la recherche d'orienter ces robots vers une autonomie. Celle-ci se rapprocherait plus du contrôle que de l'adaptation car la recherche sur les systèmes adaptatifs concernant l'autonomie dans le secteur industriel reste très limitée. L'industrie Japonaise mise énormément sur le marché des robots autonomes, elle veut être leader dans ce domaine et se donne les moyens de réussir en essayant, notamment, de se mettre en relation avec l'industrie Française pour s'insérer plus facilement en Europe [Jac04b]. La recherche et l'industrie Française essaye de rattraper un grand retard sur la robotique grâce au débloquage de fonds de la part de gouvernement donnant de nouveaux moyens au CNRS et à l'INRIA qui ont pu travailler sur des projets novateurs. Ces moyens ont donné naissance à un robot marcheur et coureur bipède, développé sous le programme de Robea [3] [CNR01a], et montrant les compétences françaises en terme de robotique [Jac04a]. A part l'aspect technique de la marche ou de la course, l'autonomie, ici, reste très réduite.

3.3 Systèmes embarqués

L'autonomie est d'une importance capitale pour le monde spatial. Elle va lui permettre d'effectuer des économies d'échelle importantes en écartant l'être hu-

[1] National Aeronautics and Space Administration.
[2] Defense Advanced Research Projects Agency.
[3] Ce nouveau programme lancé par le CNRS doit promouvoir la cybernétique française.

main des prises de décisions basiques. Ceci étant, l'aspect adaptatif[1] des systèmes autonomes offriront un plus majeur à la gestion des problèmes non connus aujourd'hui, et pour la plupart, non encore élucidés.

3.3.1 Architecture d'un système embarqué

Les systèmes spatiaux possèdent une architecture adaptée, non seulement aux besoins des clients, mais aussi aux contraintes de fonctionnement. La figure 3.1 présente la composition de cette architecture [Rio02].
Un système spatial possède deux composantes fondamentales : le satellite ou la sonde spatiale et la station au sol. Au niveau du véhicule spatial, l'architecture est composée de plusieurs modules. Le plus important d'entre eux étant le Payload, la charge utile, ce qui correspond au matériel que va utiliser le véhicule une fois qu'il sera en orbite. Cette charge utile va contenir les trois instruments maximum (caméra, transpondeur, antennes ou amplificateur) que peut supporter un véhicule spatial. Chaque instrument possèdera plusieurs sous-ensembles représentant sa composition.
La station au sol, quant à elle, permet la transmission avec le véhicule spatial. Le contrôle du véhicule se fera donc au sol, ainsi que celui des instruments embarqués. Toutes ces actions ne peuvent pas être exécutées en même temps. C'est pour cette raison qu'il existe une plannification afin que le véhicule sache la suite logique d'actions à réaliser pendant un certain laps de temps.

3.3.2 Les besoins d'autonomie des systèmes spatiaux

De part cette conception, un satellite ou une sonde spatiale possède des besoins bien particuliers afin que l'autonomie puisse y prendre sa place[2]. Les besoins concernant l'autonomie sont multiples[BP97] :

- profiter des avantages que la science améne au niveau de la technologie, comme par exemple un spectromètre à ultra-violets qui prè-planifie ses séquences d'observation, et les stocke sur la carte embarquée.
- réduire les coûts des opérations des satellites et des sondes spatiales, ce qui permettrait au satellite d'utiliser plus de ressources en même temps, et si ce n'est pas le cas, d'avoir plus de puissance de calcul.
- assurer la robustesse des opérations effectuées après un certain nombre d'erreurs, le but étant d'effectuer les actions permettant de prévenir les fautes.

[1] Certaines personnes préfèreront séparer l'aspect autonomie de l'aspect adaptatif. Ces deux aspects sont traités de manière différente au niveau de la recherche scientifique. L'un utilise des variables dites de bas niveau afin d'y effectuer un traitement décisionnel, l'autre met à jour une base de connaissances pour augmenter sa capacité de résolution. Ces deux aspects sont pourtant indisociables pour une entité autonome.

[2] Le terme "insertion de l'autonomie" pourrait être remplacé par le terme "augmentation du niveau de l'autonomie" dans quelques années. Pour un informaticien travaillant sur du "haut niveau" et plus exactement sur des problématiques liées à l'intelligence artifcielle, les satellites actuels ne possèdent aucun niveau d'autonomie.

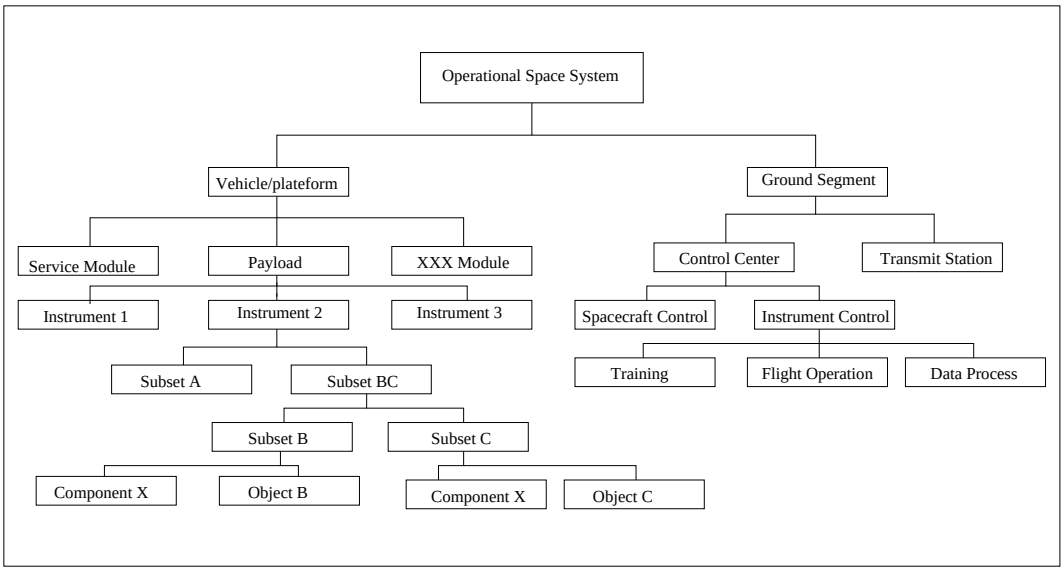

FIG. 3.1 – Architecture d'un système spatial

L'étude de ces besoins vont permettre de conceptualiser une architecture spécifique qui prendra en compte l'autonomie.

3.3.3 Design et architecture

La plupart des scientifiques et des industriels travaillant dans l'autonomie spatiale se basent sur un système multi-agent afin de modéliser une constellation de satellites ou de sondes pouvant s'entre aider en cas de problèmes ou de besoins particuliers, comme par exemple, effectuer une communication en utilisant un protocole connu d'un seul micro-satellite parmi la constellation. C'est donc ce satellite qui exécutera la tâche pour les autres. Ce type de modèle est utilisé par la société d'ingénierie Axlog pour le compte de l'**ESTEC - E**uropean **S**pace research and **TEC**hnology centre, pour le projet d'étude concernant l'autonomie des véhicules spatiaux. La **NASA**, au même titre que l'**ESA**, utilise la méthodologie agent pour accéder à l'autonomie [BDF+98].

Le design conceptualisé par la NASA se base sur une entité appelée "Remote Agent" [PBC+96]. Un satellite de ce type doit posséder plusieurs fonctionnalités pour être considéré comme autonome par l'agence. Voici ces fonctionnalités :

- réaliser les différents objectifs fixés,
- générer un plan d'exécution pour ces objectifs en fonction de l'état du véhicule aérien,
- détecter les erreurs d'actions et rétablir le bon fonctionnement,
- éviter les réponses prématurées afin de ne pas donner d'indications erronées,

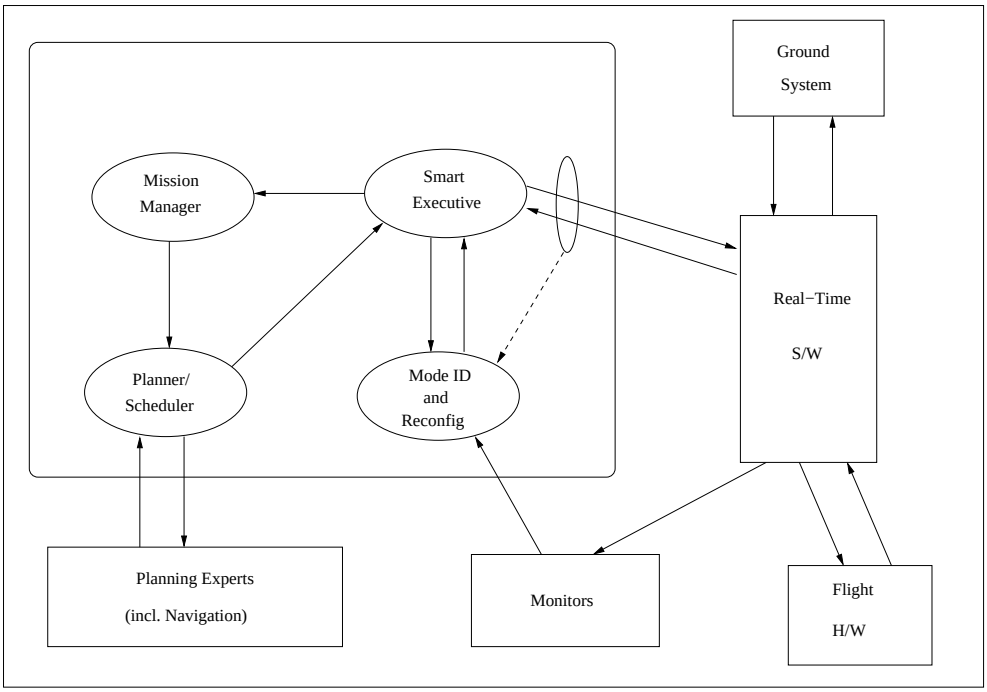

FIG. 3.2 – Architecture du “Remote Agent” (Source : NASA).

- synchroniser la partie matérielle et la partie logicielle,
- déterminer l’état de santé des modules matériels,
- effectuer une nouvelle planification en cas de changement de contexte ou d’intrusion d’erreurs,
- permettre aux commandes dites de bas niveau d’accéder au matériel,
- assurer un appel automatique à la station au sol en cas de problèmes non gérables.

Si ces actions se font automatiquement, les véhicules spatiaux pourront rester seuls, en autonomie, pendant un temps qui n’est pas mesurable actuellement, cela dépendant de nombreux paramètres qui ne sont pas tous connus.

Le “Remote Agent” possède une architecture bien particulière, la figure 3.2 montrant sa composition. Cet Agent est composé de quatre éléments essentiels : la gestion de mission, la planification, l’exécution des tâches, et la configuration. Ces éléments vont lui permettre d’exécuter des objectifs après les avoir planifiés sans attendre une demande d’un tiers extérieur. Cette architecture permet aussi à l’agent de communiquer avec la station au sol afin de traiter des demandes spécifiques.

Le modèle de l’ESA s’appuie sur une vue plus large afin d’essayer, par la même occasion, de réduire les coûts de construction d’un satellite et de son lancement.

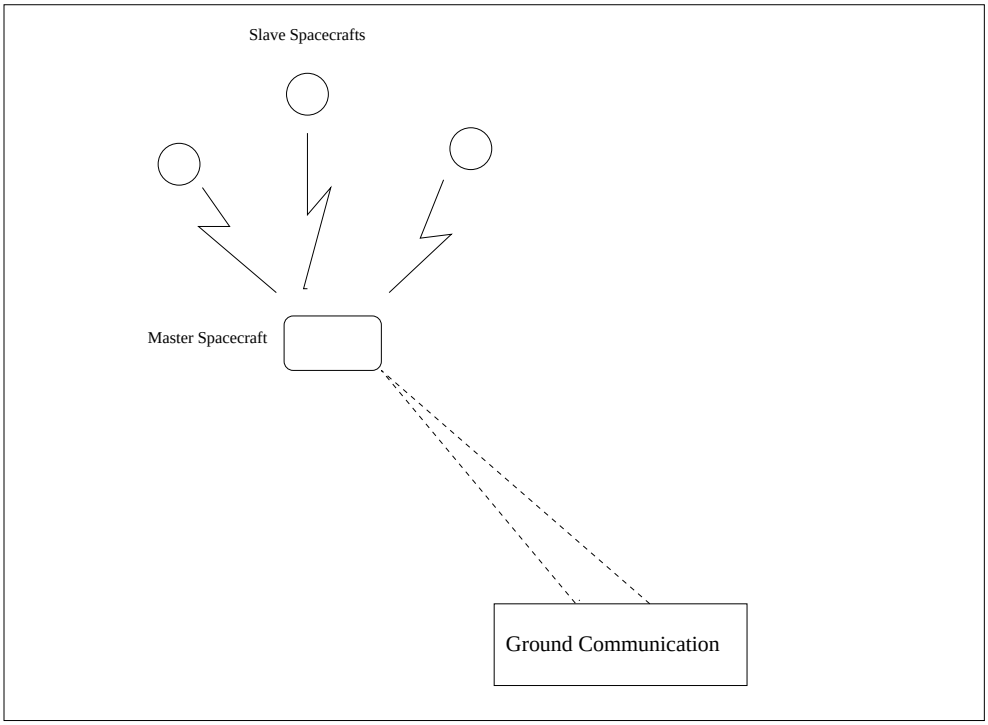

FIG. 3.3 – Hiérarchie d'une constellation de satellites

Cette vue prend la forme d'une constellation de satellites, ou plutôt, une constellation de micro-satellites, comme le montre la figure 3.3. Ce design met en place un micro-satellite maître qui partagera ses informations avec les micro-satellites sous sa coupe. Ces satellites sont dit "esclaves".

L'approche multi-agent est représentée dans la figure 3.4. L'agent maître s'occupe de la gestion de la mission que doit effectuer la constellation. C'est lui qui calculera le plan d'action en fonction de la charge utile embarquée et des objectifs à atteindre. Ce plan sera ensuite transmis aux agents soumis afin qu'une synchronisation s'opère. Pour chaque besoin, l'agent soumis lancera une requête à l'agent maître. Il attendra obligatoirement une réponse avant d'agir. Un point important à mettre en avant est que n'importe quel agent soumis pourra prendre le rôle de l'agent maître. De cette manière, si un problème majeur arrive à l'agent principal, son remplacement sera assuré. Tout ceci se fait automatiquement en action/décision.

Les fonctionnalités accrues du système amène une composition interne d'agent plus complexe que la modélisation de l'inter-communication. Un agent possède une structure sur deux niveaux comme présenté sur les figures 3.5 et 3.6 :

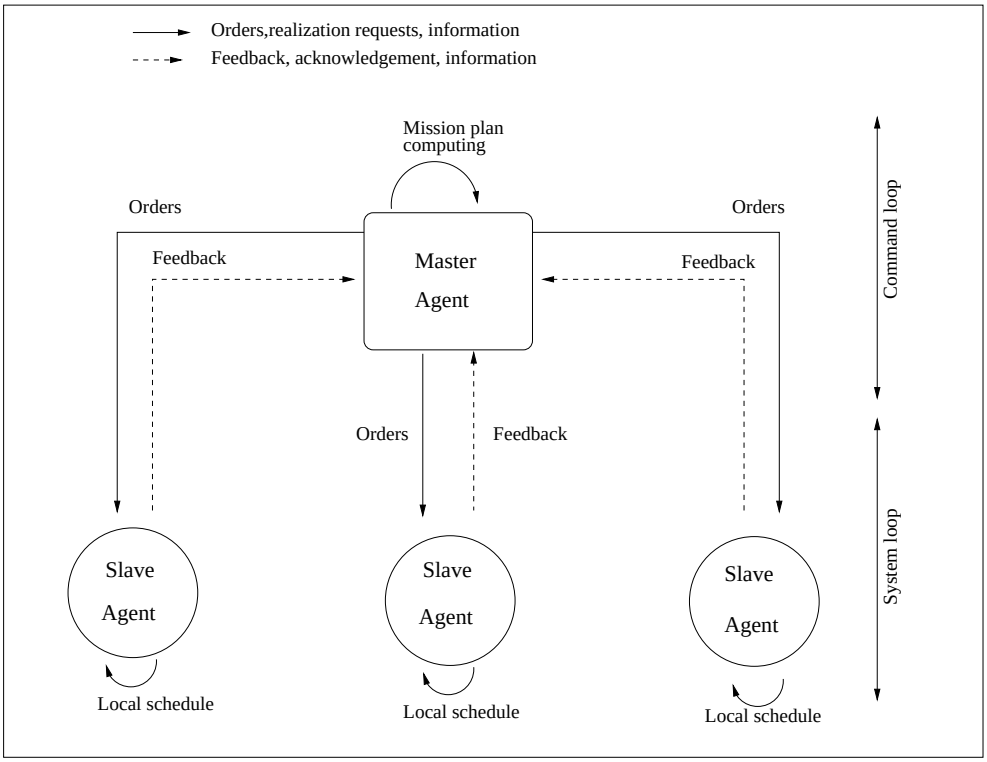

FIG. 3.4 – Approche multi-agent(Sources : Axlog Ingénierie, ESA).

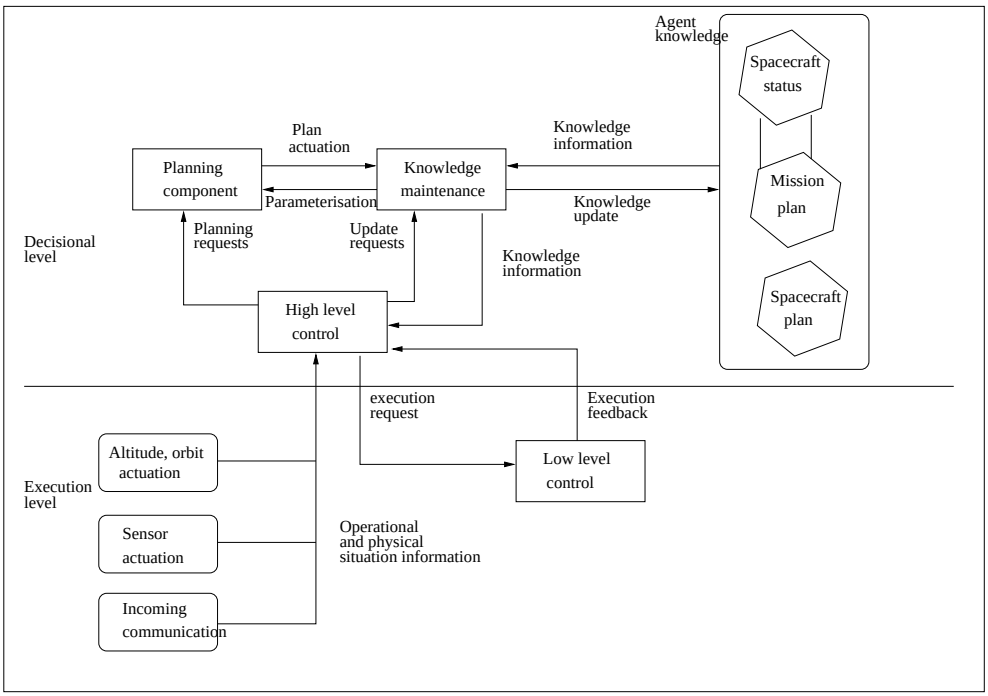

FIG. 3.5 – Architecture logicielle d'un agent (vision 1)(Sources : Axlog Ingénierie, ESA).

1. Le niveau de décision comprenant :
   - un composant de planification qui traite les paramètres de la base de connaissances pour mettre en place le plan ;
   - la maintenance de la base de connaissances pour mettre à jour les nouvelles données ;
   - un niveau de contrôle qui est en contact avec le niveau d'exécution pour lui communiquer les requêtes déduites du plan.
2. Le niveau d'exécution, comprenant aussi un niveau de contrôle qui envoie les commandes aux composants physiques et renseigne le niveau de décision sur l'acquisition de nouvelles données.

La partie planification de cette structure est très importante. C'est elle qui permet l'organisation et la bonne exécution des tâches, elle génère un plan d'action. La construction de ce plan d'action est abordée par le biais de la programmation par contraintes (PPC). La modélisation est faite selon une approche à modèles multiples qui peuvent être résolus séparément et/ou de manière concurrente. Les principaux modèles correspondent à la figure 3.7. Chacun d'entre eux est décomposé en sous-modèles de granularité plus fine attachés aux différents aspects théoriques et/ou opérationnels de la problématique. Le modèle de chemin assure le calcul de la trajectoire des différents satellites de la constellation, il est lié au modèle de temps

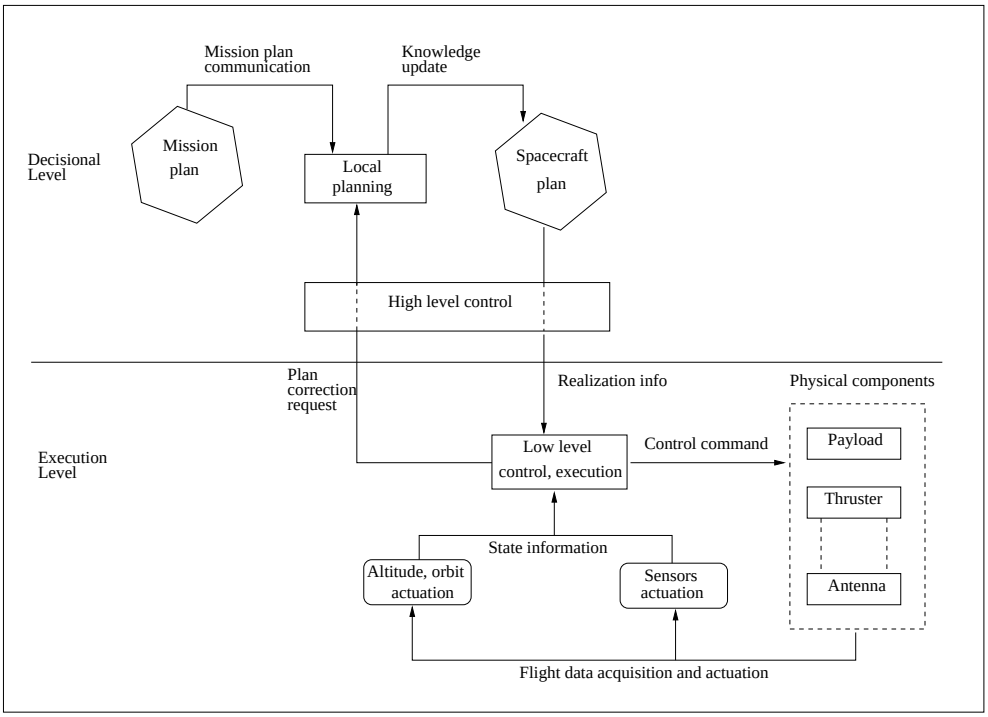

FIG. 3.6 – Architecture logicielle d'un agent (vision 2)(Sources : Axlog Ingénierie, ESA).

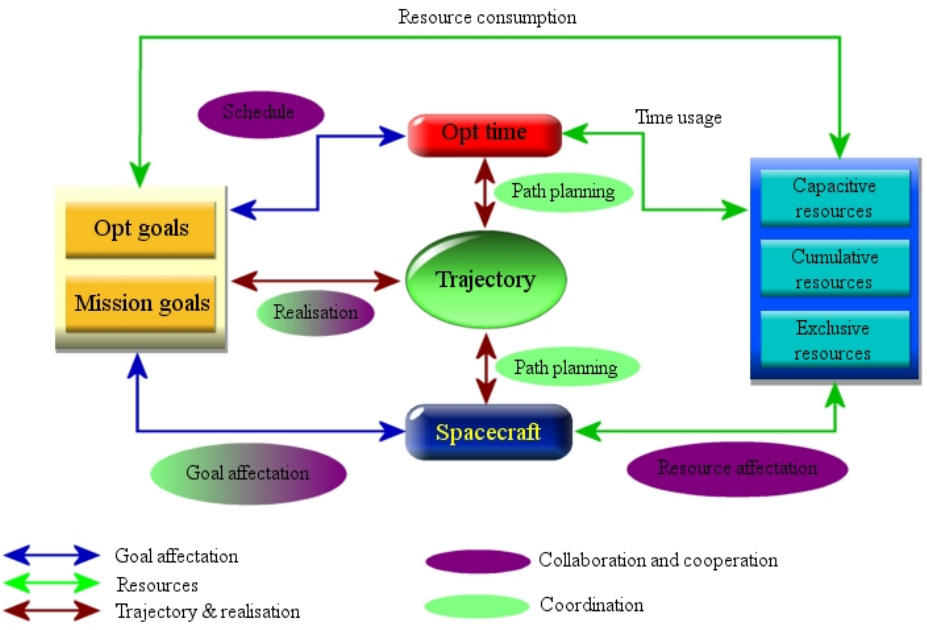

FIG. 3.7 – Modèle de planification par programmation par contraintes(Source : Axlog Ingénierie).

qui décrit les séquences d'action à haut niveau nécessaire pour la réalisation des buts de la mission. Chaque satellite possède une liste de buts à atteindre ainsi que différentes ressources. L'affectation des ressources s'effectue en fonction du temps et des buts d'une mission. Ces derniers sont affectés en fonction de la trajectoire et du temps. Enfin, le modèle de coordination permet d'assurer la faisabilité des plans locaux des agents notamment au niveau de la gestion des modes opérationnels de la plateforme et de la charge utile de chacun des satellites.

La résolution du problème de planification des trajectoires et d'affectation des buts au sein de la constellation revient à trouver des solutions à un problème NP-Difficile incluant des contraintes de ressources (notamment pour la mémoire et l'énergie disponible à bord des engins spaciaux). L'élaboration d'un plan de mission devant être aussi rapide que possible, plusieurs stratégies de résolution ont été développées mettant en oeuvre de la recherche locale, de la réparation et de l'extension de plan.

3.4 Autonomie et neuroscience

La première référence en ce qui concerne le mot "autonomie" est sans nul doute la faune et l'être humain. Il est tout à fait normal que l'on se base sur l'être humain

pour définir et matérialiser l'autonomie étant donné qu'il est doué d'intelligence et possède le plus gros rapport masse corporelle/cerveau de tous les animaux connus, présents sur la Terre.

L'autonomie est une notion liée à l'intelligence artificielle, les sciences cognitives et l'étude du cerveau humain. Tous ces termes pourraient aussi être regroupés en un seul : "la science de l'information", nouvelle matière pluridisciplinaire ayant pour objectif la copie du comportement humain afin de contruire une synergie homme/machine représentant une super intelligence [BJ02].

Afin d'effectuer un replica, il est important de connaître le mode de fonctionnement de l'être humain en ce qui concerne ses déductions, ses pensées, ses décisions. Ces trois éléments amènent logiquement à l'étude du cerveau, et plus précisément, à la neuroscience computationnelle [Tra02].

3.4.1 Mémoire et associativité

La mémoire est une partie mystérieuse du cerveau humain. Elle permet à celui-ci de ne pas recommencer ses erreurs et d'améliorer certaines de ses actions quand elles sont répétitives. C'est grâce à cette mémoire que l'être humain est capable d'apprendre des choses très compliquées pour que celles-ci apparaissent automatiquement dans le futur.

Certaines recherches médicales sur le cerveau ont montré que la matière du cerveau avait un lien direct avec l'interprétation des choses, que ce soit la vision, l'ouïe ou le toucher [KW01]. Des expériences sur l'interprétation de la vision utilisant de l'imagerie médicale ont montré que les signaux récupérés prennent une forme particulière pour chaque objet contemplé. Les cellules présentes dans le cerveau peuvent être comparée à une immense matrice composée elle même de nombreuses matrices. Chaque cellule qui s'excitera fera partie d'un chemin composé en temps réel à la manière d'un réseau d'interactions. Sauf que le but ici n'est pas de prendre une décision, mais de générer une pensée, ou un début de pensée. La précision de cette pensée dépendra de la profondeur de la matrice. Celle-ci pourra être représentée, mais seulement en quatre dimensions, et prendra la forme d'un hypercube polymorphe qui aura la particularité d'être instable.

Chapitre 4

Les systèmes adaptatifs et auto-adaptatifs

4.1 Définitions

Les systèmes adaptatifs sont aussi appelé systèmes adaptatifs complexe par la communauté anglo-saxone. A l'heure actuelle, il existe plusieurs définitions dans la communauté :

Définition de John H. Holland [Wal92] : *Un système adaptatif complexe est un réseau d'agents dynamique (qui peuvent représenter des cellules, des espèces, des individus, des nations) agissant en continue de manière parallèle et réagissant aux actes des autres agents. Le contrôle d'un système adaptatif complexe est décentralisé. Si il n'y a pas de comportement cohérent dans le système, il opère une compétition et une coopération entre les agents. Le comportement global du système est le résultat d'un grand nombre de décisions prise à chaque instant par les agents de manière individuelle.*

Définition de Kevin Dooley [Doo97] : *Un système adaptatif complexe évolue en respectant trois principes clés : l'ordre est émergent contrairement à la prédétermination, l'historique du système est irréversible, et le comportement futur du système n'est pas prédictible. La base de construction d'un système adaptatif complexe est l'agent. Les agents scannent leur environnement et développe un schéma de représentation pour interpréter ce dernier pour ensuite créer des règles d'actions. Ces schémas sont sujet à changement et à évolution dans le temps.*

4.2 Propriétés

La base d'un Système Adaptatif (SA) sera pour nous un Système Multi-Agent (SMA). La différence entre ces deux systèmes se situe au niveau des caracté-

ristiques et des propriétés de haut niveau comme la complexité, l'émergence et l'auto-organisation. Un SMA est un système composé de multiples agents interagissant entre eux. Un SA est une collectivité complexe d'agents similaires adaptatifs interagissant aussi entre eux. Les systèmes adaptatifs sont caractérisés par un haut degrés d'adaptation et de résilience face à des perturbations. Il existe d'autres propriétés importantes comme la communication, la coopération, la spécialisation, l'organisation spatiale et temporelle et la reproduction.

4.3 Systèmes existants représentatifs

4.3.1 Projet SMARTHERM

SMARTHERM est un projet de recherche et développement propre à SYLVERSOFT Inc. qui a pour but de mettre au point un système de contrôle par ordinateur de la température ambiante domiciliaire basé sur les technologies de l'intelligence artificielle. Le projet a pris naissance au début des années 90 et fonctionne en laboratoire et en résidence témoin depuis plus de 8 ans. Son processus de mise au point est relativement long et laborieux, compte tenu qu'il doit être mis à l'épreuve dans un contexte réel d'utilisation, particulièrement au cours des rigoureuses conditions hivernales qui prévalent au nord du 49ème parallèle.

Par temps froid, plusieurs phénomènes fort complexes se conjuguent pour contribuer à la perte de chaleur accumulée dans une résidence, et par le fait même de la difficulté pour tout système de chauffage d'atteindre et de maintenir un degré de chaleur donné à l'intérieur de cette résidence. Les phénomènes les plus déterminants sont :

- La température extérieure (la perte de chaleur croît de façon exponentielle avec l'abaissement de la température extérieure).
- La vitesse et la direction des vents (le facteur éolien détermine sur les corps calorifiques une température effective inférieure de plusieurs degrés à la température nominale).
- Le degré ou l'absence d'ensoleillement (l'ensoleillement contribue à accroître de façon significative la température des corps qui y sont exposés, particulièrement si ces derniers se trouvent dans un endroit hermétiquement fermé).
- La température intérieure désirée (la difficulté d'atteindre et de maintenir chaque degré supplémentaire de température ambiante à l'intérieur d'une maison croît de façon exponentielle, voir figure 4.1).
- Le facteur d'isolation de la maison (l'état de l'isolant, la superficie de fenestration, l'orientation de la maison, etc.).
- Les habitudes de vie des occupants (seuil de tolérance au froid, fréquences des entrées/sorties, ouverture/fermeture des fenêtres, ouverture/fermeture des rideaux ou stores vénitiens, etc.).

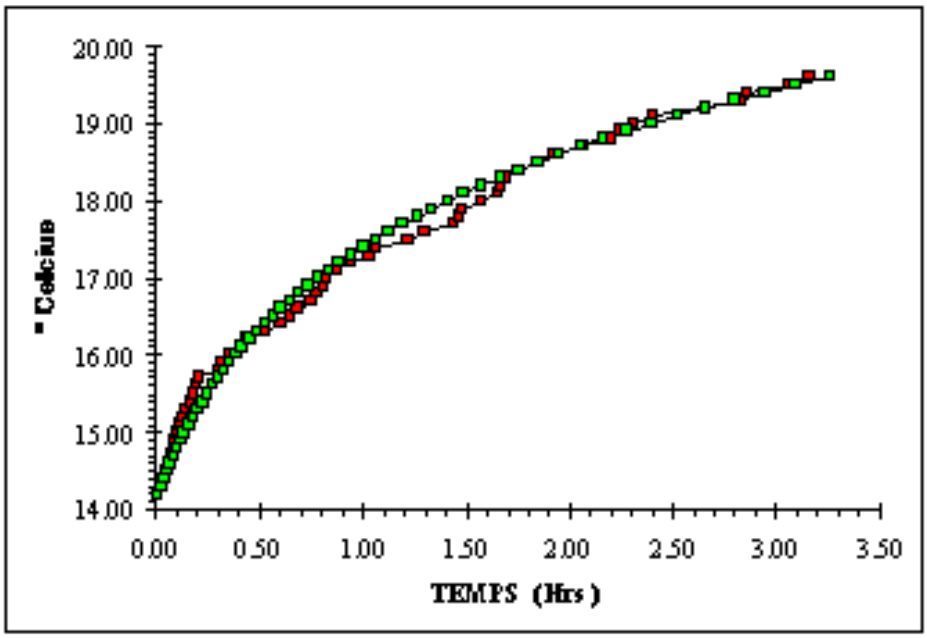

FIG. 4.1 – Courbe de difficulté du maintien de la température ambiante.

Le SMARTHERM est un système adaptatif qui, grâce aux différents senseurs dont il est muni et à l'intelligence artificielle dont il est pourvu, s'efforce de prendre en compte la plupart des facteurs environnementaux susceptibles d'affecter l'effort calorimétrique nécessaire pour assurer l'atteinte et le maintien du degré de confort thermique souhaité à l'intérieur d'une résidence par temps froid. Dans sa version de base, il dispose d'un senseur de température intérieure et d'un senseur de température extérieure. Dans sa version plus évoluée (le SMARTHERM Plus), il dispose en plus d'un senseur de vent (anémomètre) et d'un senseur d'ensoleillement (capteurs solaires).

Le SMARTHERM est composé de quatre éléments fondamentaux :

- Une station météorologique qui regroupe l'ensemble des senseurs extérieurs (température, vent, soleil).
- Un thermomètre électronique intérieur (senseur).
- Une carte électronique pour PC (interface de premier niveau avec le matériel et les équipements).
- Un logiciel de contrôle pour PC (interface avec l'utilisateur).

Le système 'monitore' constamment les conditions environnementales existantes par l'intermédiaire de la carte électronique qui sert entre autres d'interface avec les différents senseurs. Les données climatiques et environnementales obser-

vées sont enregistrées régulièrement à toute heure du jour et de la nuit dans l'ordinateur.

Dans le cours normal de son fonctionnement, le logiciel se sert de la base de connaissances accumulées pour prédire le comportement calorimétrique de la résidence dans laquelle il est installé et ceci, à tout moment où un apport calorifique est requis (maintien de confort thermique) ou sur le point de l'être (planification anticipée de l'effort calorifique à fournir pour supporter un accroissement de température imminent). Au moment jugé opportun, suite à l'analyse des différentes données, le logiciel passe les commandes appropriées au système de chauffage, encore une fois par l'intermédiaire de la carte électronique qui sert également d'interface avec la fournaise électrique, à l'huile, au gaz ou autre.

Au terme de chaque effort fourni (cycle de chauffage), le système auto-évalue la performance atteinte par rapport à la prévision qu'il avait préalablement faite de l'effort requis et ajuste automatiquement s'il y a lieu ses critères prévisionnels en fonction de l'écart (déviation) ou de l'absence d'écart (performance optimale) observé, afin de parfaire son éducation (auto-apprentissage). Ainsi, peu importe dans quel milieu il est installé, le système s'adapte continuellement à son environnement et améliore constamment sa performance. Même si des modifications importantes sont apportées à l'environnement (agrandissement de la résidence, amélioration de l'isolation de la résidence, amélioration du système de chauffage, etc.), le système se réadaptera de lui-même avec le temps. Nul besoin de reprogrammer le système, ni de changer certains paramètres, c'est un système adaptatif.

FIG. 4.2 – Carte PCI du SmartTherm.

4.3.2 Projet Lumière

L'objectif du projet Lumière est de donner une assistance aux utilisateurs de logiciels. C'est le projet de recherche lié aux “compagnons” utilisés par Microsoft.

Ce projet est basé sur un modèle bayésien gérant les différentes actions et requêtes. Ce projet de recherche inclus :

1. La construction d'un modèle bayésien basé sur l'observation des différentes actions et requêtes des utilisateurs.
2. L'augmentation la vitesse d'accès aux différents évènements utilisateurs.
3. Le développement un langage pour transformer les évènements en variable d'observation utilisé dans le modèle bayésien.
4. Le développement des profiles selon un résultat d'expertise sur un utilisateur.
5. Le développement une architecture pour une interface utilisateur intelligente.

L'objectif est de s'adapter automatiquement dans le temps à l'utilisateur en fonction de ses besoins et de son activité.

4.3.3 Projet Astrolab

Le projet Astrolab d'IBM [SBM97] permet de récupérer les différents états des systèmes hétérogènes présents sur un LAN pour effectuer une mise à jour de configuration de manière dynamique. Le système est scalable et fonctionne de manière distribué. La mise à jour de configuration concerne les drivers, les logiciels, ainsi que les paramètres spécifique à l'administration du LAN. Ce projet permet l'abstraction d'une administration système hétérogène en s'adaptant aux évolutions utilisateurs, logiciels et matériels.

4.3.4 Bilan de ces recherches

Les différents projets présentés dans cette section montrent que les systèmes existant s'orientent vers une adaptation dans le but d'offrir un service aux utilisateurs. Un point commun de ces projets est que le système évolue dans un environnement non stable, mais tout de même borné. Un autre point commun de ces projets est que le cœur du système ne s'auto-modifie pas pour s'adapter en temps-réel, une configuration de base et fixe est nécessaire pour le bon fonctionnement des logiciels. L'architecture de ces projets est très spécifique, les systèmes ne sont pas génériques et utilisables pour une nouvelle problématique.

Une planification dynamique de processus permettrait une auto-modification en interne débouchant sur des actions adaptées aux différentes situations envisageables dans l'environnement. Ceci permettrait aussi de faire évoluer la configuration de base sans demander d'actions particulière aux utilisateurs, la généricité du système serait alors assurée.

Deuxième partie

Modélisation et fonctionnement du système auto-adaptatif générique

Chapitre 5

La boucle systémique

5.1 Fonctionnement général

L'objectif de ces recherches est d'architecturer un système autour d'une boucle systémique permettant à ce système d'interpréter en temps-réel son environnement en fonction de ses connaissances et des ses expériences passées. Une fois l'environnement représenté et interprété par le système, celui-ci sera modifié puis réinterprété pour toute nouvelle modification. Toute l'analyse de l'environnement et la prise de décision se fait dynamiquement. Toutes les variables et structures présentent dans le système seront sujette à modification dans le temps.

5.2 Activation des connaissances

Le système dispose d'une base de connaissance dynamique, prédéfinie par le concepteur. La récupération d'informations par capteur entraîne l'activation des connaissances utiles à la représentation de ce qui se passe dans l'environnement. Il faut obligatoirement que ces connaissances soient présentes dans le système et reliées à une expérience. Une expérience est représentée par un graphe de connaissances pour que celles-ci puissent être utilisées.

5.3 Génération et représentation de scènes

Définition d'une scène : *une scène est une sélection d'une partie des entités actives du système, ce que nous appelons un focus sur une partie représentée d'un environnement. Cette partie active et sélectionnée est accessible grâce aux différents capteurs que possède le système évoluant dans l'environnement et à ses connaissances (figure 5.1).*

La représentation d'une scène à un instant t permettra de mettre en évidence l'association des connaissances utilisées et le comportement de l'entité face à son contexte. On peut penser à une "scène de théâtre" où l'entité aurait à jouer son rôle

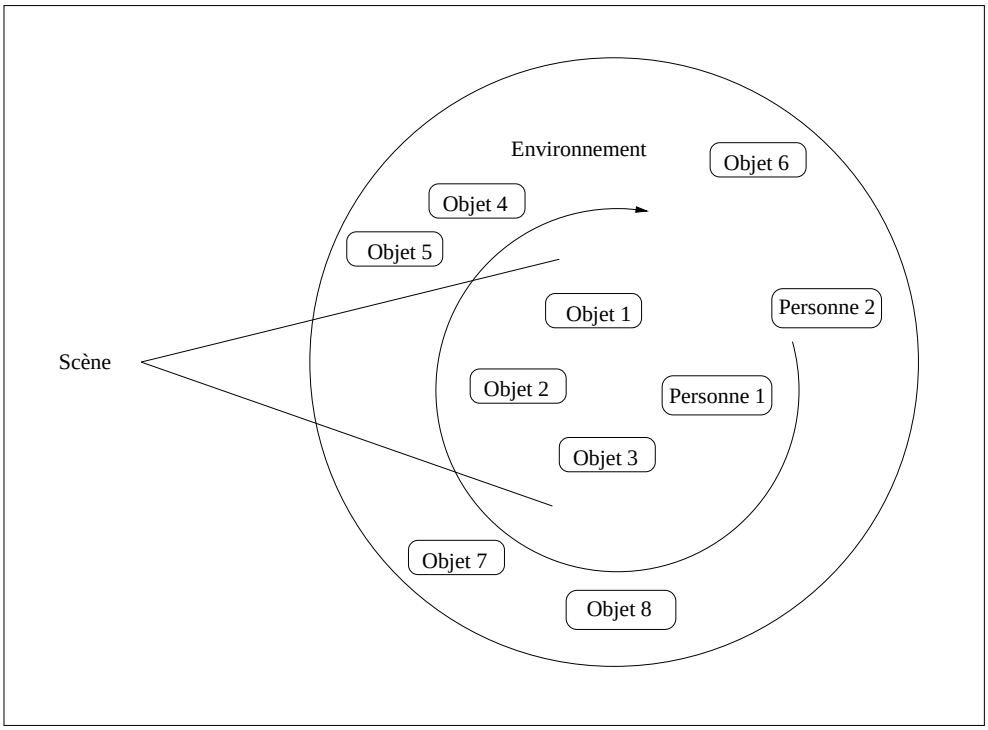

FIG. 5.1 – Une scène et ses différentes entités.

selon la situation. En fonction de cette représentation, il sera possible d'extrapoler le comportement de l'entité et de modifier certains paramètres en temps réel afin de l'orienter de manière plus précise si le cas d'activité nécessaire devait se présenter.

La figure 5.2 montre une représentation de scène (appelé aussi "Point focal") qui est assez simple. Une sphère représente un ensemble de connaissances, fortement liées, potentiellement utilisables pour l'interprétation de la scène. Les connaissances présentes dans la sphère la plus importante au niveau de la taille seront utilisées pour agir sur l'environnement. Au niveau des différentes teintes, les sphères tirant vers le bleu (teinte foncé) sont homogènes et pérennes dans la représentation alors que les sphères tirant sur le rouge (teinte claire) sont hétérogènes et sont sujettes à disparaître dans la mise à jour de la représentation. Les liens entre les sphères représentent les relations existantes entre certains agents. Cela veut dire que les agents peuvent transiter d'une sphère à l'autre, ce qui donnerait plus d'importance à certaines sphères dans le temps. La figure 5.3 montre une mise à jour s'orientant vers une représentation un peu plus complexe. Le paramétrage du système permet d'avoir une représentation de scènes plus ou moins profonde. De cette manière, il est possible de voir l'orientation future de la représentation courante.

Les figures 5.4 et 5.5 montrent deux représentations de scènes donnant une possibilité de communication afin de permettre une décision de groupe. Chaque représentation appartient à une entité bien distinct.

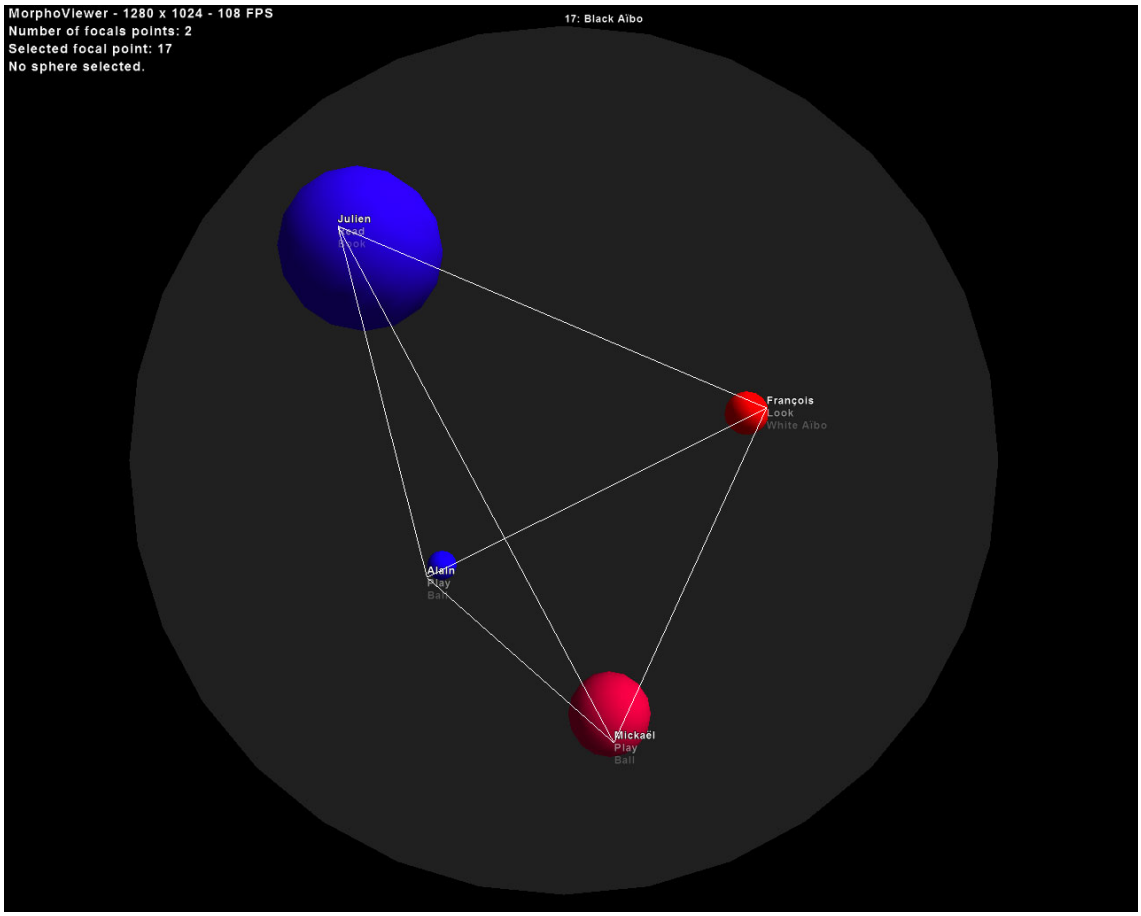

FIG. 5.2 – Une représentation de scène simple.

5.4 Vers un résultat non optimal mais satisfaisant

L'activation des connaissances et la représentation de l'environnement courant permettent de mettre en place un plan d'action que est mis à jour de manière continue. Le but est d'être capable de réaliser un objectif de manière spécifique et adaptée et ceci de manière adéquate mais non optimale. Prenons un exemple. Lorsque nous posons un verre sur une table, nous ne calculons pas la distance qu'il y a entre le bord de la table et le verre. Ce que l'on veut, c'est poser le verre sur la table, ceci est notre objectif. Celui-ci est atteint lorsque nous posons le verre sur la table. L'objectif de ces recherches est de réussir à faire la même chose avec un système

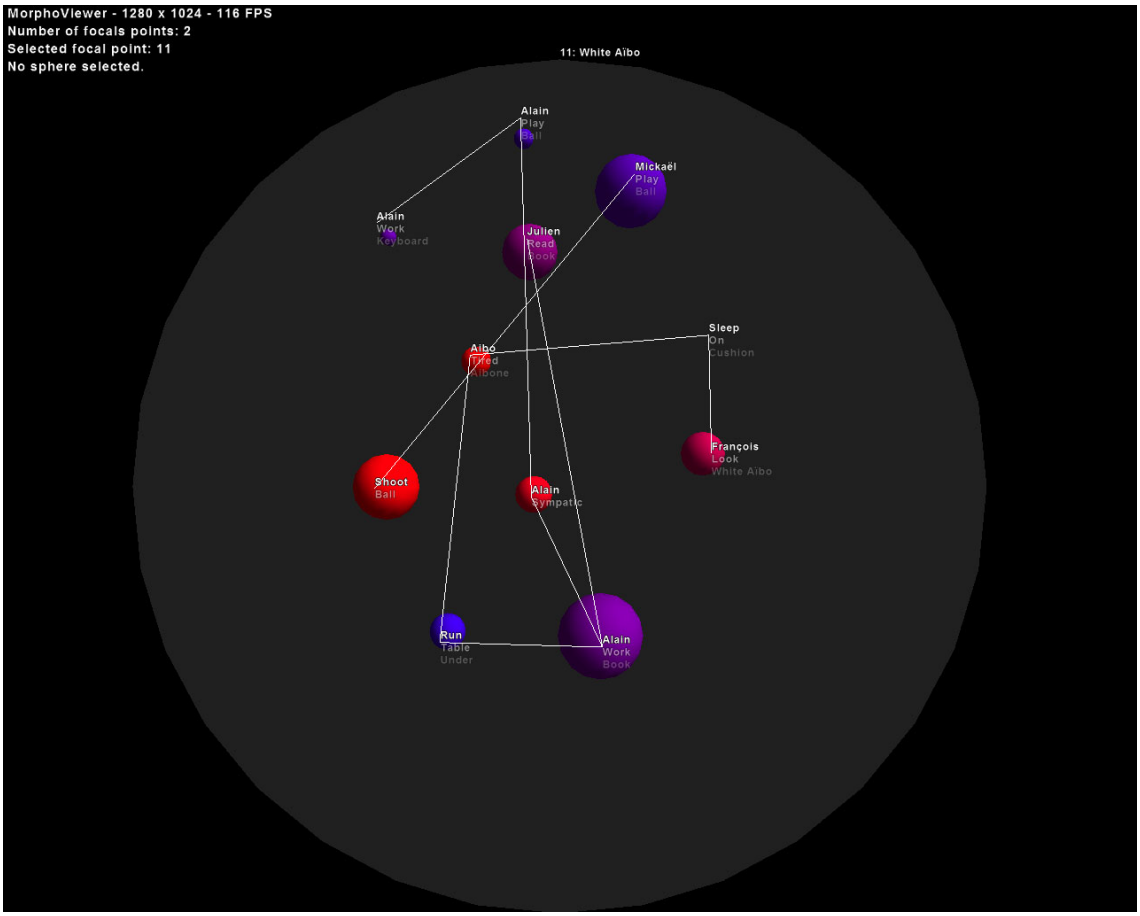

FIG. 5.3 – Une représentation de scène plus complexe.

informatique : une atteinte d'objectifs satisfaisante, rapide, efficace, utilisant des connaissances.

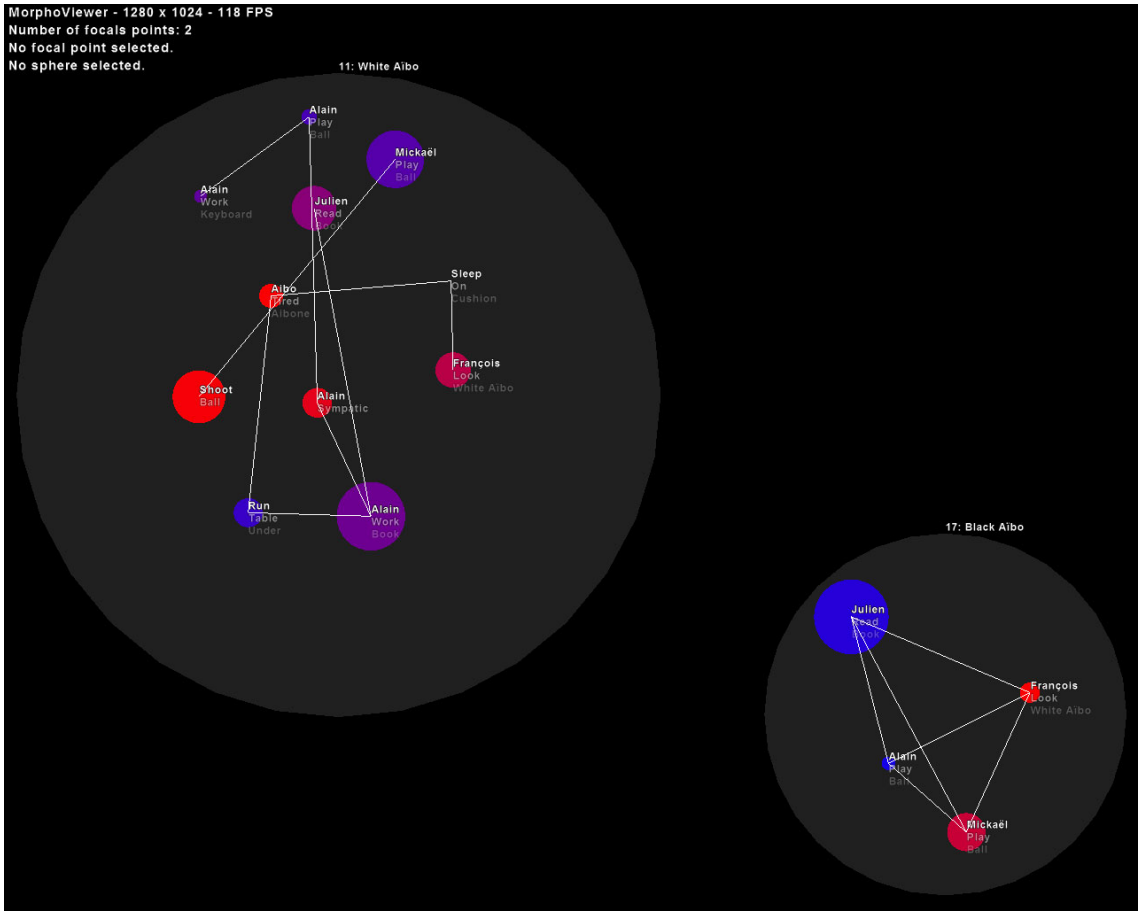

FIG. 5.4 – Deux représentations de scène.

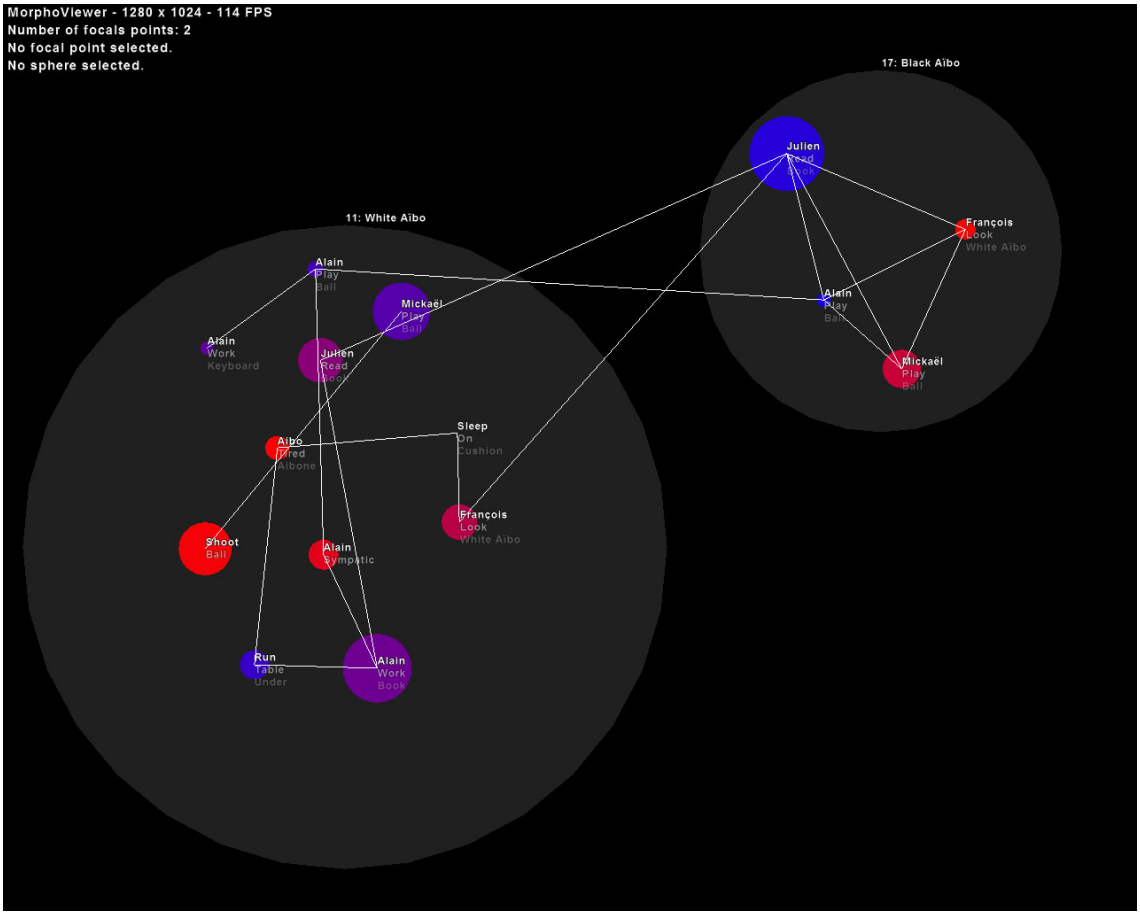

FIG. 5.5 – Deux représentations de scène communiquant entre elle.

Chapitre 6

Modèle proposé

6.1 Les ontologies

Une ontologie est une spécification pour une conceptualisation [cf. [Gru93]]. Elle est ici l'élément fondamental du système permettant de préciser son niveau connaissance [New82]. L'étape cruciale sera la transposition de la connaissance de niveau ontologique vers une structure exploitable en temps-réel par le système.

Une ontologie vient du regroupement de connaissances particulières sur un sujet donné. Certains projets comme Protegé [EFSM99] ou Conceptnet [LS04] permettent de créer des ontologies de manière automatique. La connaissance que le système utilisera peut venir de ces ontologies ou plus simplement d'une expérience personnelle.

On entend par transposition ontologique la représentation des rôles des agents en leur donnant une connaissance portant sur des sujets précis. Cette connaissance se basera sur une suite de concepts liés entre eux par des chemins ayant des distances spécifiques graduées. Ces distances sémantiques serviront à définir précisément les accointances entre les agents. Prenons un exemple concernant le robot Aibo. Un évènement s'est passé le mardi matin à dix heures : le robot a joué à la balle. Une expérience et des connaissances sont représentées dans cet évènement, il y aura un lien très fort entre les mots mardi, matin, robot (lui même), jouer et balle. Cette expérience pourra être utilisée ultérieurement par le robot puisqu'elle est mémorisée. Ce mode de fonctionnement lui permet d'apprendre plusieurs façons de faire pour résoudre un problème.

Il existe donc deux parties pour que l'ontologie soit utilisée : la description ontologique et le traitement via la transposition agent.

6.1.1 Description ontologique

En théorie, l'ontologie est décrite comme un graphe tri-dimensionnel où les noeuds sont classés de manière spécifique. Cette classification donne des informations sur les origines de la connaissance. Dans notre cas, nous allons considéré neuf éléments de classification, nombre qui pourra changer en fonction de l'évolution de l'entité dans son environnement. Cette classification se doit de respecter la logique et les usages de la société pour coller à la réalité.
Les différents éléments de la connaissance ontologique sont les suivants :

1. "Capacities" : toutes les capacités physiques de l'entité matérielle, c'est à dire, les mesures et caractères des senseurs et les effecteurs.
2. "Objects" : objets visibles simples ou composés comme un cube, une sphère, un clavier, un manteau, une table ou une chaise. Un objet en général, une chose selon les explications de Heidegger dans [Hei88].
3. "Verbs" : actuellement, les verbes en général. Mais cette section pourra évoluer vers une classification plus précise en prenant en compte les verbes passifs et actifs.
4. "Colors" : les couleurs de bases et les différents tons.
5. "State" : l'état de l'entité qui est très lié avec la partie "Verbs".
6. "People" : toutes les personnes que l'entité est capable de reconnaître (en utilisant des réseaux de neurones spécifiques).
7. "Space" : cette section établit où se trouve l'entité par rapport à d'autres éléments donnés par l'ontologie.
8. "Simple emotions" : les émotions de base comme le plaisir ou la peine.
9. "Mixed emotions" : des émotions plus complexes utilisant les émotions de bases comme la culpabilité ou l'attirance qui incluent le plaisir, la joie, la peine.

Toutes ces connaissances peuvent évoluer on-line dans le système [cf. figure 6.1].

6.1.2 Traitement dynamique de l'ontologie

L'ontologie est distribuée dans les différents agents d'un système multi-agent massif. Chaque agent a la possibilité d'être activé par une donnée quelconque transmise au système [cf. [CC05]]. Pour chaque connaissance, un rôle spécifique est généré dans un agent ou un groupe d'agents. Chaque rôle a ainsi un taux de dispersion similaire à ce que l'on peut retrouver dans la logique floue. Par exemple, le rôle "ball" pourra avoir cinq agents avec des taux de reconnaissance contextuelle différents. Les données arrivent des senseurs pour atteindre tous les agents présents dans le système. Si la donnée correspond à une connaissance d'un agent, l'agent devient actif et alerte les autres agents se trouvant dans son groupe d'accointances.

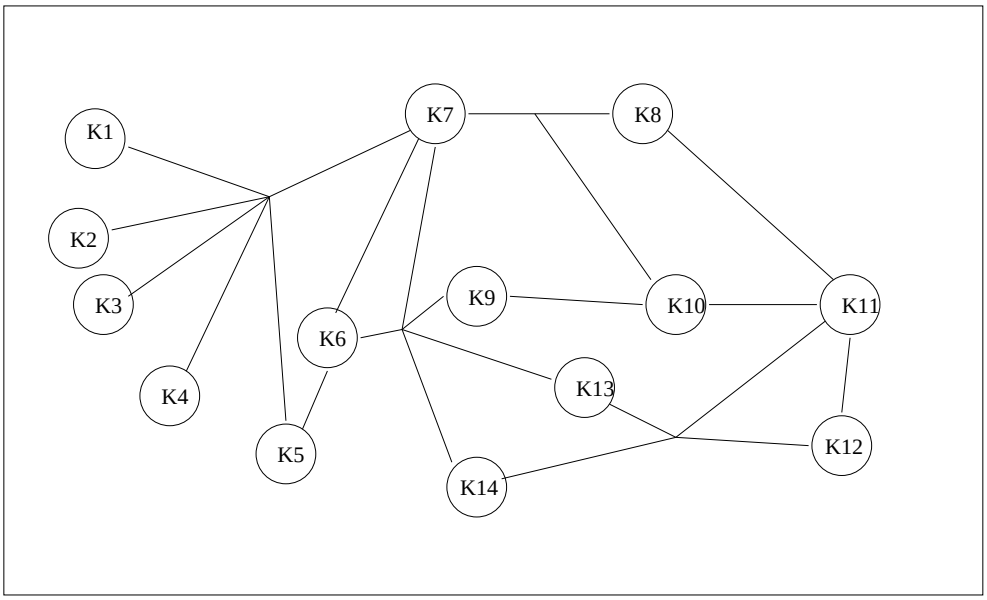

FIG. 6.1 – L'ontologie est représenté par un graphe. Tous les éléments présents dans la base de connaissances possèdent des accointances avec d'autres éléments, ces accointances sont représenté par des liens. Ici, chaque connaissance est numéroté (K1..K14).

Les catégories des rôles, des agents et les réseaux d'accointances de ceux-ci vont former des structures géométriques simple que l'on peut projeter sur des plans : **les plans de classifications**. Un plan de classification représente une catégorie cognitive venant de l'ontologie et regroupant de concepts [Gru93]. **Nous transposons ces connaissances dans des organisations d'agents, permettant d'utiliser la notion de distance dans ces organisations : nous introduisons des métriques dans les SMA**. Les agents proches dans un plan ont donc des rôles sémantiquement proches. Les agents actifs déforment ces plans selon l'intensité de leur activité [cf. figure 6.2]. Ceci est une distinction majeur avec les plans conceptuels des ontologies, qui sont statiques. On aura ainsi une représentation géométrique de l'activité agent, selon les rôles et selon les activations des agents. Il y aura de très nombreux plans de classification liés entre eux dans le système et représentés par des agents de structurants abordés dans la section 6.4.2. Ceci est défini par l'utilisateur qui entre la connaissance dans le système.

6.1.3 Activités et Intentions

Chaque groupe d'agents possède des intentions et des objectifs, qui ont un rapport direct ou indirect avec les capteurs présents sur l'entité. Ces groupes ne fonctionnent pas de manière isolée, ils communiquent entre eux pour atteindre un

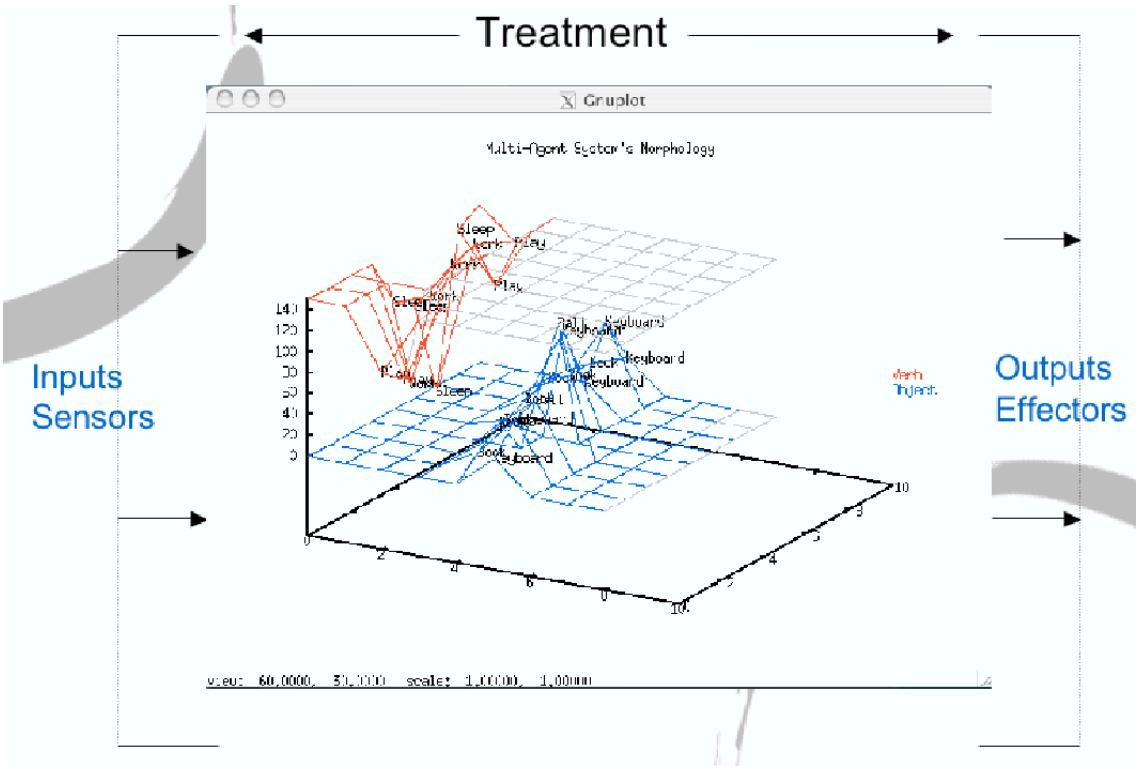

FIG. 6.2 – L'activation des agents entraîne une déformation des plans de classification ontologique.

traitement global de la connaissance et en faire une synthèse représentée par les différents états des agents [cf. figure 6.3].

Les types des agents vont être déterminants en ce qui concerne la reconnaissance des informations, leur exploitation comme connaissance et la prise de décision qui s'en suit. Tout agent traitant de la connaissance est appelé "agent Aspectuel". Il est capable de communiquer avec les autres agents selon ses accointances. Il peut de plus se cloner. Il possède un état qui évolue dans le temps. Tout agent aspectuel possède un type cognitif qui dépend de son rôle et du contenu de sa base de connaissances.

6.2 Emotions et représentation des émotions

L'émotion joue un rôle dans la prise de décision humaine expliquent Bechara, Damasio H. and Damasio A.R. in [BDD00]. C'est une découverte intéressante pour le thème de la décision et de l'optimisation en informatique. Cette explication est supportée par des scientifiques en cognition tels Lerner and Keltner [LK00] qui dé-

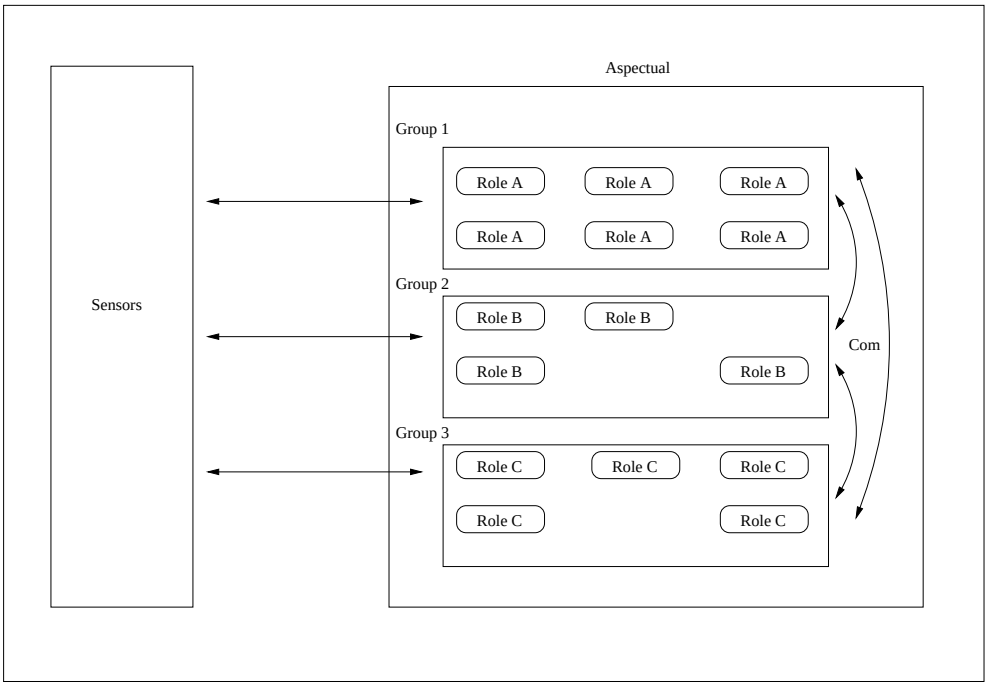

FIG. 6.3 – Les agents respectent une classification ontologique pour former des groupes d'activités plus ou moins importants en fonction du contexte ou l'entité se trouve. En plus de la communication intra-groupe, tous les groupes communiquent entre eux en donnant la possibilité au système de les agréger.

crivent un modèle comportant des émotions spécifiques qui influencent le jugement et le choix d'un être humain. Ceci revient à dire que les émotions font parties de nos jugement et de nos choix, elles modifient la raison et nous suivons ce point de vue dans le développement du processus de prise de décision du robot. Un élément crucial, à savoir est qu'il existe une différence entre la description et le traitement de l'émotion. A ce stade, une question similaire à été posée à la section 6.1 : comment l'émotion est-elle décrite dans un cerveau artificiel et comment est-elle traitée.

6.2.1 Description des émotions

Durant une vie, la connaissance et le savoir faire augmentent avec de nouveaux comportements, de nouvelles sensations et de nouvelles émotions. Avant l'âge de dix ans, un individu peut ressentir quelque chose comme de la joie associée à une autre émotion difficile à décrire. Avec l'expérience et l'apprentissage continu, il est possible de reconnaître et de nommer une nouvelle émotion. Pour Lafortune [LDD+02], l'intelligence émotionnelle existe. Une émotion peut être composée d'autres émotions. Il existe une classification des émotions selon Larivey, qui explique son système dans [Lar03]. Dans cette classification, il y a quatre éléments principaux :

1. Les émotions simples : désir, affection, excitation, joie, plaisir et peine, pour en nommer quelques unes.
2. Les émotions mixtes : des expériences émotionnelles incluant certaines émotions simples. On y retrouve l'amour ou encore la passion.
3. Les émotions repoussées : une expérience physique repoussant une expérience émotionnelle. On y retrouve la détresse, l'anxiété, la panique, la nervosité, cette liste n'est pas exhaustive.
4. Les pseudo-émotions : une métaphore décrivant la réalité : il n'y a pas vraiment d'émotions. On y retrouve la gentillesse, la sympathie, la timidité, la compassion, la confusion, et bien d'autres.

Cette classification est utilisée ici. Considérons qu'une émotion est une connaissance avec un nom qui représente une interprétation de certaines données du système nerveux. L'émotion est décrite dans l'ontologie comme représenté dans le graphe de la section 6.1.1. Un nouveau graphe est attaché au graphe de l'ontologie : le graphe des émotions (6.4), qui sera inclus dans le graphe de l'ontologie durant l'exécution de la boucle systémique[1]. Avec une grande connaissance, ce graphe est très complexe et il se complexifie encore plus avec l'expérience.

[1] Il n'y a pas de différence entre les connaissances. Il est facile pour un humain d'avoir une classification pour savoir qu'est-ce qui est sauvegardé dans l'ontologie. Le système, lui, ne fait pas la différence.

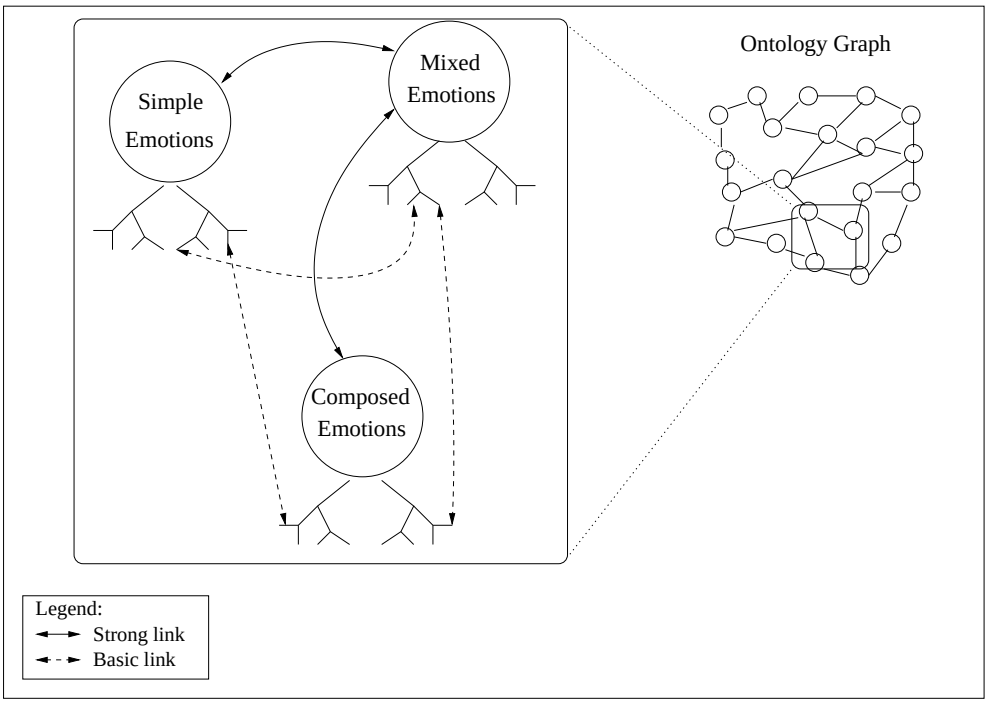

FIG. 6.4 – Le graphe des émotions est attaché à celui de l'ontologie. Il est possible de décrire et de donner une orientation émotionnelle au système seulement avec des mots.

6.2.2 Traitement dynamique des émotions

Une émotion est traitée de la même manière qu'une connaissance dans l'ontologie. Chaque émotion connue est distribuée dans le système multi-agent[1] et traitée comme une connaissance, c'est à dire une possibilité de communiquer avec des capacités physiques ou mentales. La seule différence, qui est tout de même primordiale, est que la suite d'émotions est considérée comme une suite d'objectifs paramétrable directement incorporée dans le système afin de déboucher sur une orientation synergique avec les objectifs passés par les utilisateurs. Nous avons donc des objectifs émotionels possédant la même structure qu'un objectif de base vu dans la section 6.6. Les émotions sont la pour altérer les objectifs rationnels du système, c'est à dire les supporter ou les rabaisser. La définition globale d'une émotion peut être exprimée simplement dans un contexte général. Cependant, dans un cas pratique, il est plus difficile de le faire. En fait, cette définition est liée à l'expérience et à l'apprentissage. C'est pour cette raison qu'une culture est cruciale dans n'importe quelle éducation. Une partie concernant les émotions dans le

[1] Le système entier est un système multi-agent massif. Quand l'ontologie augmente, le nombre potentiel d'agents augmente aussi. Actuellement, plus de trois milles agents ont été testés simultanément.

système multi-agent peut être observée sur la figure 6.5. Pour chaque système utilisant un cerveau artificiel, avec les mêmes senseurs et effecteurs, il est intéressant de noter que le comportement peut être totalement différent si les expériences des robots, ainsi que leurs configurations, sont différentes. Cette différence est dûe à l'intensité émotionnelle, celle-ci se résume à un signal borné continu dans le temps. Si l'amplitude du signal est grande, l'objectif en cours de traitement sera fortement perturbé par le processus émotionnel.

Utiliser cette méthode de traitement pour configurer le système avec une configuration particulière est relativement simple. Par exemple, une personne peut développer du stress plus facilement qu'une autre car cela fait parti de sa personne. Il est tout à fait possible de faire la même chose avec le système auto-adaptatif. Une plus grande importance peut être donnée à la section du système traitant l'élément "peur" par exemple, en augmentant le nombre d'agents s'y référant, ce qui augmente l'activation de l'élément dans le système. Toutes les émotions peuvent être accentuée pour transformer le système en système fortement émotionnel.

6.3 Approche SMA : création et organisation d'agents

L'usage d'un système multi-agent va permettre de traiter l'information temps-réel du robot en fonction de son importance, de sa qualité et de sa quantité. De la méthode d'agentification à partir des ontologies va dépendre l'efficacité du système.

6.3.1 L'agent et sa proactivité

La structure d'un agent est permanente dans tout le système multi-agent, ce sont les types et les rôles qui les différencieront. Cette structure est de nature très simple car la puissance du système ne s'appuie pas sur la complexité de l'architecture d'un agent, mais sur la société des agents et les rôles qu'ils y jouent [cf. 6.6]. Un agent est donc composé :

- d'une base de connaissances propre,
- d'un moteur d'inférence,
- de rôles et d'objectifs,
- d'une API de communication avec l'environnement,
- d'états.

La base de connaissances de chaque agent a une grande importance car c'est elle qui va orienter l'agent vers un rôle particulier. Cette base de connaissances aura une taille réduite car l'ontologie sera distribuée dans de nombreux agents. Cette petite base de connaissances pourra évoluer, s'enrichir ou se simplifier en fonction des échanges que pourront faire les agents entre eux. Le premier prototype du système utilise de très petites bases de connaissances, mais cette taille pourra être augmentée en fonction de la complexité des connaissances à traiter des moyens de

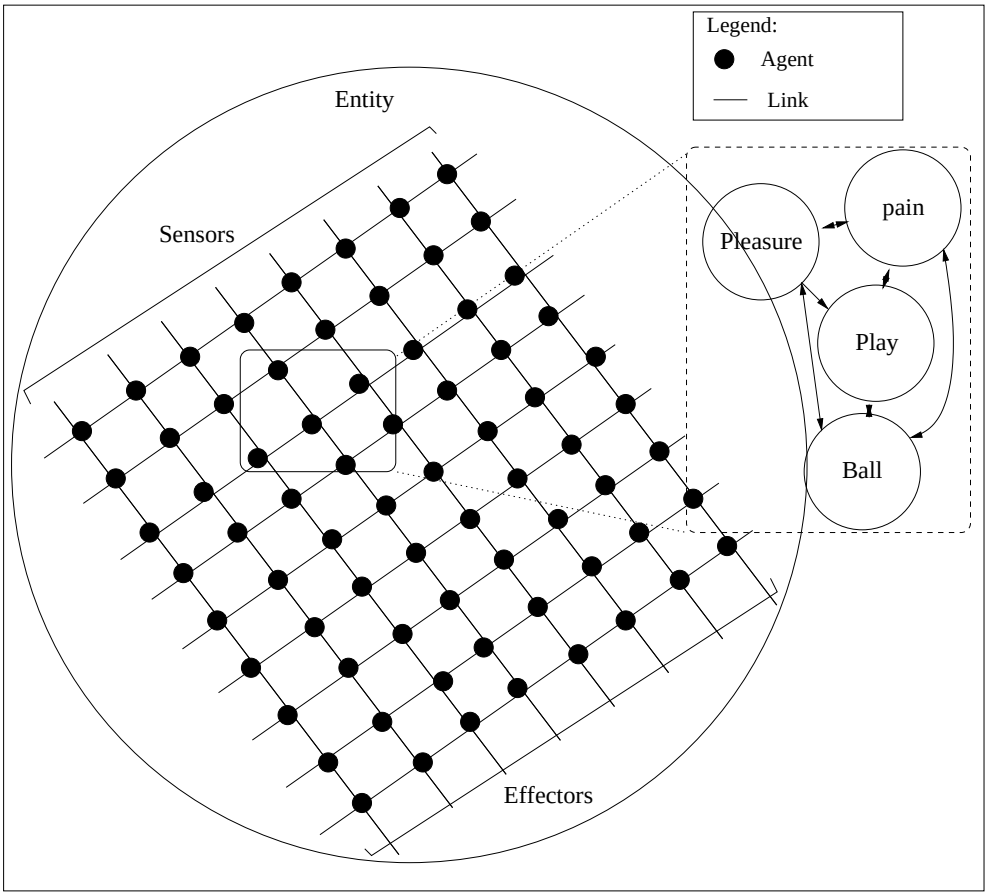

FIG. 6.5 – Le cadre en pointillé représente une section montrant différents liens entre les émotions, les verbes et les objets. Avec un peu plus d'expérience, beaucoup d'éléments dans l'ontologie peuvent être liés. Ici, il est plaisant pour le système de jouer à la balle, mais la peine est aussi une hypothèse possible car si la taille de la section change, une peine venant du passé peut apparaître.

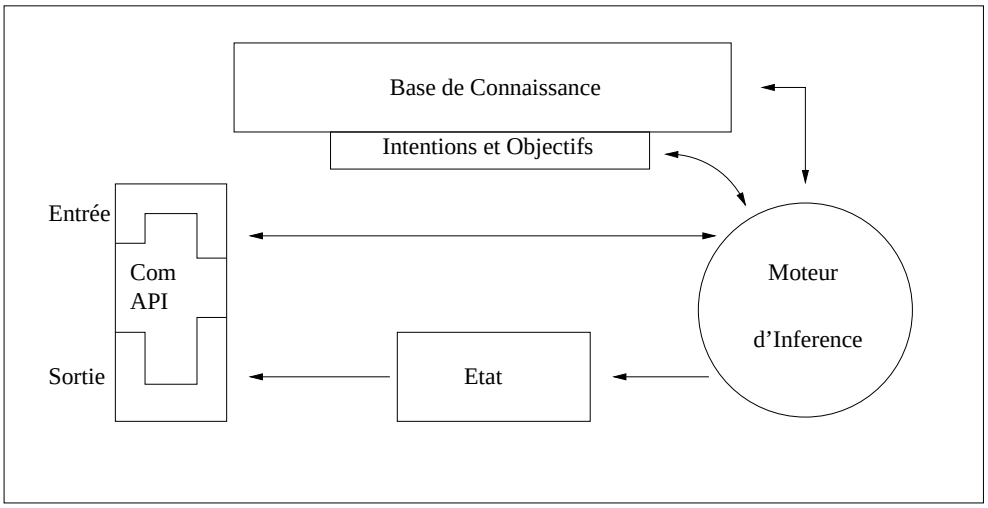

FIG. 6.6 – L'architecture d'un agent reste simple car c'est l'organisation interne qui fait la puissance du système. Cette architecture doit être minimale afin que son utilisation soit la plus souple possible.

calcul.

Les différents états que peut prendre un agent seront sauvegardés sous forme d'un historique des états. L'état en cours sera transmis à des agents de rôles différents, et correspondra à des éléments de leurs bases de connaissances. Ce transfert d'information d'agent à agent est très important, l'API de communication doit être très performante afin qu'il n'y ait pas de temps de latence entre les questions et les réponses. L'API prend en compte une entrée et une sortie, qui auront des formats différents, l'entrée de l'API correspondra au type présent dans la base de connaissances, et la sortie, quant à elle, prendra le type du statut de l'agent.

Les éléments de connaissance des agents vont leur permettre de passer dans un certain état, et d'effectuer leurs actions motivées. L'exécution de ces actions sera appelée **intention de l'agent**. L'intention de l'agent est paramétrée par un caractère numérique appelé **énergie** : un agent consomme de l'énergie et peut en distribuer. Cette énergie correspond à l'énergie globale paramétrable attribuée au système. Celle-ci est partagé équitablement entre les différents agents. Après avoir traité une information, l'agent va donc avoir l'intention de faire telle ou telle action en vue d'atteindre son objectif, utilisant une pile d'intention qui correspond aux différents objectifs présents dans la base [cf. 6.7]. Ces deux éléments sont en synchronisation et communiquent directement avec le moteur d'inférence afin que celui-ci effectue le traitement de la base de connaissances en fonction de cette synchronisation.

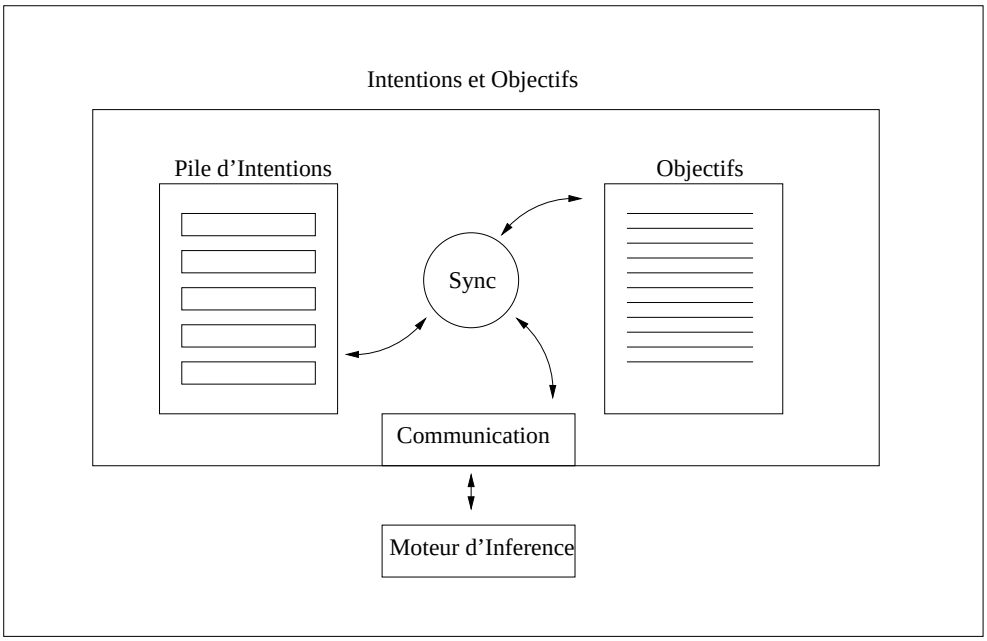

FIG. 6.7 – Chaque agent doit effectuer certaines actions pour garder un niveau d'énergie suffisant pour exister. Tous les agents stockent les différentes actions à effectuer afin de respecter ses objectifs internes qui sont directement liés à l'ontologie.

Les intentions vont permettre d'identifier les agents actifs, et, ce qui est plus important, vont permettre aux agents de se reconnaître entre eux. Nous verrons par la suite que l'organisation des agents se fera par un système de groupe évolutif en fonction des connaissances qui transitent dans le système.

6.3.2 Mise en action d'un agent dans le système

Dans le système proposé, un agent est autonome et très actif au niveau communication, il agit pour son compte avec des objectifs bien définis. Un certains nombres d'agents devrons être gérés directement par le système. Ces agents communiquent entre eux de manière asynchrone. Ces contraintes nous obligent à travailler au niveau des processus afin d'effectuer des traitements parallèles utiles au bon fonctionnement du système auto-adaptatif. Le nombre d'agents pouvant être important (supérieur à mille), il est difficilement envisageable d'utiliser des processus systèmes totalement gérés par le scheduler du système d'exploitation (souvent limités à deux milles pour une machine standard). De plus, le contrôle global du système se base sur des formes géométriques générées grâce à des études statistiques sur la communication des agents (notion de morphologie, section 8.3). Cette

communication est possible grâce à des canaux de communication entre les agents qui sont actifs en mémoire lors de l'exécution du système. Nous nous baserons donc sur des processus légers de type thread qui seront gérés par une machine virtuelle, ce qui permet de ne pas alourdir la gestion globale des processus du système d'exploitation.

6.3.2.1 Liste de threads

Le système fonctionne grâce à une société d'agents communiquants, autonomes et configurables en temps-réel. La section 6.3.1 a présenté la structure générale du fonctionnement d'un agent. Mais cette structure seule ne suffit pas à ce qu'un agent ait le comportement attendu dans le système. La création d'un rôle spécifique à l'agent lui donnant un objectif durant son existence et la caractéristique de communication asynchrone contraint l'agent à se décomposer en de multiple threads.

Actuellement, le nombre de threads utilisés par un agent est de huit et est fixe, mais il pourra évoluer en fonction de ses capacité à communiquer avec son environnement. Voici la liste des threads utiles au bon fonctionnement d'un agent :

1. Filtre interne pour la mise en relation des rôles avec les objectifs.
2. Modification des paramètres de l'agent en temps-réel.
3. Modification des accointances entre les rôles en temps-réel.
4. Lecture des messages organisationnels et redirection vers les agents typés.
5. Ecriture des messages organisationnels.
6. Envoi du niveau d'activité organisationnelle à un instant t (paramétrable).
7. Lecture des messages concernant les rôles.
8. Ecriture des messages concernant les rôles.

Le système a besoin de ces huit threads pour sa communication interne et son paramétrage en temps-réel. Les expérimentations vu dans le chapitre 9 ont montré qu'un agent avait besoin de ces threads afin de respecter les objectifs d'auto-adaptabilité du système. Les threads de lecture et d'écriture permettent un routage interne des messages vers les bon types d'agents afin de ne pas alourdir la communication globale. Les différents types de messages transitant dans le système sont présentés dans la section 7.4.2.4.

6.3.2.2 Création de threads et envoi de messages

Pour créer un système de prise de décision adaptatif, il est important de traiter les informations en parallèle afin de prendre en compte toutes les informations de l'environnement et de bien orienter la prise de décision. Le système a été développé en Oz/Mozart [cf. [RH04]], qui a aussi été conçu pour développer des systèmes

```
thread
   C  % Code qui sera exécuté dans le thread
end
```

FIG. 6.8 – Création d'un thread

```
% Dans le Thread A:
% Input est une variable de type 'Port',
% et Stream est une variable de type
% String.
Input = {NewPort Stream}

% Dans le Thread B:
{Port.send A.Input 'Message'}
```

FIG. 6.9 – Envoi d'un message entre deux threads

multi-agents [cf. [HSW93]]. Il existe trois éléments essentiels dans le système Mozart permettant de traiter l'information en parallèle.

Dans un premier temps, le système Oz/Mozart facilite la programmation par concurrence en intégrant la création de threads directement dans la syntaxe du langage. Ceci permet d'écrire très facilement un thread [cf. figure E.3].

Cette seule caractéristique ne permet pas de faire du parallélisme. Dans un second temps, une caractéristique est de dire que toute variable est instanciable. La machine virtuelle peut s'arrêter sur une variable et attendre son instance pour continuer à exécuter le bytecode.

La notion de "message passing", permet facilement d'envoyer des messages entre les threads créés par le système. La création d'un tunnel de communication se fait de manière classique : Soient deux threads *A* et *B*, le thread *B* envoie un message au thread *A* [cf. figure 6.9].

Cela permet de développer une communication inter-processus de manière synchrone ou asynchrone. Ainsi, un agent attendant un message sur un port particulier pourra le faire en écoutant un canal [cf. figure 6.10]. Un agent écoute plusieurs ports en même temps afin de partager les tâches et traiter toutes les informations transitant dans le système.

```
thread
   for @msg in @stream do
      {System.show self.id#':'#@msg} % Exec on receaving
        {Port.send @ag.input @id}
        {Record.forAll {Array.toRecord ag @agents}
 proc {$ A} {Port.send A.input @id} end}
      {self process(@msg)}
   end
```

FIG. 6.10 – Thread d'un agent écoutant un port particulier

```
% Rec est un record de 2 éléments appelés foo et length.
% P est une proc\'edure permettant d'afficher le contenu
% en fonction du label pass\'e en paramètre.
Rec = rec(foo:3 length:6)
P = proc{V}
    {Browse Rec.V}
end
```

FIG. 6.11 – Pattern matching en Oz

6.3.2.3 Design pattern

La méthode du design pattern est très utile du point de vue de la facilité de codage, de l'efficacité du code et de son évolution. Cette méthode associée à la capacité d'interpréter du code ou du bytecode Oz en temps-réel, permet d'atteindre une grande flexibilité : faire évoluer le programme sans compilation pendant qu'il fonctionne [cf. figure 6.11].

Ceci permet de gérer les types d'agents de manière générique et adaptative grâce à des "records" : le type Record en Oz. Il suffit de passer les types en paramètre pour ensuite créer les agents. De nouveaux types pourront être injectés après l'initialisation à la manière d'un "ajout" dans un record. La création d'agents par rapport à certains types prédéfinis est grandement facilité [cf. figure 6.12].

6.4 Les organisations du Système Multi-Agent

Un système multi-agent offre un nombre important de possibilités tant par son utilisation que par les méthodes existantes pour sa réalisation. Les sciences cognitives se basent de plus en plus sur cette méthode qui calque parfaitement le fonctionnement du cerveau humain [Car04] [Min88]. Le cerveau humain peut être considéré comme fonctionnant comme une société d'agents communiquant possédant de micro-connaissances qui forment toute la connaissance d'un être humain.

```
 R = param(aspectual_sens:SA aspectual_eff:EA
   morpho:MA analyze:AA)

         {Record.forAllInd R proc {$ L O}
{System.show L}
{System.show O}
for I in 0..O do
   {Dictionary.put self.types.L I
    {New CAg
     init((I + @count) self.type.as self.sens
  self nil nil)}}
end
count<-(O + @count)
     end}
```

FIG. 6.12 – Création d'agents de différents types

La connaissance est donc distribuée à travers le cerveau, qui, grâce à certaines substances, comme par exemple l'ocytocine et la vasopressine en ce qui concerne le sentiment de plaisir[PV04], peut diriger le comportement. Le but de la méthode employée ici est de s'inspirer au maximum de l'être humain en ce qui concerne la prise de décision [WJXyJcXf03].

Mais comment fait un être humain pour se comporter dans son environnement ? La réponse correspond aux besoins, aux objectifs, aux intentions et à la prise de décision possible. Chaque être doit assouvir ses besoins primaires pour atteindre le premier but : survivre. Viennent ensuite les besoins secondaires avec différents objectifs en fonction du caractère et des désirs de la personne. Pour atteindre ses objectifs, chaque être possède des intentions d'action qui le mèneront vers une satisfaction. Mais beaucoup d'obstacles vont barrer son chemin, ce qui ne l'empêchera pas d'avancer grâce aux décisions prises à la volée pour une éventuelle ré-orientation, c'est la résolution de problèmes ponctuels [NOK$^+$02]. Les intentions peuvent être étudiées en fonction des objectifs pour que le choix des actions soit satisfaisant [CL98]. Avant d'avoir l'intention de faire quelque chose, il faut prendre une décision, mais il faut aussi un élément majeur qui pousse un être à prendre une décision : l'émotion. Si une machine réussi à ressentir une émotion, elle utilisera son matériel afin d'exécuter une action qui correspond à une suite d'actions représentant la décision prise en temps-réel.

Le système multi-agent va permettre de reproduire des mécanismes complexes liés à la génération des émotions et des représentations d'environnement. Pour réussir cela, certains rôles sont à définir. Ils vont permettre de traiter l'informa-

tion en fonction de son importance et en fonction de sa quantité. Le terme employé pour cette partie est "agentification”, et de cette “agentification” va dépendre l’efficacité du système.

Les agents respectent une organisation bien spécifique afin de traiter les connaissances de manière à respecter les objectifs passés en paramètre [cf. figure 6.13].

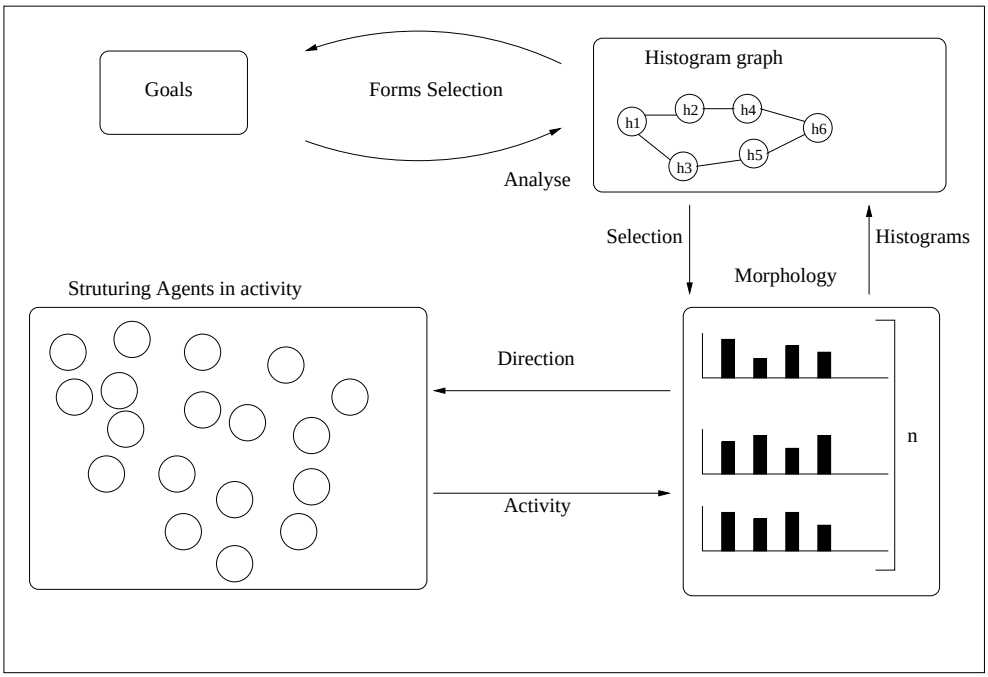

FIG. 6.13 – Les statistiques sur la communication des agents structurants permettent de construire des histogrammes afin d’avoir une vision externe du comportement du système pour sélectionner les formes aptes à la bonne réalisation des objectifs.

6.4.1 Organisation générique agent

Le système multi-agent permettant de donner de l’autonomie adaptative et des émotions à une machine est complexe par son organisation et sa communication intérieure. Le type et la structure d’agents utilisés sont communs à tous les agents présents dans le système. Cet agent est appelé Agent Aspectuel. Son rôle premier est de récupérer les informations afin de combler ses besoins primaires, ainsi que ses intentions. Ce traitement donnera un résultat, ou plutôt un état à l’agent qui sera utilisé par des homologues de manière identique, et il n’y a que le type de connaissances qui change. Le mode de fonctionnement d’un agent est identique pour tous les agents du système, tous les agents traitent la connaissance de manière identique.

Cette méthode va permettre une gestion simplifiée des agents dont le nombre ne restera pas fixe. Ce nombre va varier pour suivre l'instabilité de l'environnement. Un drône de renseignement va prendre de grands risques en s'introduisant sur le territoire ennemi, les radars sont nombreux. Si une zone est considérée "à risque" alors que ce n'était pas le cas au début de la mission, le chemin à suivre va changer. L'action s'orientera sur la détection d'obstacles et des radars afin de calculer une nouvelle trajectoire, au lieu de prendre des photos de la base ennemie par exemple. Le drône ne sera plus d'aucune utilité s'il se fait repérer, et pire, la mission aura échoué et l'ennemi connaîtra les informations détenues par son adversaire. Le drône a vraiment besoin de s'adapter rapidement à une situation.

Le cas est encore plus extrême en ce qui concerne un drône d'attaque. Celui-ci possédera une charge utile importante : matériel de détection, de communication, des armes. La prise de décision est complexe car le drône peut décider d'utiliser ses armes, mais il devra aussi recalculer à chaque instant une trajectoire, en cas de conflit, il ne pourra pas suivre une trajectoire pré calculée. Les hostilités font que les informations arrivant au drône sont diverses et variées. La possibilité de décupler ou de diminuer le nombre d'agents dans le système en fonction du flux d'information permet d'accéder à un traitement satisfaisant de la situation.

Cette technique permet de s'orienter vers un modèle calquant le fonctionnement du cerveau humain. Les neurones, les synapses sont de structures identiques, mais ne jouent pas forcément le même rôle, et n'ont pas forcément le même but. L'agent Aspectuel est la base du système pour le traitement physique de la connaissance. Mais il faut faire la différence entre le traitement physique de la connaissance et la connaissance elle même car un traitement peut être identique pour deux types d'informations différentes, ce qui donne, évidement, un résultat qui, lui, est différent au niveau de l'activité, et celle-ci sera systématiquement prise en compte.

6.4.2 Organisation structurante

Les agents aspectuels se dérivent en différents types selon les usages et activités. L'*Agent Structurant* représente l'ensemble des connaissances activables et construit un plan local. Ce plan est une suite d'actions éventuelles à effectuer par le robot. Ces actions doivent être réellement exécutables par le robot : on doit donc avoir une liste ordonnée d'actions possibles à jour en fonction des évolutions du robot. Comme dans tous les rôles présents dans ce système, la communication est primordiale, la base de connaissances présente dans les agents structurants représente l'ensemble des actions que le robot est capable d'effectuer [cf. figure 6.14].

Ces plans actions seront présents dans tous les agents structurants, selon leur rôle. Le but est que tout groupe d'agents structurants passe à un état d'action en fonction de l'interprétation de sa conformation. Nous devrons pour cela la notion

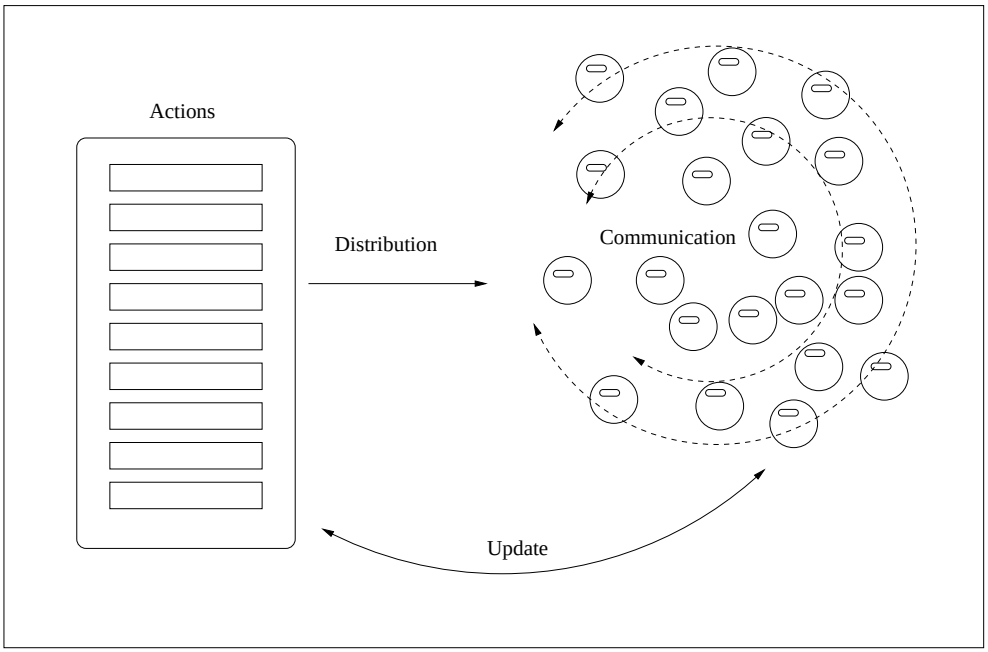

FIG. 6.14 – Toute l'ontologie du système est distribuée dans le système multi-agent massif. Dans cette ontologie, on retrouve aussi la connaissance correspondant aux différents mouvements d'actions à effectuer dans un environnement.

introduire la notion d'agent de morphologie. Le contrôle du système sera fondé sur les caractères morphologiques des agrégations d'agents structurants.

Les informations venant des capteurs sont interprétées par les agents structurants dont les co-activités dessinent des formes [cf. 6.15]. Une forme pourra exprimer un état émotionnel, comme la peur, la joie, le doute, la crainte. Le but est d'avoir le maximum de formes agrégatives d'agents structurants pour avoir un système riche qualitativement, flexible et sensible.

6.4.3 Architecture générale du système

6.4.3.1 Approche

Nous nous intéressons à des entités comprises comme des systèmes software-hardware, possédant un corps et un système de supervision autonome [cf. figure 6.16] [cf. [Bro02] et [BS93]].

Le système de supervision est lié à différents senseurs, effecteurs, capteurs de position, une caméra ou tout autre périphérique. Toutes les données sont traitées en parallèle pour organiser le flot d'information et effectuer une interprétation au

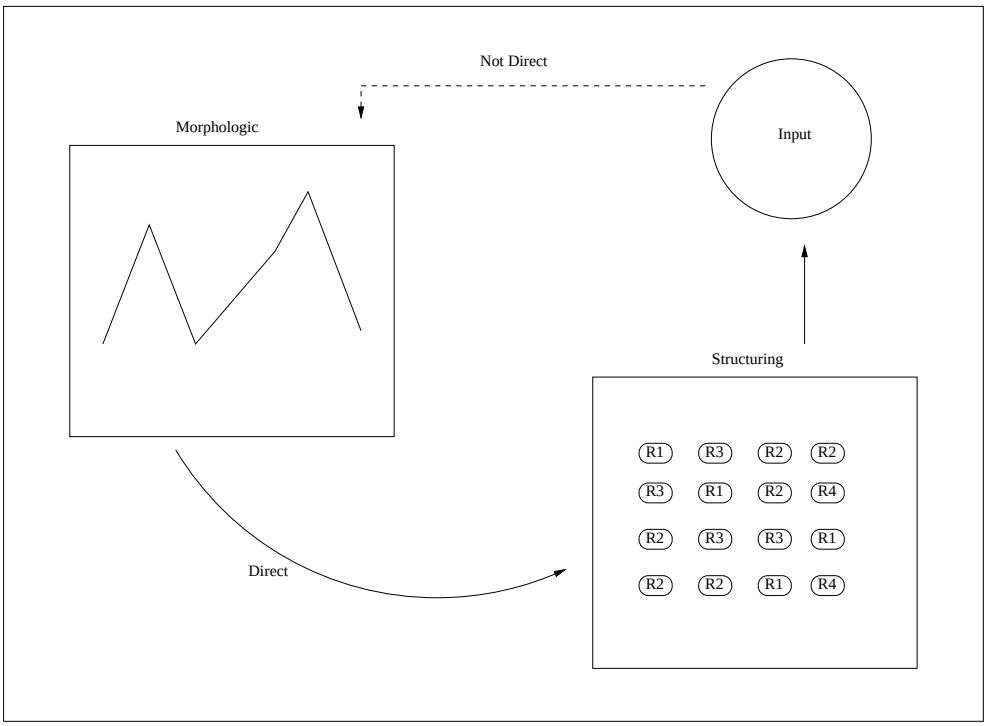

FIG. 6.15 – La forme géométrique découlant de l'activation de certains agents structurants permet l'orientation et la prise de décision du système afin de respecter les objectifs passés en paramètre.

niveau connaissance selon Newell [New82]. Cinq niveaux de traitement pour une prise de décision en environnement instable comme décrit dans [CC05] rentrent en considération : représenter une situation contextuelle, diriger l'attention sur les éléments spécifiques, comme des objets ou des évènements, et ressentir les émotions liées à ces éléments comme le montrent [CC05] et [CCC05]. Il faut aussi vérifier que ces éléments peuvent être utilisés pour atteindre un objectif, construire un plan d'action comportemental et réagir aux retours d'actions. Tous ces niveaux forment une boucle systémique décrite sur la figure 6.17 :

Nous posons que le système est en exécution bouclée continuelle.

6.4.3.2 Architecture

Le système évolue dans un environnement instable possédant différents contextes d'interprétation. Pour des raisons de performance, le système de contrôle n'est pas embarqué dans le robot [cf. figure 6.18]. Une station de travail distante exécute le système en restant continuellement en contact avec celui-ci . L'environnement et le matériel ne sont pas dissociés, ils forment un tout synergique évoluant en parallèle en fonction de l'un et de l'autre.

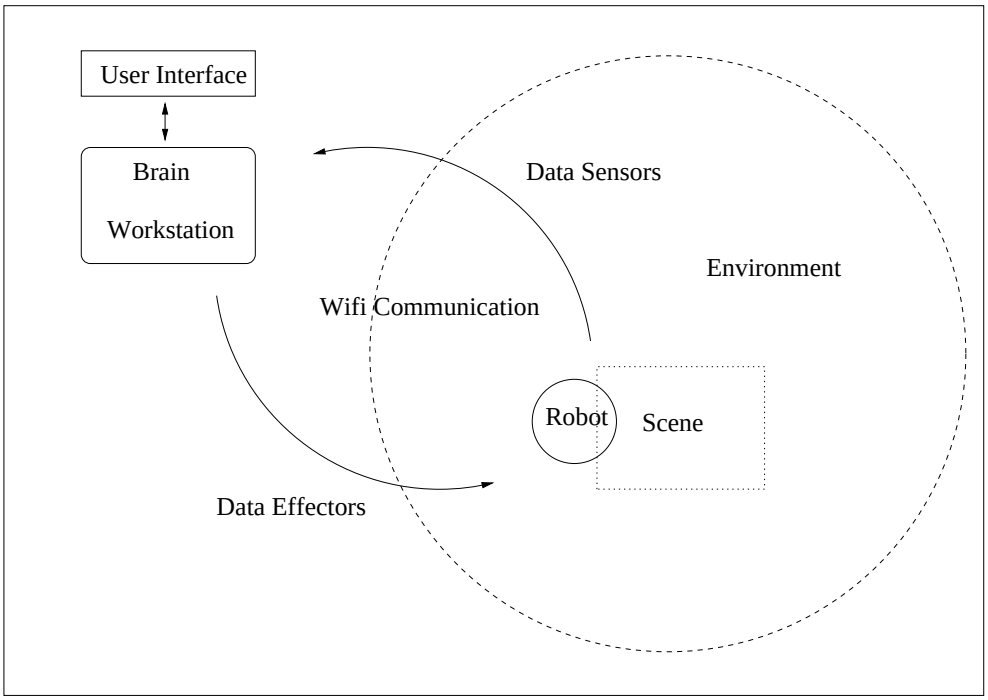

FIG. 6.16 – La station de travail permettant l'évolution dans un environnement instable n'est pas embarquée dans le matériel. Cependant, la communication entre la station et le matériel reste continue pour traiter les informations en temps-réel.

...senseurs ⟶ représentation ⟶ interprétation ⟶ plan d'action ⟶ effecteurs ⟶ senseurs...

FIG. 6.17 – Boucle systémique.

Nous verrons dans la section 6.1 concernant l'ontologie que celle-ci est totalement distribuée dans un système multi-agent. Cela permet d'attribuer des connaissances locales à tous les agents du système, et de leur donner des objectifs à court terme afin qu'ils évoluent en communiquant entre eux [cf. [Min88]]. Les agents seront représentés chacun par des processus. La distribution de base active un certain nombre de processus à l'intérieur des agents, ce qui permet leur réelle autonomie dans un environnement qui se limite à la mémoire utilisée par le système en fonctionnement sur un ordinateur [cf. figure 6.19].

Nous appelons scène ce que l'entité pourra percevoir de l'environnement en utilisant tous ses capteurs et sa connaissance, la communication interne entre les agents du système va permettre une analyse très fine de la scène ou se trouve l'entité

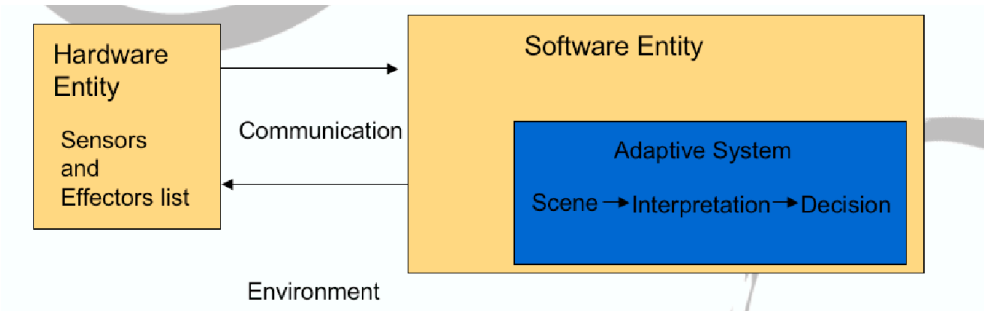

FIG. 6.18 – Une boucle systémique se forme entre l'entité et l'environnement pour former un tout indissociable. La communication entre la partie matérielle et la partie logicielle est continue et ininterrompue. Une action modifiera l'environnement ce qui provoquera une nouvelle action.

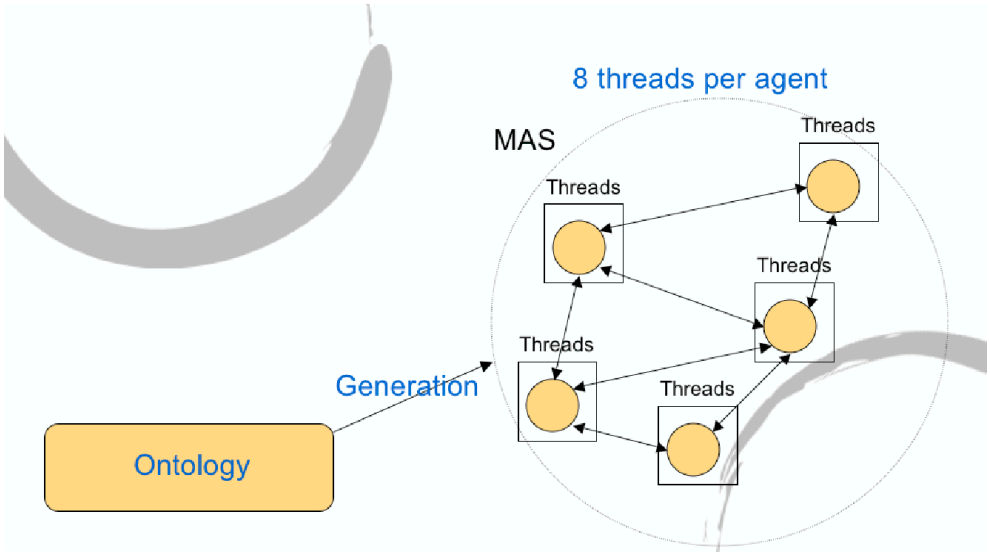

FIG. 6.19 – La totalité de l'ontologie est distribuée dans un système multi-agent massif où chaque agent utilise huit processus distincts afin d'évoluer dans son environnement restreint.

matérielle. Cette analyse pourra se faire grâce à certaines activations des connaissances des agents nécessaires pour comprendre dans quel environnement et dans quel contexte l'entité évolue. L'activation des connaissances est représentée par une activation de certains agents s'organisant de manière spécifique. Elle permet de construire en temps-réel une forme géométrique correspondant à un instant t de l'évolution de l'entité dans l'environnement [cf. [Car04] et [Cam05]]. La notion de conformation de l'ensemble des processus actifs (agents) est centrale pour le

contrôle de notre système. La notion de forme géométrique permet d'effectuer une comparaison entre la forme courante et une forme cible qui correspond aux objectifs, pour orienter correctement le fonctionnement système [cf. figure 6.20]. Le système prendra des décisions de manière continue afin de s'approcher le plus possible de son objectif, un peu à la manière d'un algorithme anytime [cf. [Cam02]].

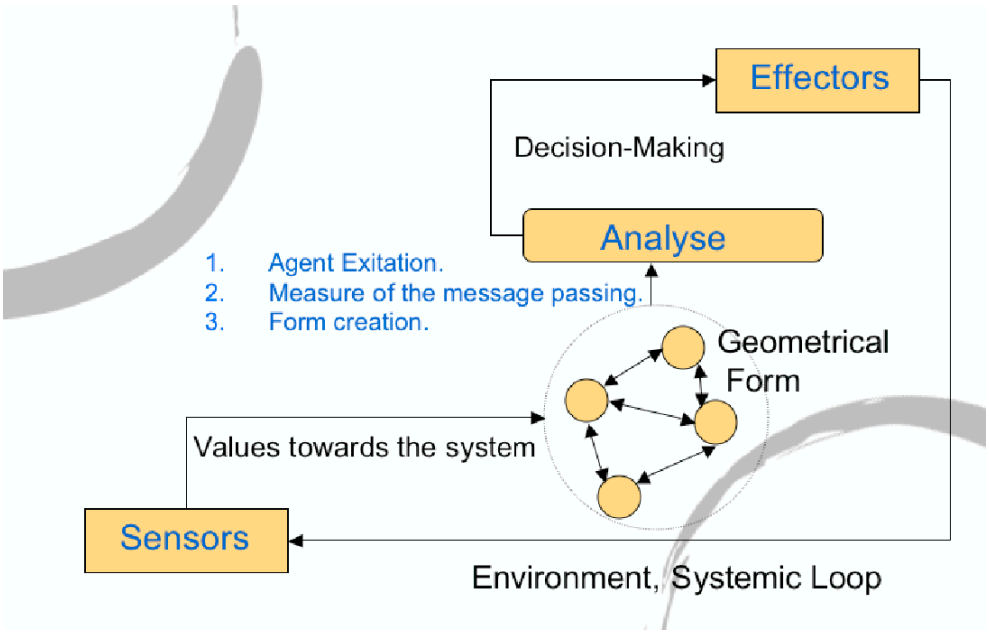

FIG. 6.20 – La construction d'une représentation courante est une activation des agents liés à la connaissance nécessaire pour la représenter. Les agents communiquent entre eux grâce à un passage de messages. Tous ces messages permettent de créer une forme particulière qui sera utilisée pour la prise de décision.

Le robot évoluera donc dans un environnement instable en interprétant de manière continuelle différentes scènes qu'il construit et qui se présentent à lui : il y a bien un bouclage systémique [cf. figure 6.21]. Les différentes activations d'agents provoqueront une comparaison de formes géométriques afin de produire une suite de "points focaux" correspondant au comportement physique que pourra avoir le robot [cf. [CC06a]].

6.4.4 Organisation de morphologie

La représentation morphologique des objets ou des éléments appartenant à un environnement à l'intérieur du cerveau a été évoquée par la Neurobiologie [PAF+03]. La morphologie nous permet d'interpréter la représentation de l'information et de lui donner une forme géométrique. Elle possède bien sûr plusieurs

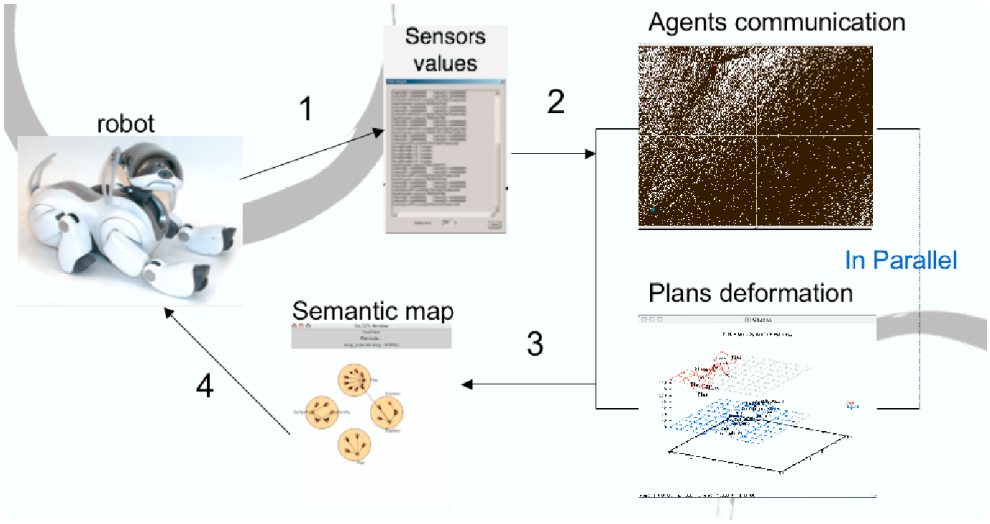

FIG. 6.21 – Le robot se fera une interprétation de scène qui lui permettra de comparer la forme géométrique créée avec ses objectifs. Si ces derniers sont différents, une décision est tout de même prise mais le système sera orienté en fonction des objectifs.

définitions correspondant à différentes sciences. Une définition de la morphologie se trouve en Annexe B. Celle-ci présente la morphologie comme une science se basant sur l'étude des formes d'un point de vue extérieur pour en retirer une signification.

Le représentation morphologique va s'effectuer en fonction de l'organisation des agents structurants. Ces derniers vont récupérer les informations venant des capteurs du robot, pour assouvir leur besoin primaire comme vu plus haut, ce qui va déboucher sur un état spécifique. C'est l'observation de l'ensemble des états des agents qui permettra d'attribuer une forme géométrique qui clôturera l'interprétation, comme le montre la figure 6.22.

Cette observation sera faite par les agents morphologiques. Pour ces agents, la connaissance se situe au niveau des états, tout comme leur but. Bien que ce soit tous deux des états, ils sont différents. L'état des agents structurants est un état environnemental (c'est à dire actif ou non en fonction des capteurs d'entrée), alors que l'état d'un agent morphologique est un état de statistique, c'est à dire qu'il utilise les statistiques faites sur la communication des agents structurants. Pour qu'il y ait état de statistique, il faut obligatoirement un état environnemental. L'agent morphologique possède une base de connaissances d'état environnemental, l'association de certains de ces états le met dans un état de statistique qui sera lui aussi interprété. Il y a donc énormément de communication et d'échange d'infor-

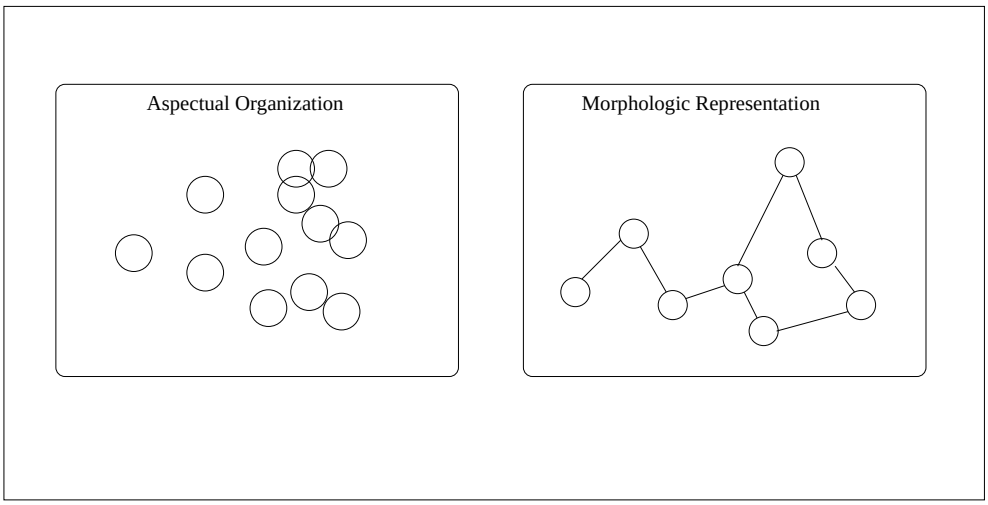

FIG. 6.22 – Organisation Aspectuelle et Représentation Morphologique

mations dans le système entre les agents de rôles identiques et les agents de rôles différents. La partie Structurante est en continuelle discussion, délibération avec la partie morphologique.

6.4.5 Organisation d'analyse

Il s'agit maintenant d'interpréter les conformations morphologiques (les différentes formes résultant de l'analyse statistique sur la communication des agents), d'analyser les différentes formes géométriques produites par les agents structurants. Cette analyse débouche sur la création d'un graphe d'analyse qui est soumis à un traitement spécifique afin de sélectionner une forme pour orienter le système et se diriger vers une forme cohérente avec les objectifs courants à atteindre qui sont passés en paramètre par l'utilisateur du système. Ce procédé permet au système de s'adapter à une évolution physique ou à un changement radical de l'environnement. Nous utilisons des mesures d'écart entre des histogrammes pour représenter la morphologie des groupes d'agents structurants [cf. figure 6.23], [cf. [Cam05]].

L'analyse permet de reconnaître des formes dynamiques et aussi de faire une synthèse dégageant les rôles les plus actifs. Les différents rôles cognitifs des agents structurants prennent une place importante dans le système. Chacun des rôles aura des liens avec d'autres rôles afin de construire une représentation cognitive la plus complète possible pour tout état de l'environnement [cf. figure 6.24].

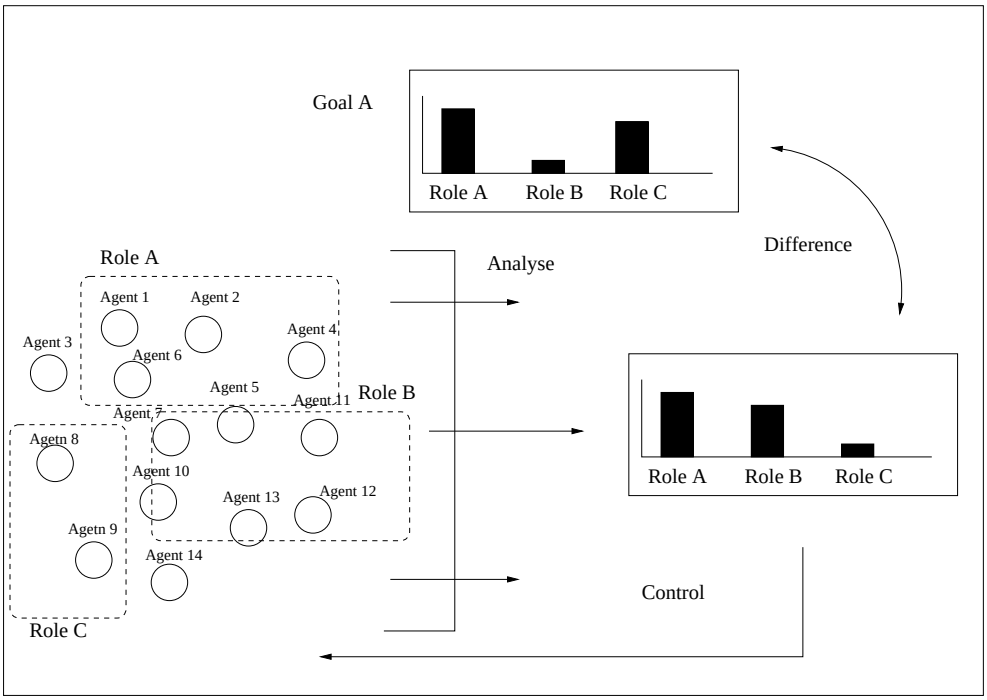

FIG. 6.23 – La morphologie des groupes d'agents structurants permet la création d'histogrammes qui seront comparé aux histogrammes générés à partir des objectifs passé en paramètre par l'utilisateur. Cela permet d'orienter le système vers certaines formes.

6.4.6 Synergie des Types

La complexité joue un rôle important dans ce système, elle permet d'avoir une organisation homogène, sans système de couches ou de hiérarchie. La complexité est décomposable en multitude d'entités simples et compréhensibles dont les objectifs sont simples. Le système comprend trois types bien distincts et très importants. Ils ont été décrits précédemment :

- Aspectuel.
- Morphologique.
- Structurant.
- Analyse.

Grâce à un système de synergie, les types s'utilisent les uns les autres, il existe entre eux une communication continuelle. Toute cette dynamique permettra au système, si le cas d'un ajout de matériel devait arriver, de connaître et d'apprendre à utiliser le nouveau matériel qui se trouve sur l'entité. La communication intensive, et incessante donnera, au final, une direction à suivre concernant l'envoi de messages. Celle-ci sera utilisée par le type Structurant, qui génèrera un plan d'action en fonction des buts du système. Ceci est possible grâce aux accointances de classification

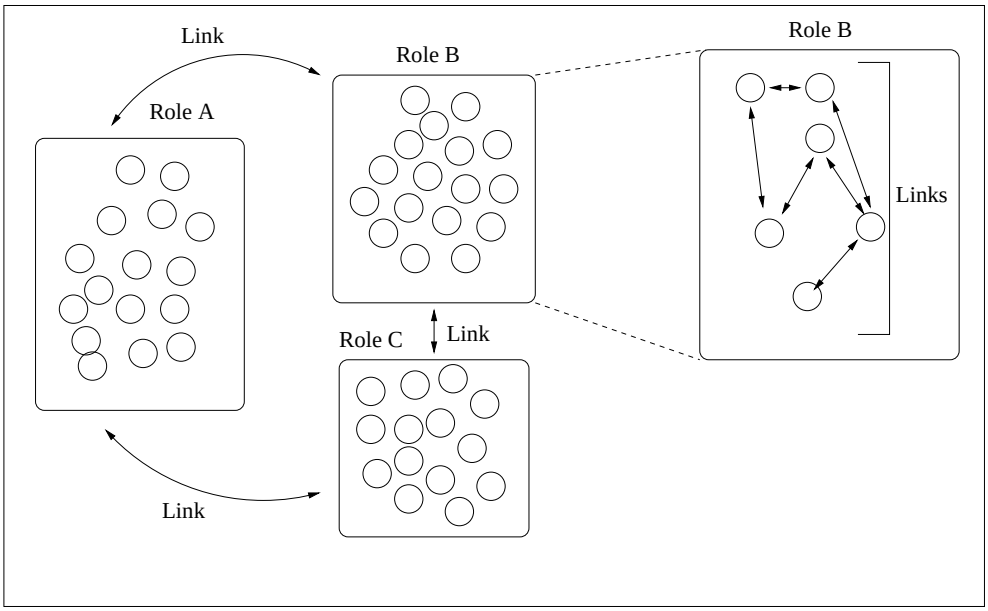

FIG. 6.24 – Les différents liens existants dans le système entre les agents et entre les groupes forment une ontologie distribuée en forme de graphe où chaque nœud est divisé en plusieurs processus.

"Physical capacities" que possèdent les agents structurants.

6.4.7 Synergie des Rôles

Les différents rôles des agents prennent une place importante dans le système. Chacun des rôles aura des liens avec d'autres rôles afin de construire une représentation cognitive d'une scène se passant dans un contexte particulier de l'environnement. Ceci est montré sur la figure 6.24. Les liens entre les rôles définiront les accointances entre les agents, ce qui permettra, en se basant sur les connaissances du système, de déduire certaines actions en cours de réalisation ou de reconnaître certains objets utiles aux besoins de l'entité.

6.5 Morphologie et formes géométriques

Le contrôle est crucial pour le Système Auto-Adaptatif (SAA). Dans les sections 6.1 et 6.2, nous avons expliqué qu'une activation au niveau des agents apparaît lors du passage des informations dans le système. Tous les agents présents dans le système sont traités dans un même temps, donc, lorsque certains agents s'activent simultanément, une forme géométrique est créée immédiatement. Le phénomène est appelé morphologie et a été découvert par René Thom [Tho89],

c'est une branche de la théorie des systèmes dynamiques. Pour contrôler un système multi-agent, nous redirigeons les lecteurs sur les recherches de Campagne [Cam05]. Le résultat est une adaptation de la morphologie de Thom pour les systèmes multi-agents. Notez que ces résultats de recherches ont étés adaptés pour une communication asynchrone et pour une auto-adaptation du système. Ils concernent essentiellement une analyse statistique des envois de messages des agents en interne pour ensuite créer des histogrammes correspondant aux différents moments de l'évolution du système dans un environnement non connu, c'est à dire le nombre de messages envoyés par agent par rapport au temps.

6.5.1 Notion de morphologie et stabilité structurelle

Pour René Thom, l'univers n'est pas un chaos, il est peuplé d'objets dont les formes sont récurrentes et qu'il nous est possible de désigner par des mots. Ces formes, pourvues d'une dénomination, sont nécessairement stables. En effet, R. Thom stipule qu'un objet donné naturellement est toujours soumis à des influences perturbatrices de la part du milieu extérieur qui, si faibles soient-elles, auront un effet sur sa forme. Cependant, pour rester reconnaissable aux objets similaires de sa classe, ces perturbations ne devront pas trop changer la forme de l'objet en question.

6.5.2 Définition de la morphologie par R. Thom

Un phénomène naturel, sur la nature duquel nous ne faisons aucune hypothèse, se déroule dans une boîte B ; il y aura donc lieu de considérer le produit $B*T$ de B par le temps T ; ce sera le domaine siège du processus considéré. On admet que l'observateur dispose de sondes ou d'autres moyens qui lui permettent de voir ce qui se passe au voisinage de tout point x de $B*T$. Une première classification des points de $B*T$ s'offre d'emblée ; lorsqu'au voisinage d'un point x de $B*T$, l'observateur ne peut rien déceler de remarquable ; si, phénoménologiquement, le point x ne diffère en rien des points voisins, on dira que ce point x est un point régulier du processus. Par définition même, les points réguliers forment un ouvert dans $B*T$; le complémentaire fermé des points réguliers dans $B*T$ sera appelé l'ensemble des points catastrophiques du processus ; au voisinage de tout point c de l'ensemble fermé K des points catastrophiques, le processus présente une discontinuité : "il se passe quelque chose" dans toute boule de centre c. La donnée du fermé K et la description des singularités en chacun de ces points constitue la morphologie du processus.

En termes moins techniques, nous pouvons dire que R. Thom choisit de considérer un phénomène comme une morphologie, c'est à dire comme un système de discontinuités qualitatives sur un espace substrat.

En empruntant un vocabulaire mathématique, on peut dire qu'il existe donc

dans un espace E un ouvert constitué des points représentant des formes structurellement stables, le complémentaire de cet ouvert comportant seulement des formes instables, qu'une infime perturbation peut transformer.

Intuitivement parlant, une situation est structurellement stable quand les situations voisines lui ressemblent. L'importance de cette notion est donc évidente, un phénomène structurellement instable ayant peu de chances d'être observé de manière stable. D'un point de vue expérimental, l'idée de la stabilité structurelle est que si l'on peut répéter une expérience approximativement dans les mêmes conditions, on obtiendra approximativement les mêmes résultats.

6.5.3 Utilisation de la morphologie pour le contrôle du SMA

6.5.3.1 Notion de projection

Idéalement, afin de faire l'analyse du système multi-agent, nous prendrions en compte tous les paramètres du système et des agents. Nous n'allons pas procéder de cette manière, d'une part parce qu'il y a trop d'informations et certaines informations seront redondantes, et également il faut que l'information se présente sous une forme manipulable. Comme on est en informatique, il suffira d'observer le segment de mémoire où réside l'agent. Mais toutes les informations du segment de mémoire ne seront pas utilisées. On procède donc à une extraction d'informations, mathématiquement représentable par une projection d'un espace sur un autre souvent moins riche qualitativement en informations, mais plus pertinent pour le contrôle.

Alain Cardon avait donné dans [Car00] une liste générique de paramètres à prendre en compte dans l'analyse. Il l'avait augmentée dans [Car04] pour donner la liste suivante :

- apparence externe : sa situation d'acteur par rapport au autres agents et à l'environnement :
  - suprématie : mesure du fait que l'agent se situe ou non en position de force par rapport aux agents reconnus par ses accointances, qu'il a ou non de nombreux alliés et que ses ennemis sont ou non puissants ;
  - indépendance : mesure de l'autonomie de l'agent dans ses actions, précisant s'il lui est nécessaire ou non de trouver des alliés pour atteindre ses buts, si ses actions sont soumises par rapport aux actions d'autres agents, si sa structure de connaissance est incluse dans celles d'autres agents ;
  - persistance : mesure de la longévité de l'agent, de sa durée de vie qui peut être courte ou longue ;
- aspect interne : son état structurel par rapport aux fonctions qui lui sont assignées.
  - facilité : mesure du fait que l'agent a atteint son état courant en rencontrant plus ou moins de difficultés, de résistances.

  - vélocité : vitesse prise par l'agent pour atteindre les buts qu'il s'est lui même assignés ou qui lui ont été donnés ;
- état organisationnel interne : son fonctionnement propre entendu comme un processus d'échange d'informations entre ses modules qui forment sa structure :
  - intensité du flux d'activation interne : mesure de la quantité d'informations échangées entre ses composants internes lui permettant d'aboutir à une activation visible de l'extérieur par les groupes auxquels il appartient ;
  - complexification : mesure de sa transformation structurelle engendrée par certains dysfonctionnements ou certaines situations difficiles ayant nécessité de transformer certains éléments de sa structure ;
  - fréquence communicationnelle : mesure de la relation communicationnelle entre l'agent et les autres agents, en dehors de toute sémantique, représentant une mesure de son couplage dans l'organisation ;
  - écart organisationnel : évaluation prenant en compte l'opérationnalité de la structure de l'agent et qui lui a permis de réaliser certaines actions sociales le conduisant à poser lui même une appréciation de la distance entre son propre état et l'état global de son environnement, tel qu'il se le représente.

6.5.3.2 Formes géométriques et métriques

Nous nous intéressons à la forme que peut prendre une organisation d'agents. La notion de forme sera précisée jusqu'à être celle de forme géométrique précise. Nous allons donner quelques exemples de mesures de formes concrètes et quelques types de formes que nous pouvons construire à partir de celles-ci (voir figure 3.4.1). Par exemple, à partir d'une valeur numérique propre de domaine R (taille mémoire occupée, valeur particulière d'une variable, durée de vie...), il est possible de construire des histogrammes. On peut étudier la distribution et donc la forme de ces histogrammes. En généralisant à plusieurs dimensions : l'espace est de domaine R^n, où n est le nombre de dimensions (le nombre de mesures différentes effectués sur l'agent), on peut construire des nuages de points ou des histogrammes multidimensionnels. Si les agents portent une sémantique (par exemple déterminée par leur rôle), on peut étendre ce domaine par la dimension sémantique : $R^n * S$, où S est l'ensemble représentant la sémantique. Cela permettra de mettre en évidence, comment les aspects qui font sens pour le concepteur (les rôles des agents, par exemple) sont en relation les uns avec les autres.

On peut également s'intéresser aux relations entre les agents : on dispose d'une valeur numérique par rapport à un autre agent. Il est alors possible de construire un graphe (préférence d'accointances, fréquence de communication ...). A partir d'un tel graphe, on peut étudier d'autres propriétés de la structure, telles que la connexité, le diamètre...

Finalement, le développement, dans le temps, de la forme peut apporter des informations supplémentaires sur le comportement de l'organisation d'agents. Cela se rapproche de la notion de morphologie d'un phénomène se déroulant dans le temps, comme nous l'avions cité dans la section 6.5.2.

Dans le cas où l'on envisage effectivement de faire du contrôle (et non juste de l'observation), la mesure effectuée sur l'agent qui sert à la construction de la forme devrait correspondre à une mesure qui peut varier de façon contrôlée, directement ou indirectement, en modifiant un certain paramètre de l'agent ou bien de son environnement. Par exemple, ou bien la mesure porte sur un paramètre de l'agent modifiable directement et le contrôle et la corrélation s'en trouvent simplifiés ; ou bien la valeur mesurée n'est pas directement modifiable, et il faut peut-être intervenir sur d'autres agents, ou bien sur l'environnement de l'agent, ou bien encore supprimer ou rajouter des agents particuliers pour modifier les valeurs afin de faire tendre la structure globale de l'organisation vers la forme recherchée.

Afin de pouvoir manipuler les formes qui décrivent la structure et la dynamique de l'organisation, il est souhaitable que l'espace de ces formes soit doté d'une métrique, ou du moins d'une topologie. Cette métrique sera utile pour faire de la classification, de l'apprentissage, contrôler la convergence du comportement. Si la forme est construite à partir d'un nuage de points, on pourra par exemple s'appuyer sur des méthodes statistiques (ou mécaniques) pour faire une comparaison [MH87]. Si le type de forme est fondé sur des graphes, il existe des algorithmes de comparaison de graphes, mais les calculs sont plus ardus. En effet, la comparaison de graphe [MB00], ou le calcul d'isomorphisme de graphes ou de sous-graphes, sont des problèmes difficiles et gourmands en ressources.

A partir de la métrique, on peut aisément définir une classe d'équivalence, en posant comme définition d'équivalence : deux formes A et B sont e-équivalentes si $d(A,B) < e$.

Sinon, s'il s'avère très difficile d'avoir une pseudo-métrique, on pourrait se contenter de la définition d'une relation d'équivalence entre deux formes. Cela donnera un moyen minimal pour classer les formes.

En ce qui concerne nos travaux, la forme sera construite à partir d'un nuage de points.

6.6 Le contrôle du système : l'organisation des objectifs

Nous avons vu précédemment que le rôle de la morphologie est de diriger le système vers une forme spécifique pour atteindre un objectif défini. La communication entre agents cognitifs et agents de morphologie est continuelle. La morpho-

logie permet de vérifier si la forme du système est bien celle attendue. En fonction de la situation, la morphologie accroît ou décroît l'importance de certains groupes d'agents.

Les différentes connaissances présentes dans l'ontologie prennent la forme de rôles attribués aux agents Structurants, ce sont des rôles cognitifs. Les agents vont se baser sur ces rôles pour s'activer. L'organisation du SAA n'est pas aléatoire, elle respecte un protocole statique spécifique régulé par un certain nombre de contraintes dirigeant l'organisation vers une forme attendue. Afin que le SAA soit sous contrôle, il faut que l'utilisateur le configure pour lui donner un certain nombre d'objectifs à atteindre. L'utilisateur pourra suivre la réalisation des objectifs en temps réel grâce au monitoring présenté à la section 8.3.

Nous avons vu dans la section 6.4 concernant l'organisation du SMA qu'il existait différents types d'agents utilisant la même architecture de base appelée Aspectuelle. Le type d'agents qui utilisera les connaissances, ajoutées par l'utilisateur dans l'ontologie, sous forme de rôles est le type Structurant. Celui-ci va permettre de gérer tout les rôles cognitifs pendant le fonctionnement du SAA. Tous ces rôles cognitifs vont être représentés par un Graphe Ontologique Global (GOG) totalement dynamique dans le nombre des nœuds, du placement et du nombre de ses liens. Mais la section 6.4 ne présente en aucun cas le moyen de diriger l'organisation du SAA de manière à contrôler le système.

Le SAA réalise une boucle systémique qui évolue dans le temps et ne s'arrête jamais. Il doit donc s'auto-organiser afin d'évoluer de manière intelligente dans son environnement. Pour cela, il a besoin d'une base opérationnelle de comparaison qui modifie automatiquement ses paramètres. Un nouveau type d'agents, toujours sur la base Aspectuelle, va voir le jour afin de produire cette base opérationnelle pour orienter l'organisation s'effectuant entre les agents Structurants et leurs rôles cognitifs. Ce nouveau type s'appelle Pattern d'Evènements Spécifiques (PES), et pourra gérer les éléments cognitifs concernant les différents objectifs que devra atteindre le SAA. Un objectif est une attente de résultat venant d'un acte délibéré et donnant une configuration particulière à l'environnement en total accord avec l'objectif. L'utilisateur enrichira donc l'ontologie avec les éléments d'objectifs qui prendront la forme d'une combinaison spécifique de connaissances, qu'elles soient physiques ou mentales.

La combinaison d'objectifs doit être logique et doit utiliser des éléments de connaissance présents dans l'ontologie. Si les éléments ne sont pas présents, le SAA essaiera d'utiliser certaines connaissances connues afin de définir plus précisément son objectif, ce point est abordé précisément dans la section 10.1. Les objectifs seront donc gérés par les agents PES et prendront la forme d'un graphe appelé graphe d'objectifs où les nœuds représenteront des rôles cognitifs et les liens la distance souhaitable entre ces rôles. Ce graphe correspondra à une forme particu-

lière que pourra prendre l'organisation du SAA. La figure 6.25 montre un objectif devant être atteint par le SAA. Une synchronisation entre le graphe d'objectifs, qui est la base opérationnelle de comparaison, et la forme générale que prend l'organisation du SAA à la lecture des données d'entrée correspondant aux paramètres de son environnement, doit s'effectuer afin d'orienter le comportement du système. Aucun parcours de graphe ou de comparaison ne sont effectués ici, les agents sont totalement autonomes, ils se présentent donc d'eux mêmes si l'environnement est favorable, ce qui évite de parcourir un nombre important de solutions possibles. Les statistiques de l'organisation permettent l'orientation du système auto-adaptatif.

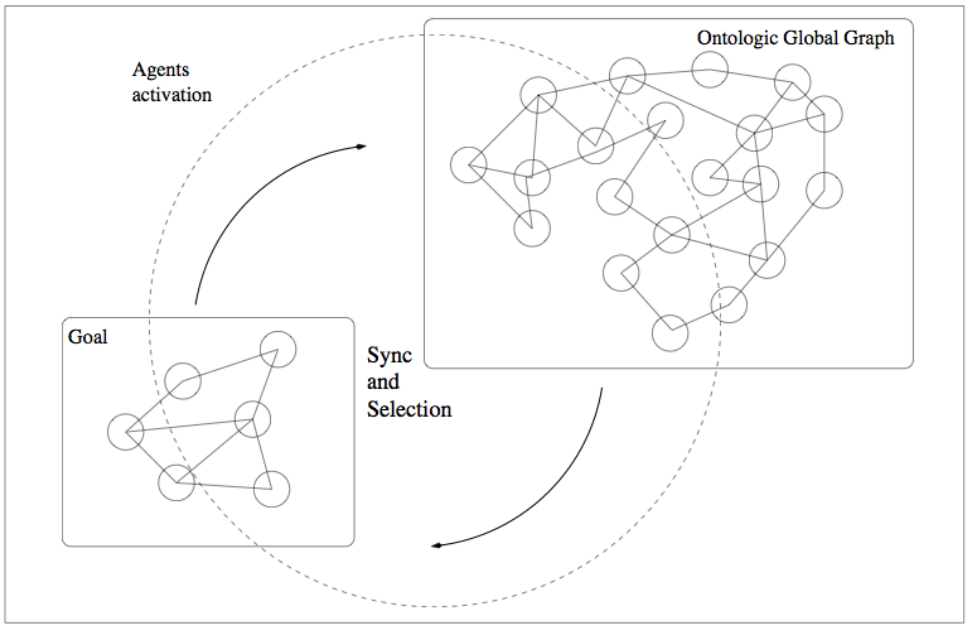

FIG. 6.25 – Sélection d'un objectif dans une activation générale du Graphe Ontologique Global. Il faut que la forme de l'objectif corresponde à la forme générale du système.

Chapitre 7

Implémentation du système

7.1 Introduction

Notre objectif est de nous rapprocher d'une organisation similaire à celle du cerveau d'un être humain pour traiter les données et prendre des décisions adaptées à une situation changeante. Pour atteindre cet objectif, nous basons notre système sur la théorie des systèmes complexes et les systèmes multi-agents. Cette méthode rejoint l'idée du Professeur Marvin Minsky [Min88] pour laquelle l'esprit serait composé d'une société de très nombreux éléments conceptuels jouant différents rôles. La différence ici est que l'objectif est de construire un système copiant le comportement humain, et non faire de l'observation pour comprendre le cerveau. Mais comment construire effectivement un tel système ? Comment choisir les composants de niveau programmation ? Il existe de multiples langages en informatique, mais un langage intéressant aura les caractéristiques suivantes : la programmation concurrente, la programmation logique, le design pattern et la programmation fonctionnelle. Ces caractéristiques permettent de construire un système adaptatif avec la composition d'entités capables de communiquer et de jouer différents rôles dans le temps. Le système Oz/Mozart possède toutes ces caractéristiques, c'est un langage multi-paradigme permettant le développement objet, la programmation logique, la programmation par contraintes, la programmation fonctionnelle et le scripting. Afin de respecter la contrainte du contrôle en temps réel, un choix de robot a aussi été fait. Celui-ci s'est porté sur le robot Aibo de Sony permettant un nombre de comportements important dans des environnement pouvant changer de contexte. Ce robot est totalement contrôlé par nos soins, seul les comportements physiques d'actions dans un environnement ont été gardés.

7.2 Matériel utilisé : l'Aibo de Sony

Ce robot prend la forme d'un petit chien capable d'imiter sensiblement les comportements d'un animal [KMK+04]. Dans sa nouvelle version, le ER7, l'Aibo joue aussi le rôle d'un juke-box, d'un organiseur, d'un réveil, d'un gestionnaire

de photos. Le plus intéressant dans ce petit robot est sa capacité à effectuer des mouvements complexes grâce à ses multiples capteurs et effecteurs.

7.2.1 Capteurs et effecteurs

Ceux-ci sont multiples et permettent à l'Aibo d'avoir un panel de comportements important face à des situations préprogrammées. Voici une liste non exhaustive des capteurs présents sur le robot :

- moteurs des quatre pattes,
- moteur de la tête,
- moteurs de la queue et des oreilles,
- caméra,
- capteurs sensitifs sur la tête et sur le dos,
- capteurs sensitifs sous les pattes,
- capteur de position dans l'espace,
- capteur de distance,
- sonars.

En tout, l'Aibo possède plus d'une trentaine de capteurs, moteurs lui permettant de faire des mouvements fluides et de se comporter dans un environnement quelconque.

7.2.2 Vers une autonomie adaptative

Rendre un Aibo autonome et adaptatif est un chemin complexe où plusieurs points doivent être pris en compte :

- Le matériel embarqué n'est pas assez puissant pour avoir une autonomie en charge utile ;
- La synchronisation avec l'environnement est très complexe, ce qui entraîne une prise de décision qui n'est pas assez rapide.
- Une prise de décision tardive entraîne le non respect de l'objectif ;
- Un coût important d'investissement matériel si le robot ne satisfait pas son besoin primaire : sa survie.
- Un énorme flux d'informations à traiter pour réussir à prendre une décision.

Le travail d'autonomie sur un robot ou une population de robots mène à la modélisation d'un module permettant de faire une **prise de décision dynamique multicritère en temps réel.** Ce module de décision sera utilisé pour l'ensemble de la population de robots afin d'écarter le plus possible l'être humain utilisateur du processus de décision.

7.2.3 Vision réaliste de l'autonomie à court terme

L'autonomie de ces robots est très limitée car le matériel embarqué manque de puissance. Pour réussir à avoir une autonomie qui se rapproche le plus possible de l'être vivant, il faudrait user de calcul massivement distribué qui utiliserait le

modèle ou plutôt le méta-modèle[1] cité précédemment.
Il est difficile de penser à court terme que les robots pourront avoir une autonomie aussi évoluée que celle d'un être humain. Par contre, la télécommunication étant très efficace entre un Aibo et sa station de conduite grâce au wifi, il est envisageable de penser à un système distant, basé sur le méta-modèle de l'autonomie adaptative, qui serait capable de rendre un robot et même une formation de robots, totalement autonome, en n'ayant qu'un point faible : la communication. Ce point faible disparaîtra grâce aux dernières avancées des micro-processeurs, qui permettront une puissance de calcul phénoménale[2] dans un espace totalement réduit.

7.3 Langage utilisé

Le langage employé pour développer les système auto-adaptatif est le "Oz", le langage de développement du système Oz/Mozart. "Mozart " est le nom du compilateur, le Oz est un langage"byte compiler", il n'existe aucune compilation native. Les caractéristique du langage employé pour faire fonctionner le système auto-adaptatif sont :

- La programmation objet.
- La programmation fonctionnelle.
- La programmation par concurrence.
- Le design pattern.

Une présentation plus détaillé de Oz/Mozart se trouve en Annex E.

7.4 Réalisation du système

7.4.1 Modèle conceptuel de donnée pour l'ontologie

Il existe une différence fondamentale entre le traitement de la connaissance et sa sauvegarde. Le traitement permet d'utiliser des paramètres associés à la connaissance afin de lui donner une dynamique spécifique par rapport à une utilisation particulière, c'est à dire, utilisé ou non pour atteindre un objectif. La sauvegarde permet de garder non seulement la connaissance mais aussi ses paramètres sur une mémoire de masse. Vu qu'il est difficile, voir impossible, de penser à tous les cas possibles en ce qui concerne un système autonome évoluant dans un environnement pour créer un algorithme adapté, il est préférable de penser à un niveau supérieur afin d'être capable de générer automatiquement cet algorithme qui changera très souvent au fil du temps pour s'adapter à un contexte. Cette section présente le Modèle Conceptuel de Données permettant de sauvegarder toutes les informations

[1] Etant donné qu'un modèle doit figer l'organisation d'une entité, employer ce mot ici est un abus de langage. Le mot composé méta-modèle semblerait plus convenir, car il représente le début d'une architecture interne qui se construira lors de l'exécution et du traitement des informations.

[2] Il est intéressant ici de penser quelle seraient les performances d'un processeur se basant sur le méta-modèle de l'autonomie adaptative. La plupart des puces électroniques ont été pensée et réalisée algorithmiquement avant d'avoir une existence matérielle.

utiles à la création de la planification dynamique de processus.

Le Modèle Conceptuel de Données est représenté sur la figure 7.1. Il est entendu que le modèle conçu possède différents types d'agents permettant d'architecturer le fonctionnement du programme de manière dynamique. Certains de ces types doivent être configurés par l'utilisateur, d'autres sont générés automatiquement en s'adaptant à la configuration. Voici les types d'agents générés automatiquement :

- L'agent Analyse.
- L'agent Morphologie.

Les paramètres de génération de ces agents correspondent au nombre d'agents Structurants ainsi qu'au nombre d'expériences entrées dans le système. De cette manière, il est possible de ressortir l'expérience passée qui a été utilisée afin d'atteindre un objectif. Il est donc important à l'utilisateur de travailler avec une méthode cognitive afin de donner une logique aux connaissances et aux objectifs qu'il entre dans le système. Voici les éléments à configurer par l'utilisateur :

- L'agent Aspectuel.
- L'agent Structurant.
- L'agent PES.
- L'expérience.

L'agent Aspectuel représente la structure de base de l'agent utilisé pour construire le SMA, puis le SAA. Nous avons deux types de base Aspectuel : l'agent Structurant, et l'agent PES. Nous retrouvons donc une table "agents_types", qui permet de classer les agents en fonction de leur type, c'est à dire Structurant ou PES, et une table "agents" qui permet d'identifier et de sauvegarder toutes les informations liées a un agent. Il y a ici deux partie à séparer : les tables et les attributs concernant les agents Aspectuels, et les tables et les attributs concernant les rôles attribués aux agents. Concernant les agents Aspectuels, voici le éléments à utiliser :

- La table "params" et sa relation avec la table "agents".
- La table "params_types" et sa relation avec la table "agents".
- La table "params_title" et sa relation avec la table "params".

Chaque agent possède des paramètres afin d'être traité par le système et utilisé par le SAA. Afin de ne pas se limiter aux types d'un seul langage, tous les types utilisés par un agent ont été créés et sauvegardés dans la base de données. Les tables et les relations citées précédemment permettent d'ajouter ou d'enlever des types de paramètres en fonction de l'évolution du système. Prendre cela en compte au niveau du Modèle Conceptuel de Données permet d'économiser un temps important dans le développement et surtout, permet au système d'être réellement adaptatif dans le temps en créant de nouveaux types utiles pour une ou plusieurs entités, pour agir sur l'environnement ou percevoir celui-ci.

Concernant les rôles, voici les tables, et leur association avec la table "agents", qui sont à prendre en compte :

- La table "class", permettant de classifier tous les rôles présents dans la base

(verbes, objets, couleurs, personnes, ...).
- La table “accointances”, permettant de lier certains rôles entre eux et de donner une valeur à chaque lien, ce qui crait le Graphe Ontologique Global (GOG).
- La table “expériences”, permettant d’ajouter un graphe ontologique nommé dans le GOG.

Un agent possède donc un rôle qui est classifié, possédant des liens évalués avec d’autres rôles, représentant un graphe nommé permettant de retrouver l’origine d’une décision. Les seuls rôles qui ne font pas partie du GOG sont les rôles attribués aux agents PES permettant de donner différents objectifs aux SAA. Ces rôles PES sont représentés par un graphe d’objectif qui peut évoluer dans le temps et qui permet d’agir sur la dynamique et l’organisation du SAA afin de respecter les différents objectifs passés en paramètre.

7.4.2 Classes et messages

7.4.2.1 Gestion des senseurs et des effecteurs

Dans le cas d’un robot ou même celui d’un logiciel, les senseurs et les effecteurs lient les entités à leurs environnement. Ils permettent de récupérer les informations et les interprètes pour ensuite envoyer d’autres informations dans le but de modifier l’environnement. Dans le cas d’un robot, les senseurs prendront la forme de capteurs, que ce soit de distance, de toucher, de vision, d’écoute et bien d’autres. Les effecteurs, quant à eux, prendront la forme de moteurs pouvant agir sur les objets de l’environnement ou des synthétiseurs de voix permettant de communiquer de manière langagière par exemple. Dans le cas d’un logiciel, les senseurs prennent la forme de tous les évènements et actions que peut faire les utilisateurs comme les clics de souris, les cases à cochers, les listes déroulantes, les zones de textes et bien d’autres. Ici, les effecteurs seront représentés par tous les évènements et résultats qui seront envoyés par et pour les utilisateurs.

Il est donc bien entendu qu’une entité, dans sa définition conceptuelle, peut être matérielle ou logicielle et en possession de senseurs et d’effecteurs hétérogènes permettant son évolution dans un environnement. Afin d’être le plus adaptatif possible, un modèle de gestion de capteurs a été conçue et développé, il comprends trois classes de fonctionnement essentielles et une classe utilisatrice comme montré sur la figure 7.2 :
- CSensor : cette classe permet de représenter un senseur ou un effecteur de manière générique.
- CMsg : cette classe permet de représenter génériquement un message interne au système lié aux différents capteurs de l’entité.
- CNet : cette classe permet la non embarcation du système auto-adaptatif en travaillant avec le protocole TCP/IP.
- La classe utilisatrice : CEntity qui permet la représentation générique d’une

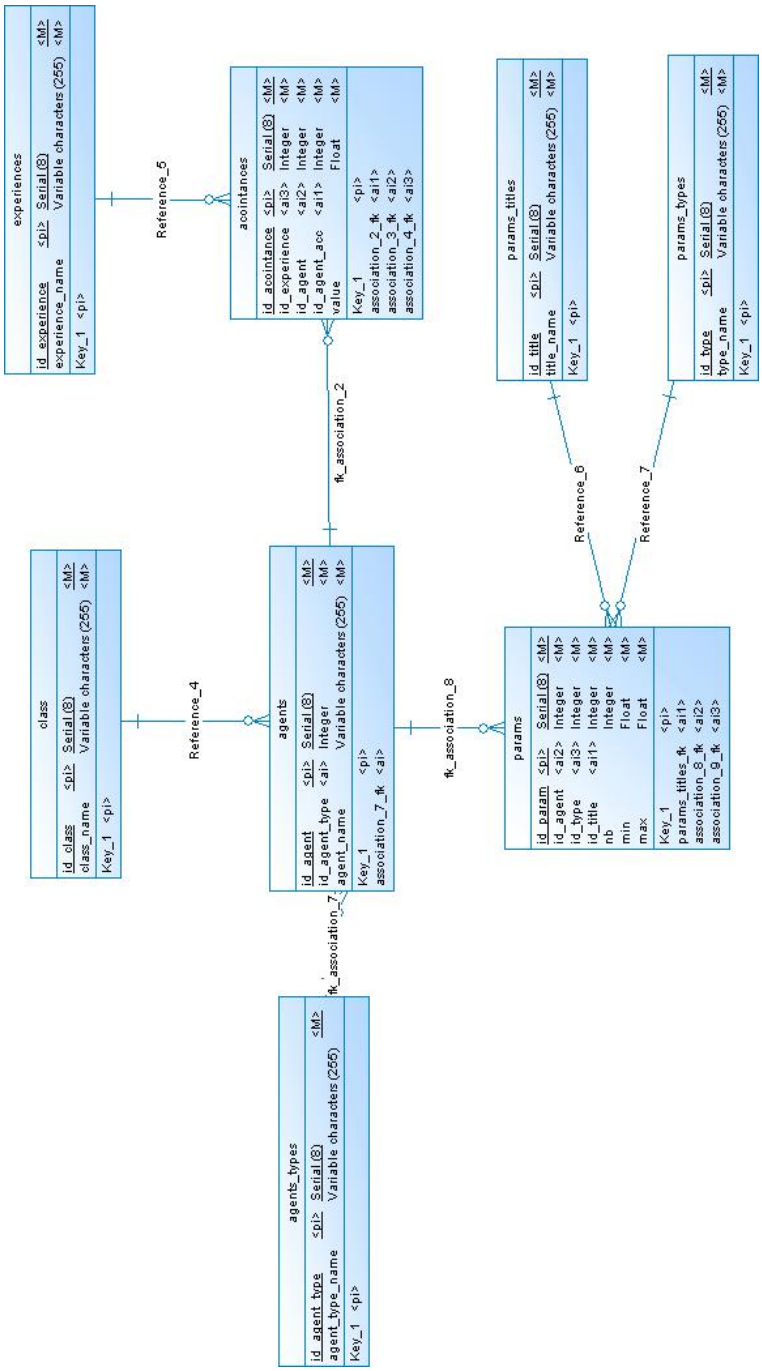

FIG. 7.1 – Modèle conceptuel de données pour l'ontologie

entité matérielle ou logicielle en se basant sur la définition qu'on lui en fait, c'est à dire, le nombre de systèmes auto-adaptatifs, le nombre de senseurs, le

nombre d'effecteurs, ainsi que leur description successive regroupant l'ontologie et les spécifications techniques des capteurs.

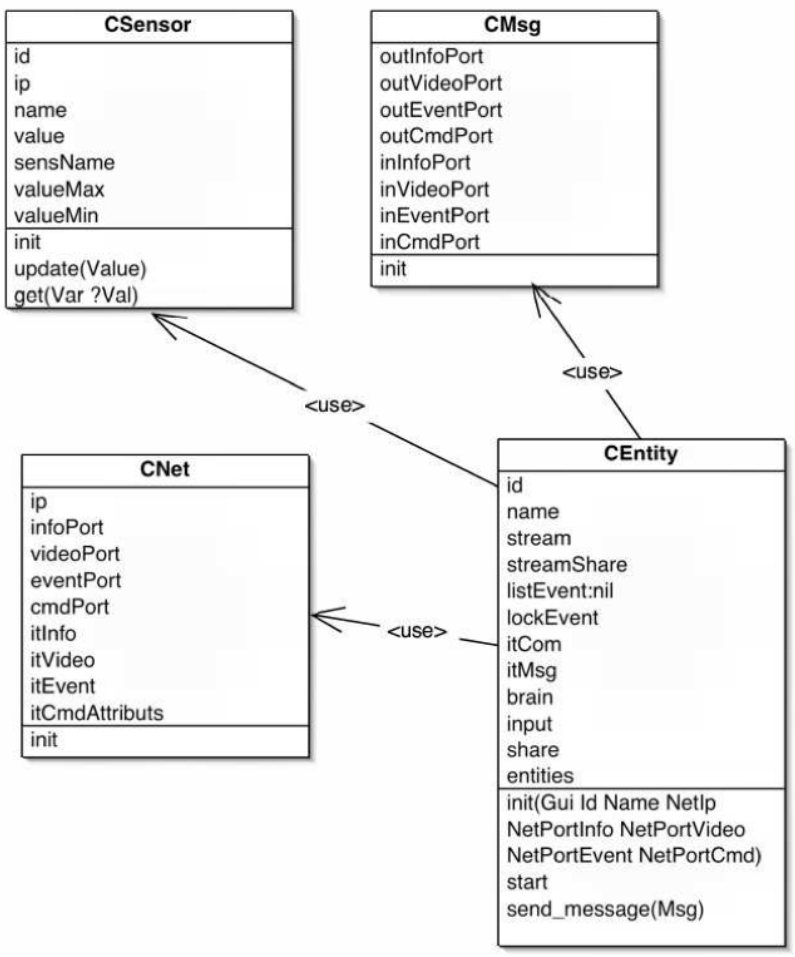

FIG. 7.2 – Diagramme de classes concernant la gestion des capteurs.

La classe CSensor possède différents attributs et méthodes lui permettant de gérer et de traiter ses données. Voici ses attributs :

- id : identifiant du capteur.
- ip : adresse ip du capteur dans le cas ou le système n'est pas embarqué.
- name : le nom attribué au capteur.
- value : la valeur du capteur.
- sensName : le nom officiel du capteur.
- valueMax : la valeur maximal que peut prendre le capteur.
- valueMin : la valeur minimal que peut prendre le capteur.

– itCom : abstraction du moyen de communication interne du capteur.

Voici ses méthodes :

– init : initialisation du capteur.
– update : mise à jour du capteur.
– get : récupération générique d'attributs.

La classe CMsg possède principalement des attributs car le traitement des données se fait de manière externe à la classe. Voici les attributs de CMsg :

– outInfoPort : port d'envoi de messages d'information.
– outVideoPort : port d'envoi de messages liés à la vidéo.
– outEventPort : port d'envoi de messages liés aux évènement.
– outCmdPort : port d'envoi de messages liés aux capacités physiques.
– inInfoPort : port de récupération de messages génériques.
– inVideoPort : port de récupération de messages liés à la vidéo.
– inEventPort : port de récupération de messages liés aux évènement.
– inCmdPort : port de récupération de messages liés aux capacités physiques.

Tout comme la classe précédente, le traitement des données de la classe CNet se fait de manière externe. Voici les attributs de CNet :

– ip : adresse ip de l'entité matérielle ou logicielle.
– infoPort : port concernant tous les capteurs.
– videoPort : port pour la vidéo.
– eventPort : port pour les évènements.
– cmdPort : port pour les commandes envoyé à l'entité.
– itInfo : informations générales.
– itVideo : information sur la vidéo.
– itEvent : information sur les évènements.
– itCmd : informations sur les commandes.

7.4.2.2 Création des entités pour groupes de robots

Le système est capable de gérer plusieurs entités hétérogènes en parallèle dans le but de créer une cohésion et une décision de groupe. Pour ce faire, un grand travail d'abstraction a été effectué sur les capacités physiques et ontologiques des entités, grâce à de la génération de code. La figure 7.3 montre un extrait du diagramme de classes permettant de gérer la création des entités. Toutes les classes ont un nom commençant par la lettre “C” sauf les classes abstraites donnant, en aval de la génération, une suite de classes paramétrées permettant une adaptation structurelle à l'ontologie.

Les classes n'ayant pas besoin d'abstraction pour modifier les attributs ainsi que les méthodes pour une construction ontologique, sont au nombre de cinq :

1. CSim : cette classe est le premier niveau du système. Elle permet de créer les entités en créant entre elle un canal de communication permettant d'échanger des informations, des connaissances ou des interprétations sémantiques.

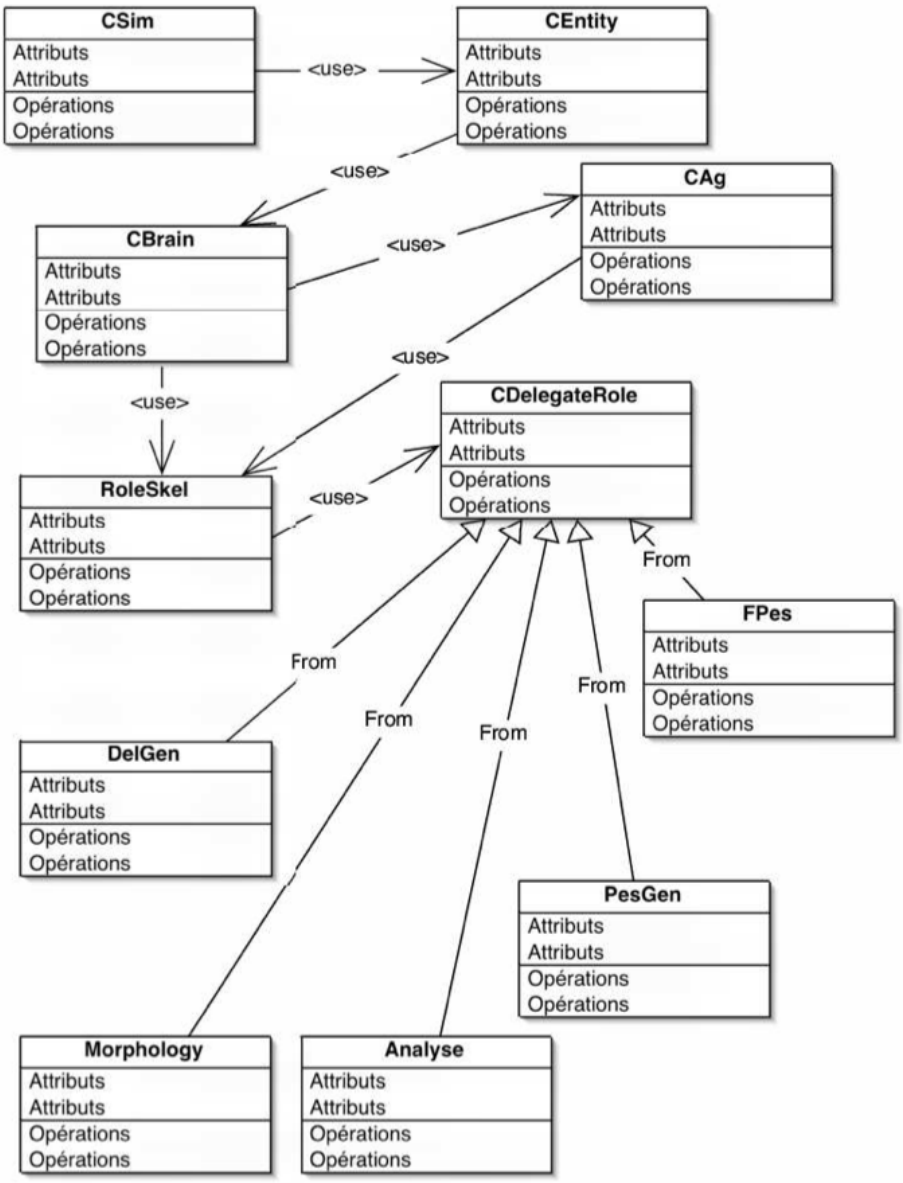

FIG. 7.3 – Extrait du diagramme de classes concernant la création des entités.

2. CEntity : cette classe permet de créer une abstraction d'une entité matérielle ou logicielle et de donner la possibilité de créer des groupes hétérogènes physiquement et ontologiquement. Elle s'occupe de relier les informations des capteurs au système de traitement des connaissance.
3. CBrain : cette classe permet de créer et de gérer tous les agents utiles au bon fonctionnement du système en temps-réel. Elle permet de garder une liste de

référence concernant tous les agents pour le bon transit des différents types de messages en ayant la possibilité de choisir le type d'accès, que ce soit par nom, par identifiant, par type ou par type de classification ontologique. Chaque type d'agent, que ce soit Structurant, de Morphologie, d'Analyse ou de Pattern est créé grâce à la méthode privée "Create" que l'on peut découvrir sur la figure 7.4.

4. CAg : cette classe permet de créer une abstraction de l'agent Aspectuel. Il est important de savoir qu'un agent n'est pas du tout représenté par un objet en mémoire lorsque celui-ci est en activité mais par plusieurs objets actifs en parallèle, de manière concurrente. Tous les paramètres d'un agent peuvent être modifié en temps-réel grâce à un système de messages. Cette capacité est représenté par la figure 7.5.
5. CDelegateRole : cette classe permet de donner une base structurelle, et non fonctionnelle, au rôle qui sera attribué à l'agent Aspectuel. Cette classe reste simple, comme le montre la figure 7.6, mais permet d'ajouter des spécificités fonctionnelles en fonction du type d'agent lors de la création en mémoire.

```
      meth Create(Role Label)
{Record.forAll
 {Record.map
  {Record.map
   {Dictionary.toRecord r Role}
   proc {$ O ?Rs} Rs = {RG.roleSkel O} end
   $}
  proc {$ O ?LR} {RG.det O LR} end
  $}
 proc {$ O}
    {List.forAllInd O
     proc {$ I O}
{Dictionary.put
 self.types.'aspectual_sens' (I + @nb_agents)
 {New CAg
  init( (I + @nb_agents)
self.type.as
self.sens
self Pag.Label O)}}
     end}
    nb_agents := @nb_agents + {List.length O}
 end}
     end
```

FIG. 7.4 – Création d'une suite d'agent correspondant à un type spécifique.

```
      meth Chanel_live()
 thread
    for Msg in @chanel_stream do
       case Msg
       of Var#Val then {self set(Var Val)}
       else
                    {Raise chanel_live__bad_arg(Msg)}
       end
    end
 end
      end
```

FIG. 7.5 – Modification en temps-réel des paramètres d'un agent Aspectuel.

Les classes paramétrables permettent d'adapter l'aspect structurel à l'aspect fonctionnel en fonction du type d'agent créé, ainsi qu'en fonction du rôle qui lui aura été attribué, ce qui permet d'augmenter le nombre de types différents ou de rôles, sans avoir à procéder à une écriture de code. Dans le système, il existe six classes paramétrables, générées lors de l'exécution du programme. Toutes ces classes, sauf "RoleSkel" héritent de "CDelegateRole" et sont englobées par "RoleSkel" pour la partie fonctionnelle :

1. RoleSkel : cette classe est le squelette fonctionnel d'un rôle qui sera attribué à un agent Aspectuel. Elle utilise la classe "CDelegateRole" afin d'avoir une structure de base. Le comportement de base d'un rôle est contenu dans cette classe. Cette configuration permet de travailler en quatre dimensions à la manière d'un hypercube : le cube recouvrant le cube plus petit est le squelette fonctionnel permettant au petit cube d'être connecté au monde.
2. DelGen : cette classe permet de créer un rôle cognitif qui sera attribué à un agent Aspectuel en héritant de "CDelagateRole", ce qui transformera l'agent en agent Structurant. Elle sera toujours englobée dans la classe "RoleSkel" afin d'avoir un comportement qui soit adapté au système multi-agent.
3. Morphology : cette classe représente le rôle morphologique. Le nombre d'agents de Morphologie variera en fonction du nombre d'expériences qui seront entrées dans l'ontologie. Cette classe permet d'effectuer une analyse externe sur l'organisation du SAA.
4. Analyse : cette classe permet de créer un graphe d'état du SAA qui est échelonné dans le temps et mis à jour en temps-réel, en se basant sur les résultats de la morphologie. Le nombre d'agent d'analyse dépend aussi du nombre d'expériences entrées dans l'ontologie.
5. PesGen : cette classe permet de créer des rôles représentant les différents objectifs du SAA, les patterns d'évènements. Ces objectifs prennent la forme de différents graphes qui devront être retrouvé dans l'activité du SAA.

6. FPes : cette classe permet de planifier et de gérer les priorités des différents objectifs du SAA. Certains objectifs sont activés et d'autres désactivés en fonction du court, du long terme ou de la survie.

7.4.2.3 Distribution de l'ontologie et paramétrage des rôles

Tous les éléments contenus dans l'ontologie sont sauvegardés dans une base de données, celle-ci a été présenté à la section 7.4.1. La classe "COnto2agent" donne la possibilité à "CBrain" de récupérer les informations concernant l'ontologie pour les distribué dans un SMA. Pour ce faire, "COnto2agent" possède quatre attributs cruciaux à la récupération des données :

- sqlExpPerAgent : cet attribut permet de récupérer toutes les expériences qui sont liées à un rôle.
- sqlRolesSelected : cet attribut permet de récupérer les paramètres des rôles passés en paramètre.
- sqlParamsSelected : cet attribut permet de récupérer tous les paramètres liés aux différents rôles.
- sqlAccointancesSelected : cet attribut permet de récupérer toutes les accointances présentes dans l'ontologie. Le résultat est ensuite utilisé pour appliquer une nouvelle requête, concernant les rôles, pour avoir toutes les accointances associées à un rôle.

La classe "COnto2agent" donne donc plusieurs possibilités d'utilisation en proposant une suite de méthodes associées :

- init() : initialise la classe lors de la création de l'objet.
- getExpListName(?R) : retourne toutes les expériences contenus dans l'ontologie.
- GetExpPerAgent(Role ?R) : retourne les expériences par rôle, c'est une méthode privée, d'où la majuscule à la première lettre du nom.
- getExpListAgent(?R) : retourne la liste des rôles par expérience.
- getSelectedRole(Type L ?R) : retourne la liste des paramètres des rôles passés en paramètre.
- getRoleList(Type ?R) : retourne la liste complète des rôles présents dans l'ontologie.
- get(Attr ?R) : méthode générique de récupération d'attributs.

La classe "COnto2agent" est en faite une classe permettant au langage Oz d'avoir un module de lecture SQL, qui est développé pour l'instant pour le Système de Gestion de Base de Données MySql. La figure 7.7 montre un exemple de fonctionnement interne en ce qui concerne la récupération des expériences par rôle. Le choix de mettre le mot "Agent" dans le nom des méthodes a été fait car l'utilisation de ce module est fait au niveau agent, ce qui évite les confusions.

La récupération des rôles va permettre de construire le SMA qui sera la base du SAA, les paramètres des différents rôles définirons le nombre d'agents qui leur seront attribués. Afin de rester le plus évolutif possible, les paramètres des rôles

n'emploient pas des types existant spécifiquement pour un langage, ces types sont créés de toutes pièces et sont sauvegardés dans l'ontologie comme présenté dans la section 7.4.1. Si au cour d'une expérimentation concernant un cas spécifique des paramètres viendraient à manquer, il suffirait d'enrichir l'ontologie des types de ces paramètres. Voici les paramètres actuellement présents dans l'ontologie :

- list_float : ce type comprends un *dx* et un *dy* qui ont respectivement un *nb*, un *min* et un *max*. Le *nb* corresponds au nombre d'agents, le *min* a la valeur minimale pris par la variable et le *max* la valeur maximale prise par la variable. Les variable *dx* et *dy* servent à la morphologie pour l'axe des x et l'axe des y concernant la génération des histogrammes.
- list_int : ce type comprends un *delay* qui possède un *nb*, un *min* et un *max*. Le *nb* corresponds au nombre d'agents, le *min* a la valeur minimale prise par la variable et le *max* la valeur maximale prise par la variable. Le *delay* corresponds a la densité acceptée lors de la configuration concernant le timer.

Dans l'ontologie, il existe les rôles cognitifs qui sont représentés par des agents de types Structurant et des rôles d'objectifs qui sont représentés par les agents de type Pattern d'évènement, les PES. Ces rôles possède un paramétrage différent. Un rôle cognitif prend en paramètre deux list_float et un list_int comme le montre la figure 7.8, un rôle d'objectifs, lui, prend seulement un list_int, comme la montre la figure 7.9. La différence entre les deux est que la morphologie n'a pas besoin de mesurer l'activité des rôles d'objectifs qui sont, eux, gérés par les agents de type Pattern d'évènement fondamentaux.

7.4.2.4 Communication interne et types de messages

Les rôles attribués aux différents types d'agents utilisent des messages de communication de types hétérogènes afin de donner un sens à l'organisation global du système. Voici la liste des messages gérés de manière interne à chaque agent, ils sont aux nombres de onze :

1. morphoDescrHisto1 : ce type message permet aux agents de Morphologie d'envoyer des informations concernant la morphologie des agents Structurants aux agents d'Analyse.
2. morphoTargetHisto1 : ce type de messages permet aux agents Structurants d'envoyer des informations statistiques concernant leurs communication aux agents de Morphologie.
3. morphoTargetHisto1 : ce type de message permet aux agents d'analyse de préciser aux agents de morphologie la forme globale que doit prendre l'organisation du système pour satisfaire les objectifs.
4. focalPointGraph : ce type de message permet d'envoyer des informations concernant le point focal, c'est à dire "la représentation courante", aux outils de monitoring.
5. morphoOrderHisto1 : ce type de message permet aux agents de Morphologie de réguler les envois de messages des agents Structurants.

6. egraph : ce type de message permet aux agents d'Analyse de donner un graphe d'états, mis à jour, de l'organisation des agents aux agents de type PES.
7. robot_command : ce type de message permet aux agents Structurants d'envoyer des ordres aux différents capteurs de l'entité.
8. aspectDescr : ce type de messages permet de compléter les informations des agents de Morphologie envoyées aux agents d'analyse concernant l'aspect général de l'agent dans l'organisation.
9. aspectVect : ce type de messages permet aux agents Structurants d'envoyer des informations statistiques concernant l'envoie de messages aux agents de Morphologie.
10. preferredShape : ce type de message permet aux agents de Pattern contenant les objectifs de sélectionner une forme adaptée dans le graphe d'états.
11. pes_control : ce type de message permet aux agents de Pattern Fondementaux de planifier la réalisation des objectifs représentée par les agents de type PES.

La gestion des messages principaux sont gérés au niveau agent de manière concurrente, comme le montre la figure 7.10. Ils sont gérés à ce niveau pour donner la possibilité aux agents de changer de rôle de manière dynamique. Le message concernant l'auto-analyse statistique qui sera envoyé aux agents de Morphologie est géré grâce à un timer permettant de séquencer les envois. La figure 7.11 montre l'initialisation du timer, la création du message "aspectVect", ainsi que son envoi.

```
class CDelegateRole
      attr
 ag
 msgFrom

      meth winit()
 msgFrom := {Dictionary.new}
 {self init()}
      end

      meth get(X ?Out)
 Out = @X
      end

      meth set(X V)
 X := V
      end

      meth setAcointance(Name Value)
         [...]
      end

      meth destroy()
 skip
      end

      meth storeMsgFrom(Msg)
 if {Value.hasFeature Msg msgFrom}
 then
    AgId OldV NewV in
    AgId = Msg.msgFrom
    try {Dictionary.get @msgFrom AgId OldV}
    catch _ then OldV = 0 end
    NewV = OldV + 100
    {Dictionary.put @msgFrom AgId NewV}
 else
    skip
 end
      end

      meth getMsgFrom(?Out)
 {Dictionary.entries @msgFrom Out}
 {Dictionary.removeAll @msgFrom}
      end

   end
```


FIG. 7.6 – Structure de base d'un rôle traité par un agent Aspectuel.

```
      meth GetExpPerAgent(Role ?R)
 Exp = {NewCell nil} in
 {@sqlExpPerAgent fetchSQL("Select experience_name
           from experiences as exp, accointances as acc, agents
           as ag where exp.id_experience=acc.id_experience and
           (acc.id_agent=ag.id_agent or
           ag.id_agent = acc.id_agent_acc)
           and ag.agent_name='"#Role#"'
           group by experience_name")}
 {@sqlExpPerAgent procToAll(proc {$ A}
Exp := {Atom.toString
                                  A.experience_name $} | @Exp
     end)}
 R = @Exp
      end
```

FIG. 7.7 – Méthode de récupération des expériences par agent.

Params

Add new

id_param	Title	Type	nb	min	max	id_agent	Edit
2	list_float	dx	1	-90	90	1	Edit - Delete
3	list_float	dy	1	-90	90	1	Edit - Delete
1	list_int	delay	1	50	1000	1	Edit - Delete

FIG. 7.8 – Liste des paramètres pour un rôle cognitif.

Params

Add new

id_param	Title	Type	nb	min	max	id_agent	Edit
307	list_int	delay	1	1000	2000	103	Edit - Delete

FIG. 7.9 – Liste des paramètres pour un rôle d'objectif.

```
meth Internal_message()
 thread
    for Msg in @internal_stream do
       case Msg
       of morphoDescrHisto1(...)       then
                  {self Dispatch_analyze(Msg)}
       [] morphoTargetHisto1(...)      then
                  {self Dispatch_morpho(Msg)}
       [] focalPointGraph(...)         then
                  {self Dispatch_focal_point(Msg)}
       [] morphoOrderHisto1(orders:L ...) then
                  {self Dispatch_list_aspectual(L)}
                  {self Dispatch_msg_scan(Msg)}
       [] egraph(...)                  then
                  {self Dispatch_pes(all Msg)}
       [] robot_command(C)             then
                  {self Dispatch_robot(C)}
                  {self Dispatch_msg_scan(Msg)}
       [] aspectDescr(...)             then
                  {self Dispatch_analyze(Msg)}
       [] preferredShape(...)          then
                  {self Dispatch_analyze(Msg)}
       [] pes_control(pesId:Id ...)    then
                  {self Dispatch_pes(id(Id) Msg)}
       [] pesAspectualControl(...)     then
                  {self Dispatch_aspectual(Msg)}
       else
                  {self Try_dispatch_aspectual(Msg)}
               end
             end
          end
 end
```

FIG. 7.10 – Gestion interne des messages au niveau agent.

```
meth Init_timer()
 proc {PP}
    lock @csend_nbLock then
       Id = self.id
       AgRef = self % key
       F = {Int.toFloat @csend_nb} % value
       Msg = aspectVect( id    : Id
 agref : AgRef
 freq  : F
 role  : ((self.role).rolename)
 peslist: @pes_list
       )
    in
       if @csend_nb == 0 then
  countZeroCsend := @countZeroCsend + 1
       else
  {self Dispatch_morpho(Msg)}
  {Port.send self.brain.monitor.inputs.cGraph Msg}
  csend_nb := 0
       end
    end
 end
      in
 case self.type_name
 of 'Aspectual_sens' then
    t<-{New Time.repeat
              setRepAll(action:PP delay:({self.brain.gui
                getConfig($)}).fd)}
    thread {@t go()} end
 [] 'Classification' then
    t<-{New Time.repeat
              setRepAll(action:PP delay:({self.brain.gui
                getConfig($)}).fd)}
    thread {@t go()} end
 else skip
 end
 end
```

FIG. 7.11 – Auto-analyse statistique d'un agent au niveau de son envoi de messages.

Chapitre 8

Utilisation du système

8.1 Système et options d'utilisation

L'utilisation du système est associé à une base de données gérée par un gestionnaire de bases de données comme mysql, postgreSql ou Oracle. Cette base de donnée contient les informations correspondant à l'ontologie utilisée pour une problématique spécifique. Il y aura autant de bases de données que de cas applicatifs. L'utilisateur se devra de vérifier que l'ontologie est renseignée de manière adéquate afin que le programme puisse s'exécuter correctement. Il est à souligner que l'interface utilisateur pour renseigner l'ontologie est une page web : un serveur web est donc recommandé pour utiliser le système.

Voici les différentes options que propose le système :

- mode : il peut être de deux types, 'simulation' ou 'robot'. Cela précise la simulation ou non d'un environnement grâce à une interface utilisateur.
- conf : cette option prend un fichier de configuration au format Oz afin de décrire les entités utilisées par le programme dans le cas d'une gestion de groupe.
- name : spécifie le nom utilisé pour identifier une entité de type matériel ou logiciel. Ce nom sera complété par un numéro d'identifiant.
- db : spécifie le nom de la base de donnée à utiliser.
- host : spécifie l'adresse de la machine ou se trouve la base de données.
- ip : si nécessaire, spécifie l'adresse de la machine servant d'interface entre le logiciel et le matériel (comme par exemple l'Aibo).
- goals : spécifie la liste des objectifs à atteindre. Tous les objectifs seront renseignés en amont dans l'ontologie.
- md : spécifie le délai attribué à la morphologie. La valeur par défaut est de 3000 nanosecondes, utilisée sous Max Os X avec un G4 1,33 Ghz.
- ad : spécifie le délai attribué à l'analyse du système. La valeur par défaut est de 7000 millisecondes, utilisée sous Max Os X avec un G4 1,33 Ghz.
- fd : spécifie le délai de fréquence des messages internes concernant l'activa-

tion des agents. La valeur par défaut est de 7000 millisecondes, utilisée sous Max Os X avec un G4 1,33 Ghz.

Toutes ces options, a part 'conf', sont nécessaires au bon fonctionnement du logiciel. Cependant, la plupart des options possèdent des valeurs par défaut, ce qui offre une utilisation simple du logiciel.

8.2 Les ontologies

L'ontologie générale doit être renseignée soit par des cogniticiens, soit par l'utilisateur lui même. Celle-ci prend la forme d'un graphe évoluant dans le temps en fonction des expériences passées. L'interface à utiliser pour cette tâche est représentée sur la figure 8.1.

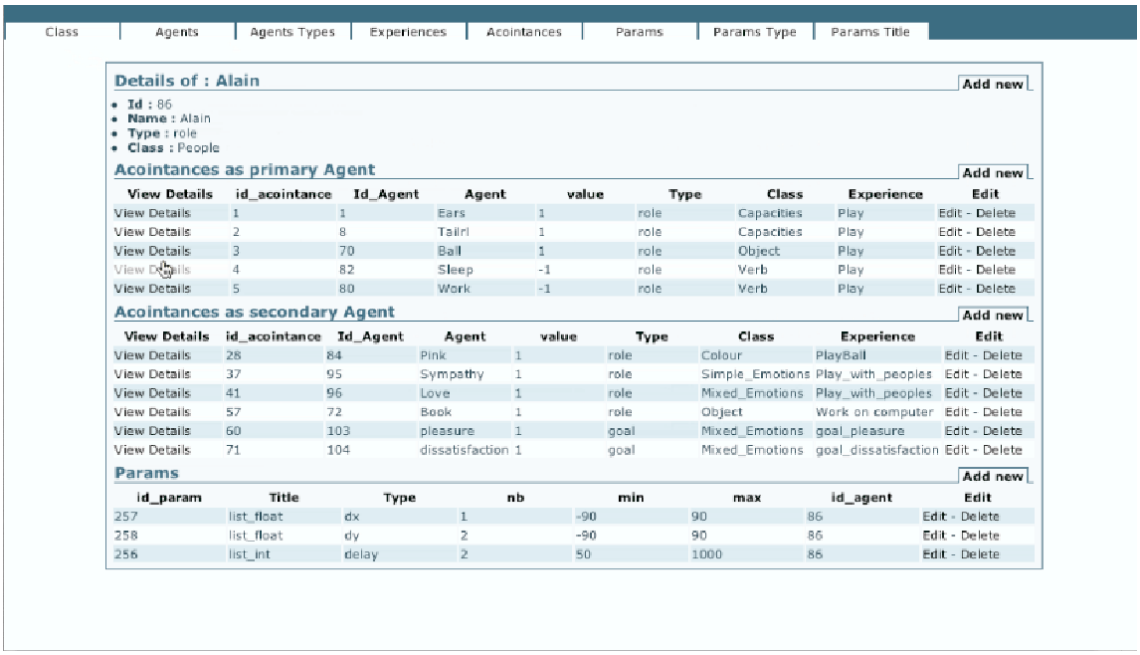

FIG. 8.1 – Fenêtre de configuration de l'ontologie. Celle-ci permet d'insérer la connaissance du système correspondant à un environnement et ses contextes associés

Voici l'ordre à respecter pour renseigner l'ontologie :

1. Aller dans la section "class" pour ajouter tous les noms de classification utilisés dans l'ontologie.
2. Aller dans la section "Experience" afin d'ajouter les noms des expériences utiles au commandement de l'entité.

3. Aller dans la section “Agents” pour ajouter les noms des agents correspondant aux connaissances devant être présentes dans l’ontologie. Pour chaque ajout, un nom de classification et un type d’agent est demandé. Ici, c’est le type “role” qui doit être utilisé, car il correspond au rôle cognitif dans l’ontologie.
4. Pour chaque agent ajouté, la partie “View Details” permet de voir les accointances de chaque agent, ainsi que ses paramètres. Il faut ajouter des paramètres à l’agent pour que celui-ci soit créé. Chaque agent a besoin de deux variables de type “list_float” (dx, dy) et une variable de type “list_int” (delay). Chaque variable comprend les labels nb, min et max. La variable “delay” correspond au minimum et au maximum en temps (en milliseconde) attribué à l’agent pour l’envoi des messages. Les variables “dx” et “dy” représente les valeurs minimum et maximum à utiliser en ce qui concerne la morphologie et la composition des histogrammes.
5. Dans “View Details”, il faut ajouter des accointances si besoin. Les accointances primaires correspondent à un lien direct, et les accointances secondaires à un lien indirect. Pour chaque création d’accointances, un nom d’expérience est demandé, ainsi que le nom de deux agents et une valeur comprise entre -1.0 et 1.0, l’élément positif représentant une grande proximité.
6. Afin d’insérer des objectifs dans l’ontologie, il suffit de créer des agent de types “goal” avec seulement un paramètre “delay” qui sera de type “list_int”. Cet agent devra posséder des accointances primaires.

Lorsque l’utilisateur lancera le programme, il précisera quels objectifs il veut atteindre grâce à l’option “goals”. Les section “Params”, Params Type” et “Params Title” ne sont pas à modifier dans un cas d’utilisation déjà étudié. Ces section permettent d’ajouter de nouveaux paramètres si les paramètres existant ne permette pas de traiter une problématique. La section “Accointances” quand à elle, résume toutes les accointances de l’ontologie. C’est aussi un moyen alternatif de créer de nouveaux liens.

8.3 Monitoring

Le monitoring du système ne se fait pas de manière linéaire, un bon nombre de fenêtres sont à utiliser pour suivre l’évolution de l’entité dans l’interprétation de son environnement et dans sa prise de décision. Cinq éléments important sont à suivre :

1. La communication entre les agents, représentée sur le figure 8.2, permet de s’assurer que le système génère des messages internes et permet d’avoir une vue d’ensemble sur l’activation des connaissances, c’est à dire la forme générale que prend le système lors d’une situation précise. Les différentes couleurs correspondent aux éléments de classification de l’ontologie.

2. Les plans ontologiques permettent de surveiller l'importance d'activation pour une connaissance en particulier. Les plans respectent la classification utilisée dans l'ontologie. Il est possible de demander n'importe quel plan en temps-réel. La figure 8.3 représente les plans concernant les objets et les verbes. Ces plans se déforment en fonction des entrées capteurs. Cela veut dire que certaines connaissances sont utiles à l'interprétation de ce qui se passe dans l'environnement.
3. L'activation des connaissances génère un flux de messages très important dans le système. Tous ces messages sont traités afin de créer le point focal courant de l'entité à un instant t. Un point focal est représenté sur la figure 8.4. Le placement des différents cercles n'a pas de signification particulière, cela aide seulement à la lisibilité du point focal. Les points représentent les agents et les grands cercles représentent leurs communautés. Plus un cercle est grand, plus la communauté est importante. Les différentes connaissances possèdent une distance topologique entre elles qui peut évoluer au cour du temps en fonction de l'expérience.
4. Chaque point focal débouchera sur un comportement spécifique de l'entité dans l'environnement. Le graphique qui est représenté sur la figure 8.5 permet de suivre l'évolution de l'exécution des objectifs passés en paramètre lors du lancement du logiciel et leur atteinte. Il y aura autant de courbe que d'objectifs à atteindre. Plus la courbe tend vers zéro, plus l'atteinte de l'objectif sera satisfaisant.
5. Dans le cas d'une entité matérielle ou logicielle, il est important de constater visuellement le comportement de cette entité en fonction de la description de la scène. La figure 8.6 prend l'exemple du comportement d'un Aibo lors d'une expérimentation concernant la robotique.

La création de toutes ces fenêtres est faite de manière automatique lors du lancement du système.

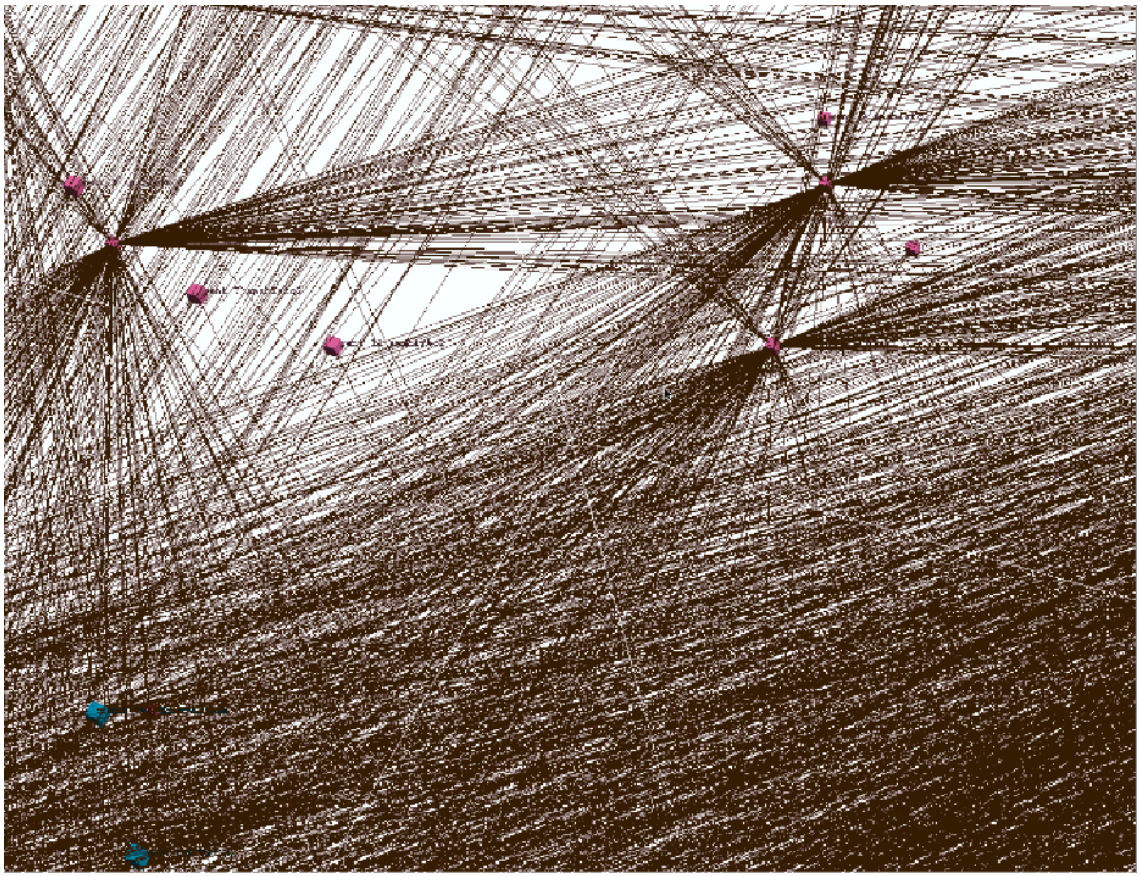

FIG. 8.2 – Communication entre agents durant un traitement d'information d'un contexte environnemental spécifique.

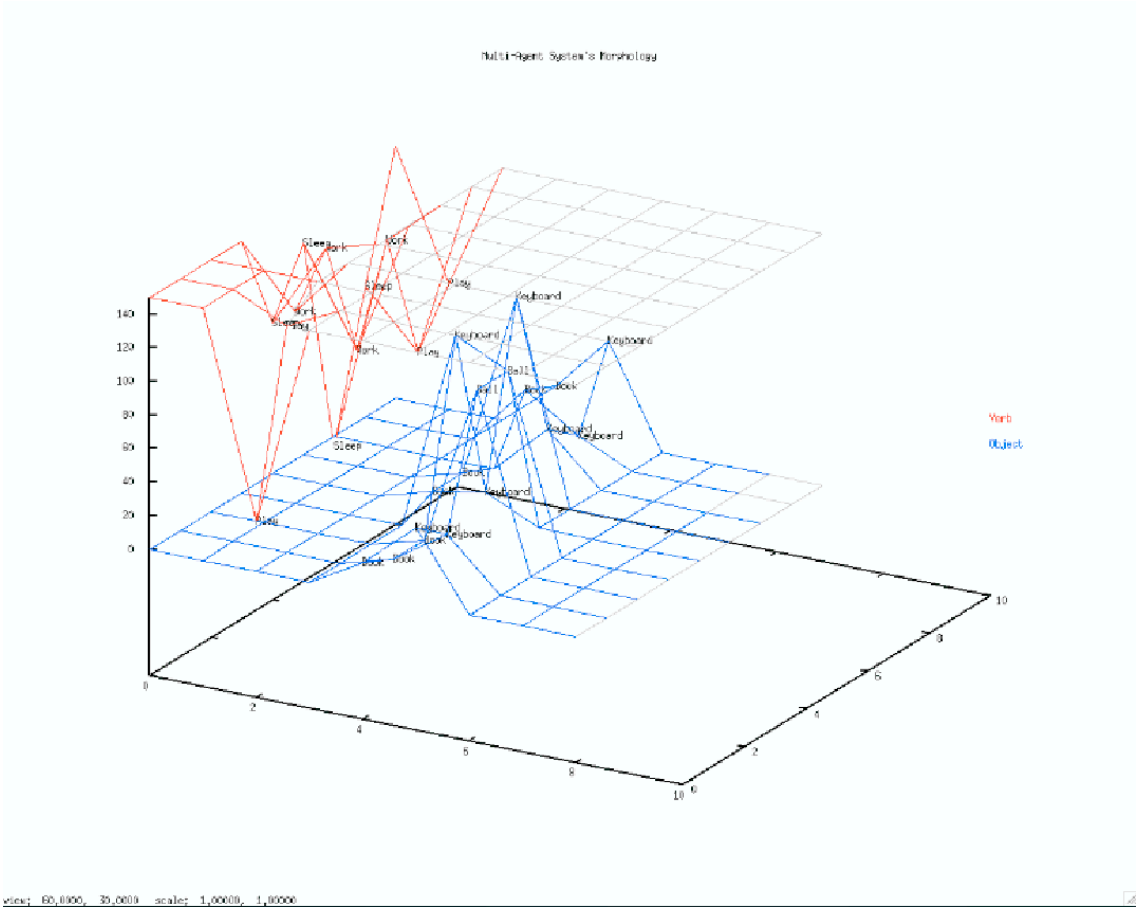

FIG. 8.3 – Le système est capable de structurer les connaissances en *n* dimensions ontologiques. Il est donc possible d'effectuer une découpe du graphe tri-dimensionnel ontologique par plan afin de constater l'activation de certaines connaissances du système. Cette découpe peut se faire en fonction du classement employé dans l'ontologie (objet, verbes, couleurs, ...).

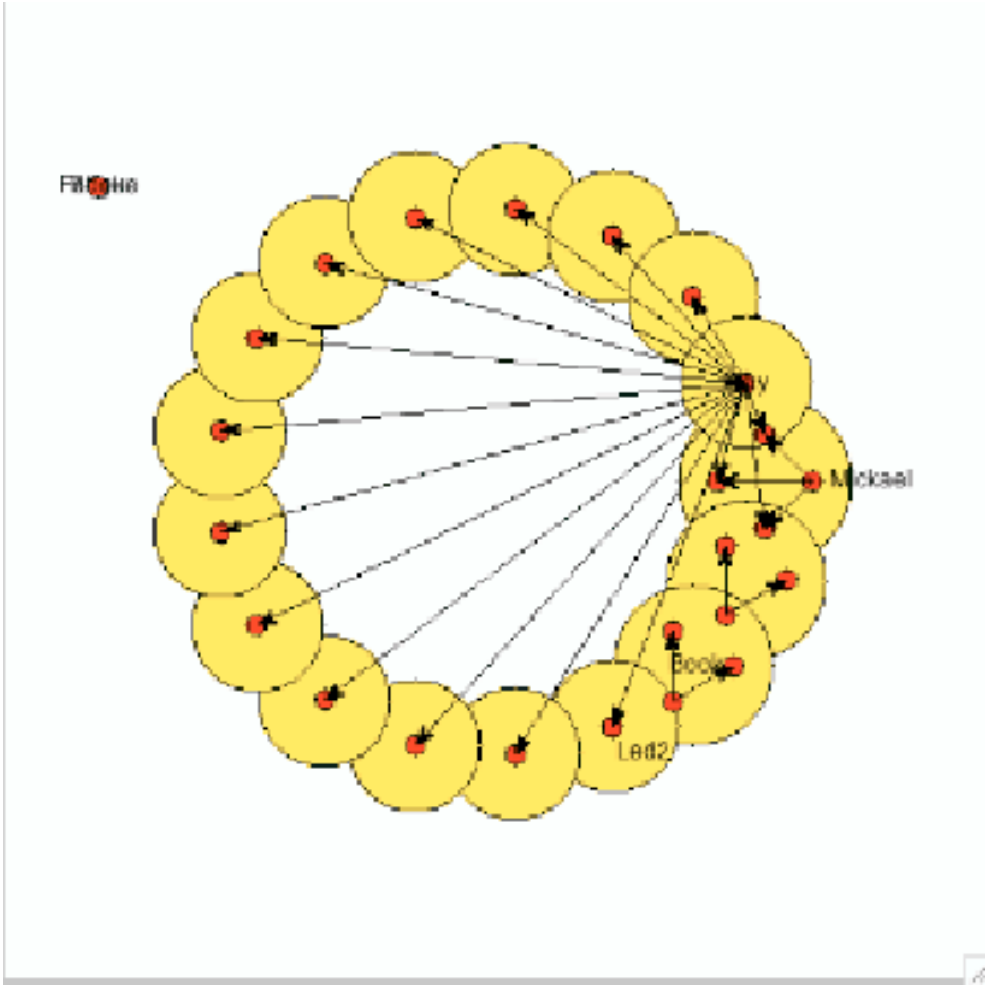

FIG. 8.4 – Il est possible à chaque instant de suivre la “représentation courante” de l’entité. Celle-ci évolue dans le temps et s’oriente vers des objectifs bien précis.

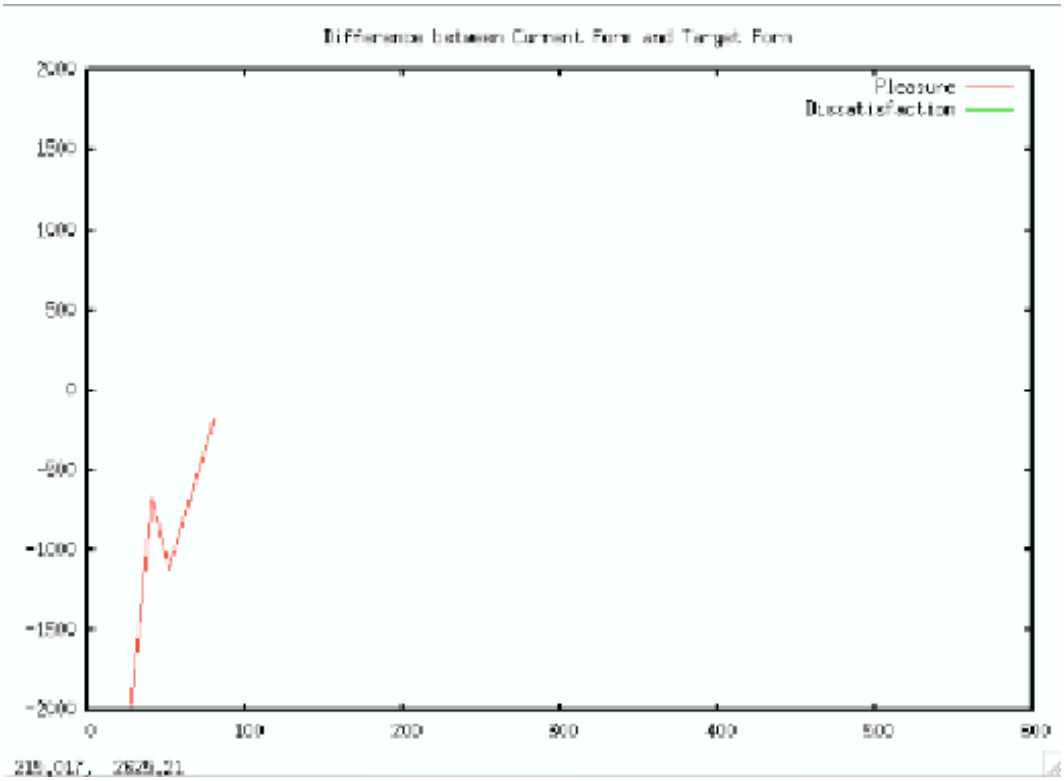

FIG. 8.5 – Le système fonctionne par analyse de formes géométriques que l'on appelle morphologie. Il est possible de donner au système des objectifs, et de suivre en temps-réel l'évolution de ceux-ci grâce à la différence entre la forme cible et la forme courante.

FIG. 8.6 – Le comportement de l'entité peut être suivi en temps réel. Il est important d'associer toutes les vues du monitoring afin de déceler des écart de comportement modifiables par un changement de paramètres.

Troisième partie

Réalisations et validation des hypothèses

Chapitre 9

Les différentes expériences et leurs résultats

Dans le cadre de nos recherches, une expérience est le fait d'appliquer une situation initiale, compatible avec le monde réel, au système auto-adaptatif. Les paramètres de ce dernier sont calibrés afin que l'étude du résultat se fasse le plus clairement possible (figure 9.1). Par exemple, dans le cas des expériences avec le robot Aibo, celui-ci est placé dans notre environnement. Ensuite, une balle lui est présentée pour activer ses connaissances. En fonction du paramétrage du système, le robot aura un comportement particulier qui sera en totale adéquation avec l'organisation des agents dans le système.

Chaque expérience développée dans ce chapitre doit montrer une caractéristique essentielle du système auto-adaptatif. Les logiciels utilisés et spécialement développés pour cette thèse sont présentés en annexe C. La mesure des résultats se fait grâce à la lecture de plusieurs graphiques et interfaces permettant d'utiliser et de conduire le système. Ceux-ci sont au nombre de quatre :

1. L'interface de suivi des messages : cette interface permet de s'assurer que les messages d'un certain type arrivent bien à destination, ce qui prouve le bon fonctionnement interne du système de communication.
2. Les graphiques des plans de classification ontologique : ces graphiques permettent de constater l'usage des connaissances employées par le système (en abscisses, un placement topologique en surface pour faciliter la lecture, en ordonnées la valeur d'activation des connaissances).
3. Le graphique du suivi des résultats : la courbe doit tendre vers zéro pour que le système s'oriente de manière satisfaisante vers les objectifs fixés (abscisses : temps en milliseconde, ordonnées : valeur de différence entre l'objectif et l'environnement courant).
4. L'interface de suivi des points focaux : elle décrit une représentation sémantique d'une scène d'un environnement.

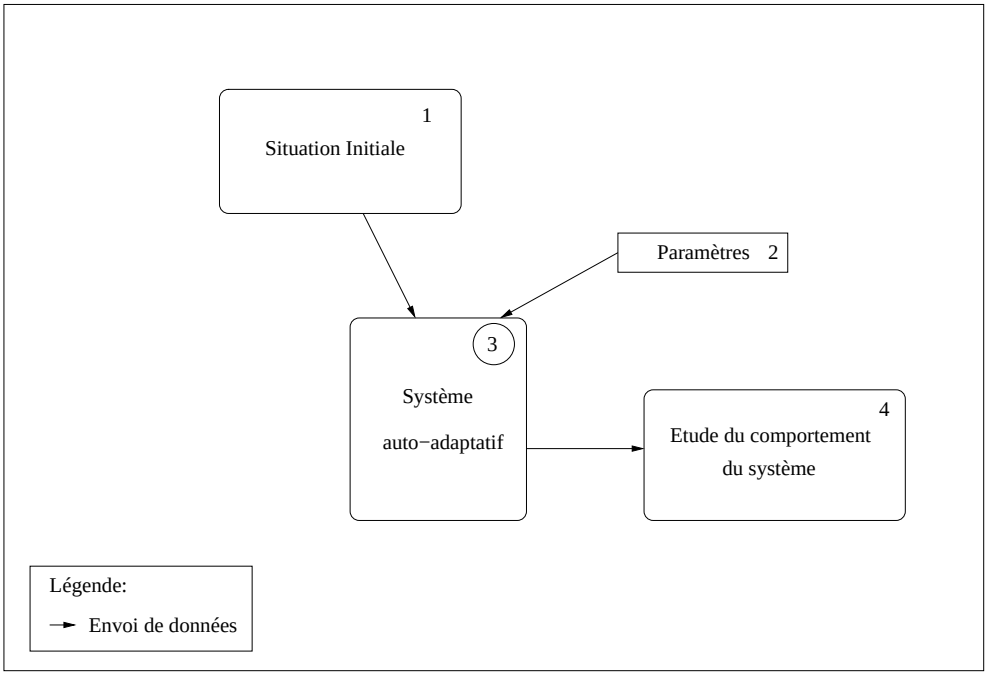

FIG. 9.1 – Schéma global d'une expérience.

Cette lecture doit se faire dans un même temps pour constater la bonne évolution du système en fonction des objectifs qui ont été passés en paramètres. Une bonne lecture permet de comprendre tous les actes que peut entreprendre le robot. Il est important de savoir qu'il ne s'agit pas de statistiques et que toutes les données représentées sur les interfaces sont traitées en temps-réel : il n'y a pas de simulation.

9.1 Composition de l'ontologie pour les expérimentations

Les quatre premières expérimentations utilisent la même ontologie. La connaissance de la composition de celle-ci est nécessaire pour la compréhension des résultats. Les éléments importants de l'ontologie sont :

- La classification.
- Les connaissances.
- Les expériences.
- Les accointances.

9.1.1 La classification

La figure D.1 de l'annexe D présente la classification employée dans l'ontologie. Cette classification peut être amenée à évoluer en fonction des nouvelles connaissances intégrées dans le système. Il existe bien sûr plusieurs moyens de

classification valides, il suffit que la classification soit compatible avec le monde réel et ses contraintes.

9.1.2 Les connaissances

Les figures D.2, D.3, D.4 et D.5 de l'annexe D montrent toutes les connaissances présentes dans l'ontologie. Les connaissances sont amenées à changer en fonction de l'évolution d'une entité dans l'environnement grâce à un apprentissage en parallèle des actions.

9.1.3 Les expériences mémorisées

La figure D.6 de l'annexe D présente les différentes expériences contenues dans l'ontologie. Chaque expérience est associée à plusieurs connaissances et correspond à un vécu précis.

9.1.4 Les accointances

Les figures D.7, D.8 et D.9 de l'annexe D présente la liste des accointances présentes entre les connaissances. Plus cette liste est de taille importante, plus le comportement de l'entité peut être compliqué, ce qui lui permet de gérer un nombre de situations possibles important.

9.2 Les différentes expériences et leur utilité

Dans ce chapitre, nous avons cinq expériences. Chacune d'elle permet de prouver le fonctionnement d'une caractéristique du système auto-adaptatif (SAA). Voici la liste des expériences :

1. Activité émergente dans le SAA : permet de montrer que les connaissances utiles à l'interprétation d'un environnement non stable s'activent bien à la lecture des données capteurs.
2. Processus émotionnel dans le SAA : permet de montrer que les émotions prennent la forme de processus concurrents altérant les processus d'objectifs courants. Les émotions font partie intégrante de la prise de décision mais ne sont pas au centre de la résolution de prise de décision dans une boucle systémique.
3. Fusion de deux systèmes auto-adaptatifs : permet de montrer que deux systèmes auto-adaptatifs peuvent communiquer afin d'effectuer une prise de décision de groupe en environnement non stable.
4. Modification des paramètres en temps-réel : permet de montrer que les paramètres d'un système auto-adaptatif peuvent être modifiés en temps-réel.

5. Génération HTML par le SAA : permet de montrer la généricité du système auto-adaptatif en modifiant son rôle principal grâce à une modification de l'ontologie.

Ces différentes expérimentations permettent de monter que le modèle que nous proposons est fait pour fonctionner selon une boucle systémique évoluant dans un environnement non stable. Nous appelons cela une **architecture systémique**.

9.3 Première expérience : activité émergente

9.3.1 Objectifs et hypothèse

Le rôle de cette première expérience est de mettre en évidence une activité en fonction d'une scène particulière, c'est-à-dire que certains agents doivent s'activer pour nous donner la possibilité d'interpréter la scène générée par le système.

Hypothèse : le robot est actif dans un environnement complexe et inconnu pour exécuter une suite d'actions correspondant aux objectifs passés en paramètre. Un objectif est représenté par un graphe de mots possédant des liens topologiques (section 6.6). Ces liens viennent d'une expérience passée dont le résultat a été évalué.

9.3.2 Méthodes d'application

Nous utilisons la reconnaissance d'objets permise par l'Aibo. Celle-ci n'est pas optimale dans un environnement peu éclairé mais cela est suffisant pour travailler au niveau connaissances. Nous avons, pour cette expérience, plusieurs entités : Alain, Mickael, Ball, Bone, Battery et Pink. Nous avons donc deux personnes, deux objets et une couleur, le tout avec un taux de reconnaissance de 90 pour cent. Les quatre phases de l'interprétation sont respectées :

1. Les données transitent dans le système multi-agent. A cette phase, il est impossible pour un être humain d'interpréter en temps-réel toutes les informations et le comportement du système.
2. Les informations sont interprétées grâce à la morphologie dans le but de créer une forme émergente.
3. Une forme particulière est choisie afin de diriger le système en respectant les objectifs passés en paramètres.
4. Un plan d'action est créé grâce à la forme émergente. Durant ce traitement, nous nommons "Point Focal" une suite de connaissances émergentes évoluant en temps-réel en fonction des entrés capteurs.

Voici la configuration utilisée pour réaliser l'expérience :

– Une ontologie de onze méga-octets contenant des verbes, des objets, des émotions simples, des émotions mixtes, des couleurs, des capacités physiques etc...

– Plus de mille agents.
– Plus de sept milles threads.
– Tous les capteurs de l'Aibo.
– Un objectif simple : avoir du plaisir, qui est lié à Ball, Play, Alain, Cushion, Pink, Cover et Bone, ou avoir de la non-satisfaction, qui est lié à Work, Generator, Alain et Fatigue.
– Un PowerBook G4 1,33 Ghz et 1,2 giga-octets de mémoire vive.

9.3.3 Monitoring et analyse

La première interface à analyser est celle regroupant tous les messages transitant dans le système. Nous voyons sur la figure 9.2 que dés le début de l'exécution, la boucle systémique se forme et le robot commence à exécuter des mouvements envoyés par le système. Dans un premier temps, ceux-ci ne respectent pas forcément les objectifs passés en paramètre.

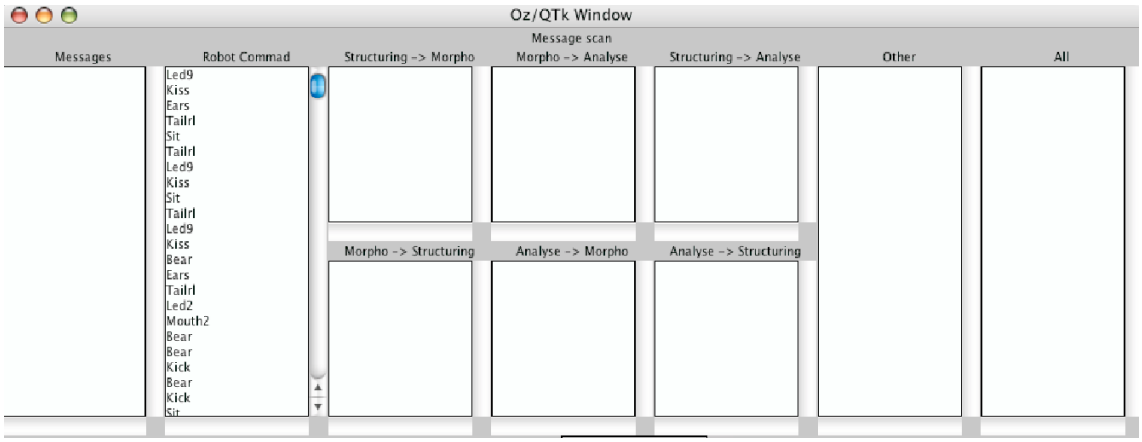

FIG. 9.2 – Dès l'exécution du système, des commandes sont envoyées au robot. Celui-ci les exécute directement.

Les objectifs sont directement pris en compte par le système. Si le comportement du robot ne se dirige pas vers la bonne réalisation des objectifs, alors le système s'auto-adapte. L'auto-adaptation se fait en modifiant l'activation de certains agents, c'est-à-dire en privilégiant certains par rapport à d'autre, ce qui modifie la forme générale de l'organisation d'agents. La figure 9.3 montre les premiers messages morphologiques vers les agents structurants. La figure 9.4, quant à elle, montre que l'envoi de ces messages est continuel, cela fait partie intégrante de la boucle systémique.

Les autres interfaces, les plans ontologiques, les points focaux ainsi que le suivi des objectifs sont à analyser de manière simultanée. Lors de l'initialisation du sys-

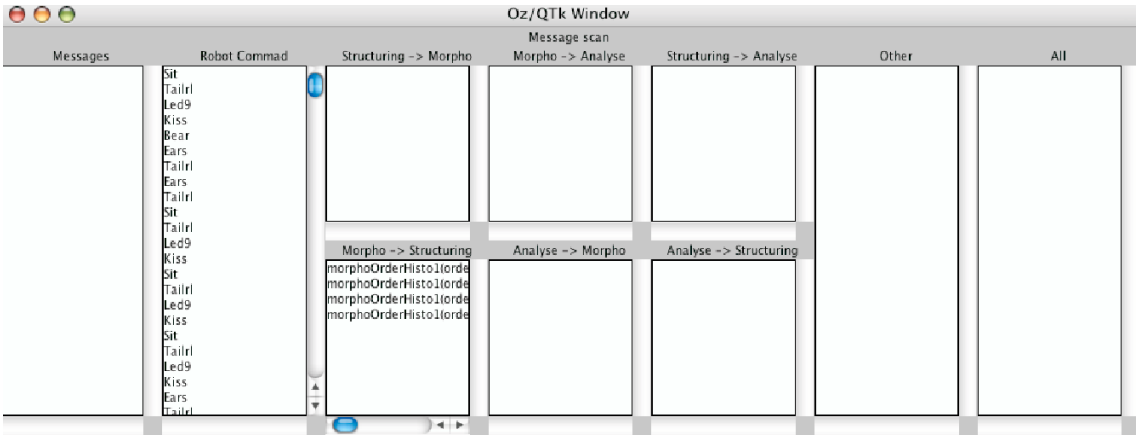

FIG. 9.3 – Le système s'adapte en fonction des objectifs. Des messages de morphologies sont envoyés aux agents structurants.

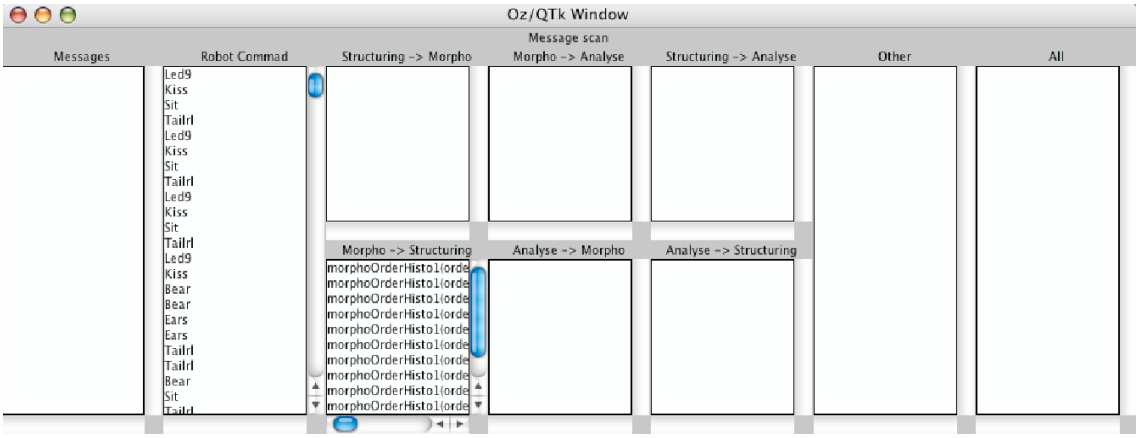

FIG. 9.4 – La boucle systémique ne s'arrête jamais et le système s'auto-adapte en fonction de son environnement. Tant que le système fonctionnera, les agents structurants recevrons des messages de morphologies.

tème, c'est à dire la création des agents, les plans ontologiques sont inertes comme le montre la figure 9.5. C'est après l'initialisation que le système commence à interpréter la scène ou il se trouve, ce qui donne une déformation de plans comme sur la figure 9.6. La visualisation de cette déformation permet de voir quelles connaissances présentes dans l'ontologie sont utilisées pour l'interprétation de la scène. Le nombre de plans ontologiques dépend de l'ontologie, ils peuvent atteindre un nombre important. Parallèlement à cela, un point focal, "la représentation courante", est en construction[cf. figure 9.8].

Sur ce point focal nous constatons que le système réussit à se focaliser sur certains agents en étant capable de les lier entre eux. Chaque point focal veut dire quelque chose. Celui-ci veut dire que le robot veut jouer (verbe “Play”) avant de, peut être, utiliser un livre (“Front of” et “Book”). Ce point focal correspond à un comportement physique direct. Avec le temps, le système évolue et les plans ontologiques changent de formes comme le montre la figure 9.7. Le verbe “Play” sort du lot et de nouveaux points focaux orientés sur ce verbe sont ainsi générés [cf. figures 9.9 et 9.10].

Les plans ontologiques et les points focaux permettent de constater l’évolution du système et l’émergence des connaissances, mais le système possède aussi certains objectifs à atteindre, définis par l’utilisateur. Ce dernier peut suivre la bonne réalisation de ceux-ci grâce à une courbe représentant la différence entre l’objectif et les actions courantes. Comme le montre la figure 9.11, les premières actions sont déjà mesurées. Il faut que la courbe tende vers zéro. La régularité de la courbe va dépendre des paramètres passés au système concernant les agents. Quand l’utilisateur donne une certaine liberté aux agents, la courbe n’est pas toujours régulière mais s’orientera toujours vers le zéro [cf. figures 9.12 et 9.13]. Ce léger écart par rapport à de l’objectif permet une plus grande improvisation en cas de problèmes ponctuels.

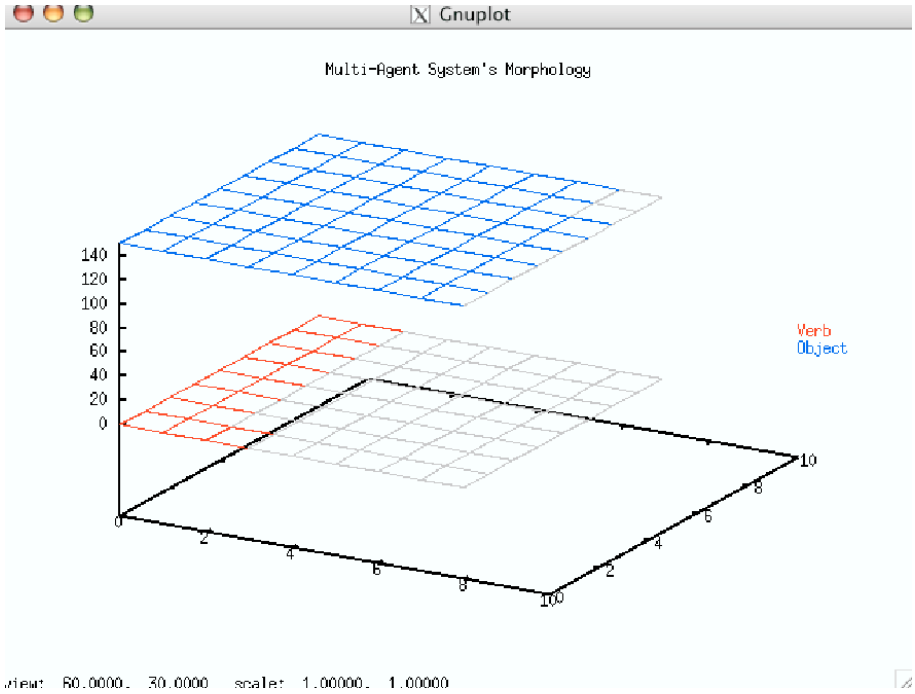

FIG. 9.5 – A l’initialisation du système, les plans ontologiques restent statiques.

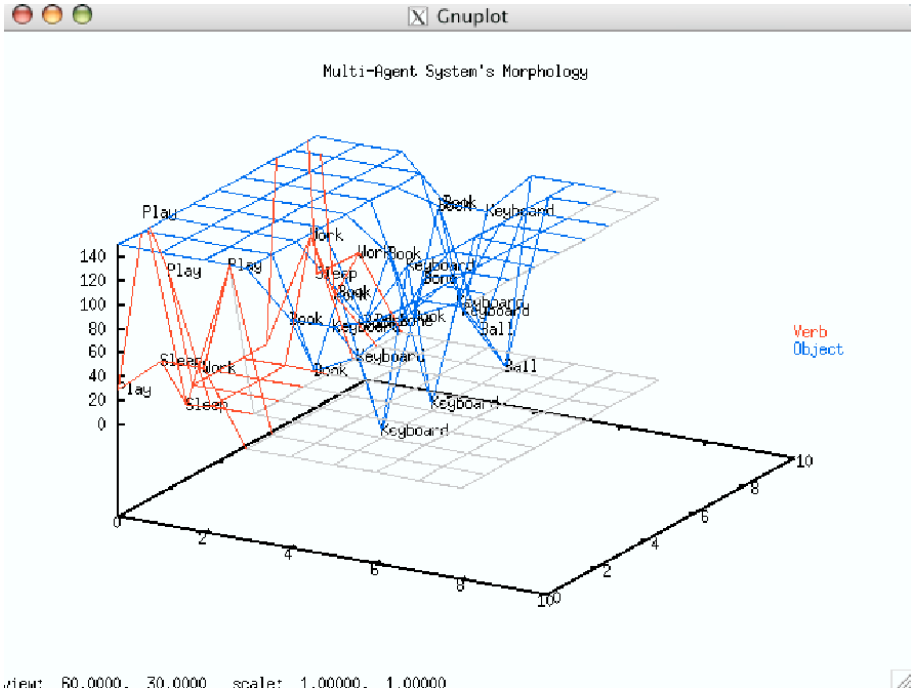

FIG. 9.6 – Durant l'évolution du robot dans l'environnement, les plans ontologiques se déforment laissant apparaître une activation particulière de connaissances.

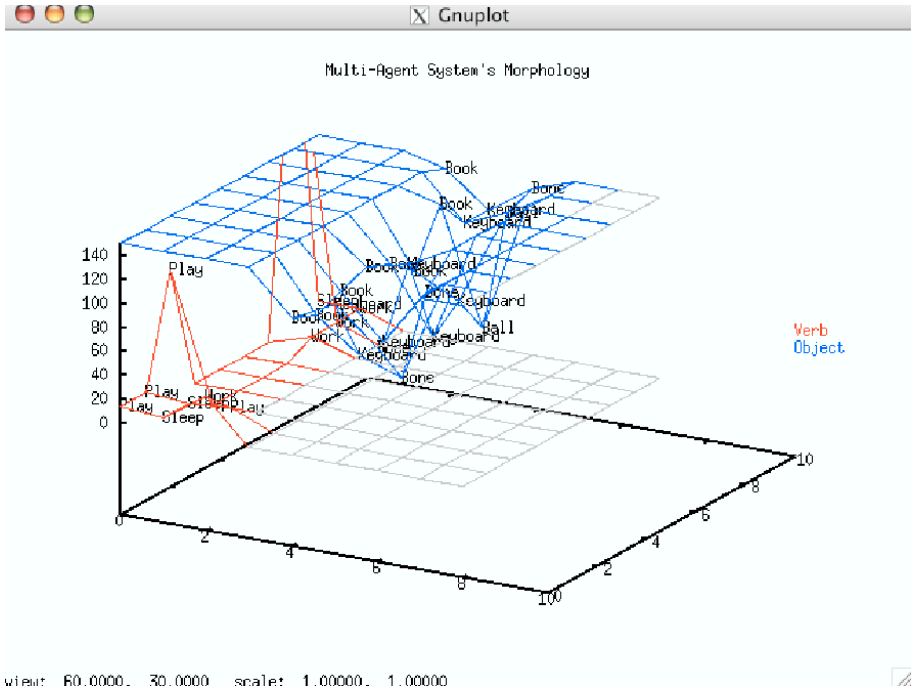

FIG. 9.7 – Une autre déformation des plan ontologiques. Ces plans dépendent de la classification employée dans l'ontologie.

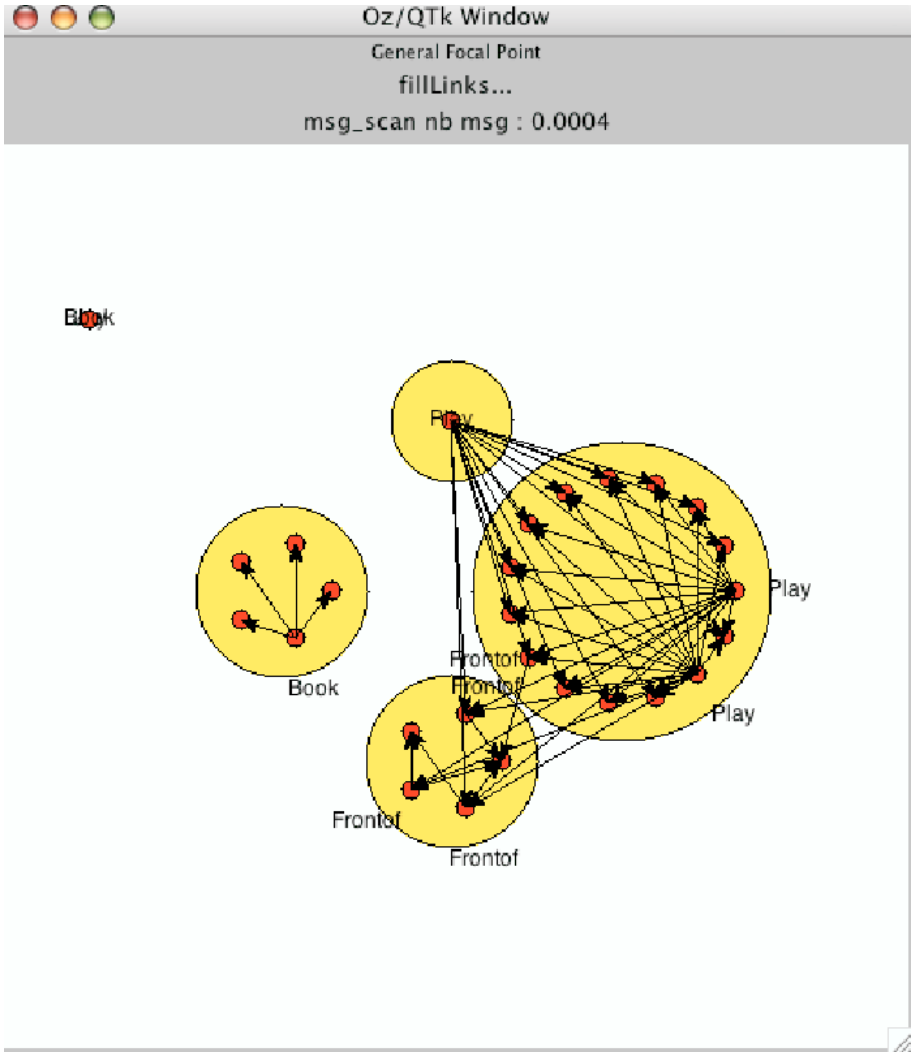

FIG. 9.8 – Une représentation courante.

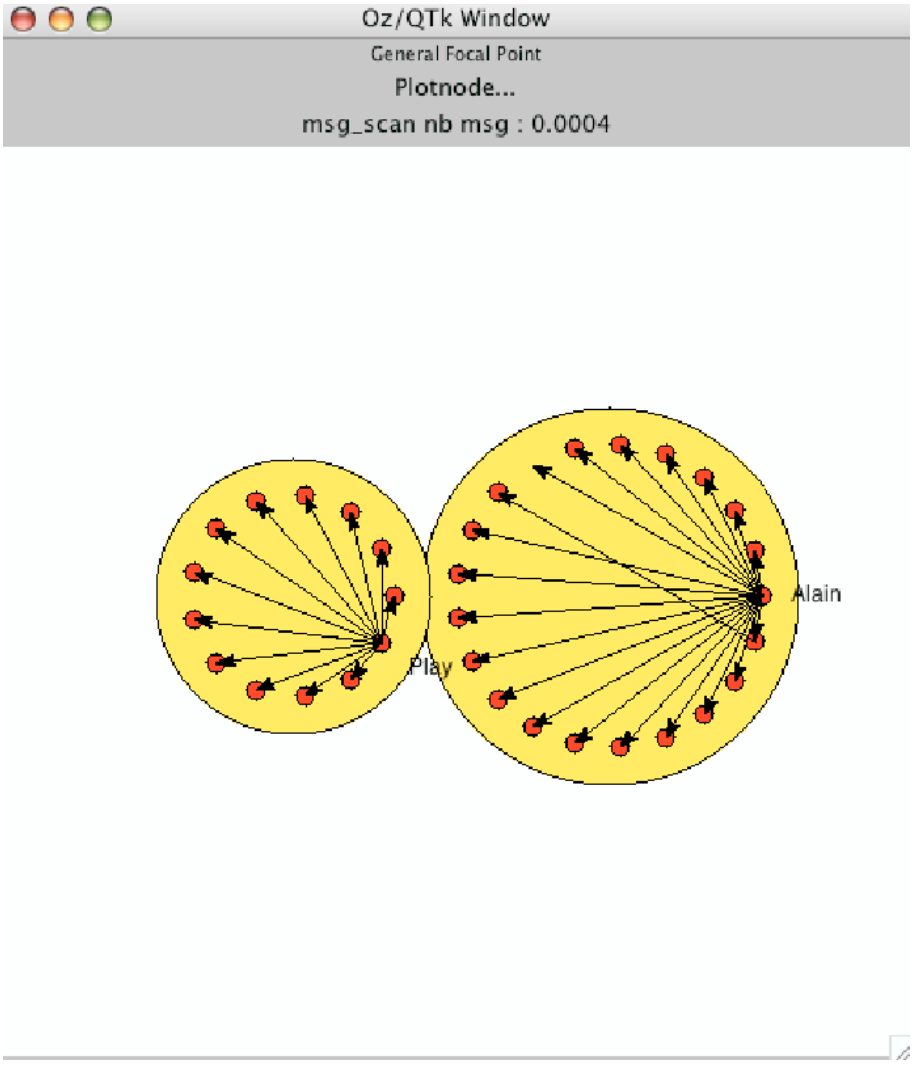

FIG. 9.9 – Une deuxième représentation courante.

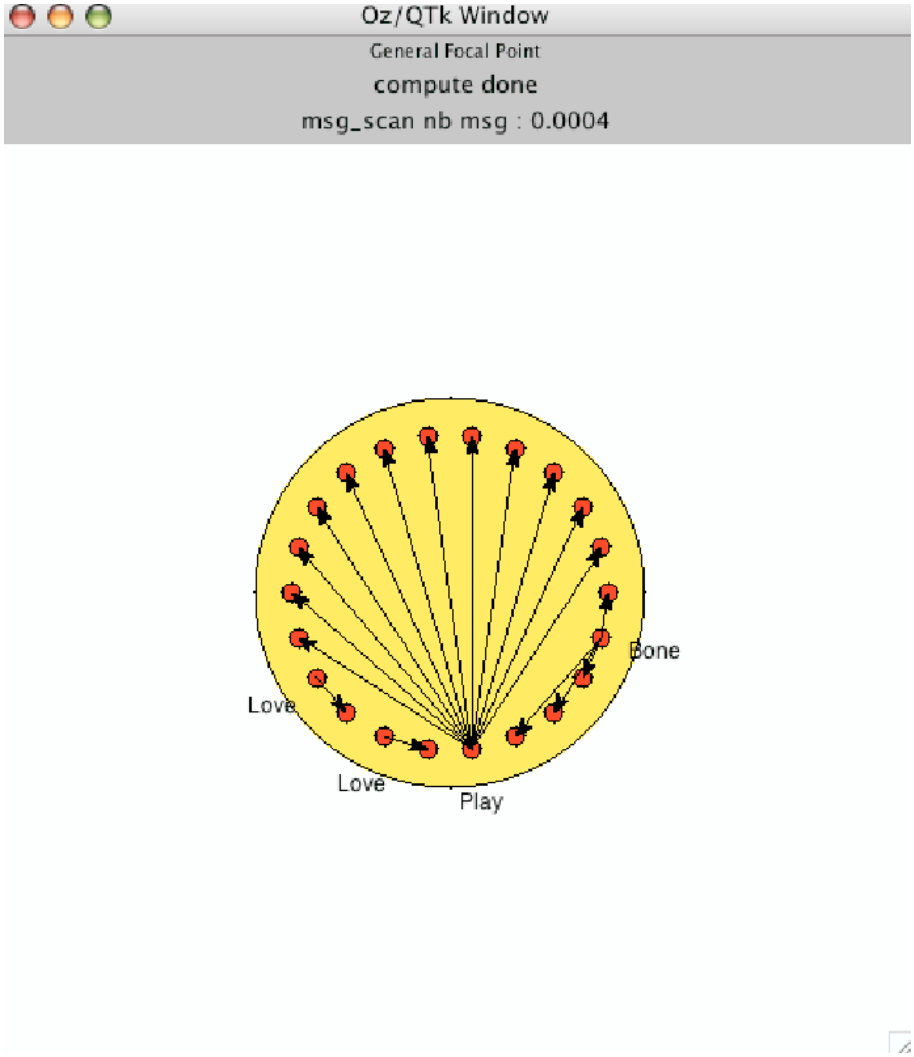

FIG. 9.10 – Une troisième représentation courante.

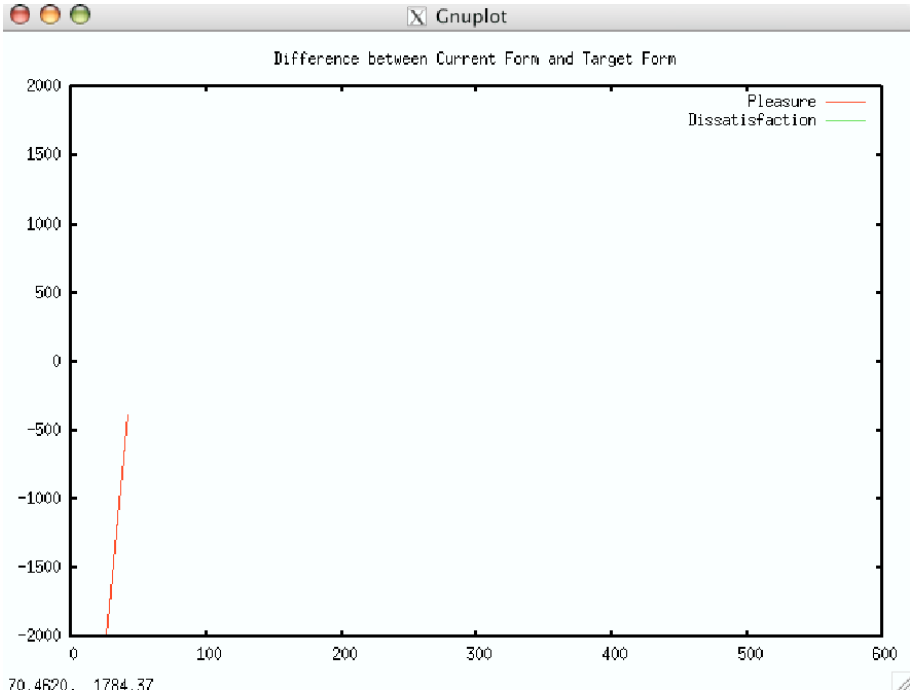

FIG. 9.11 – Une première évaluation de l'atteinte des objectifs.

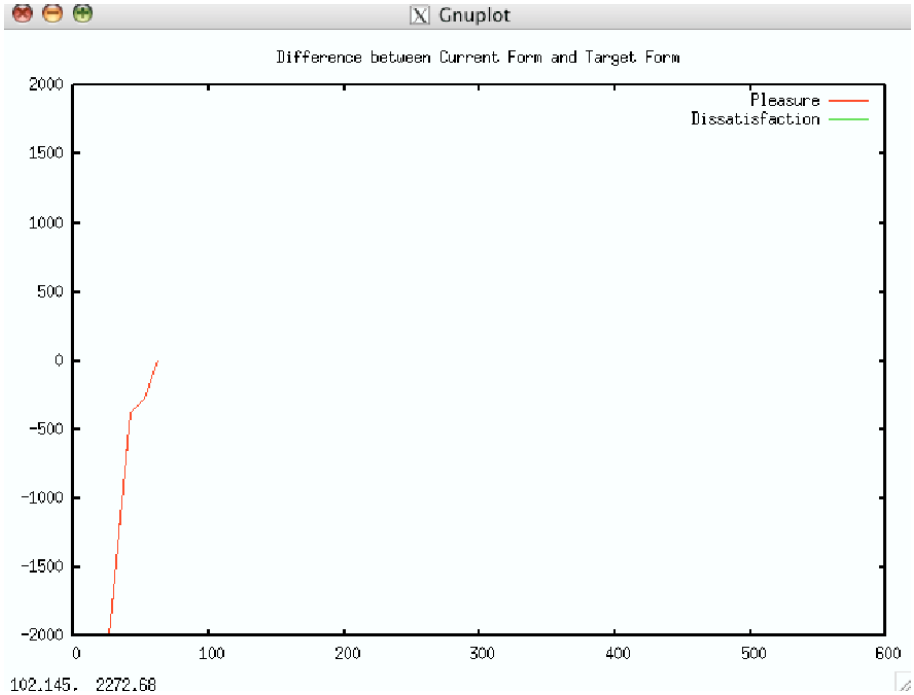

FIG. 9.12 – Une seconde évaluation de l'atteinte des objectifs.

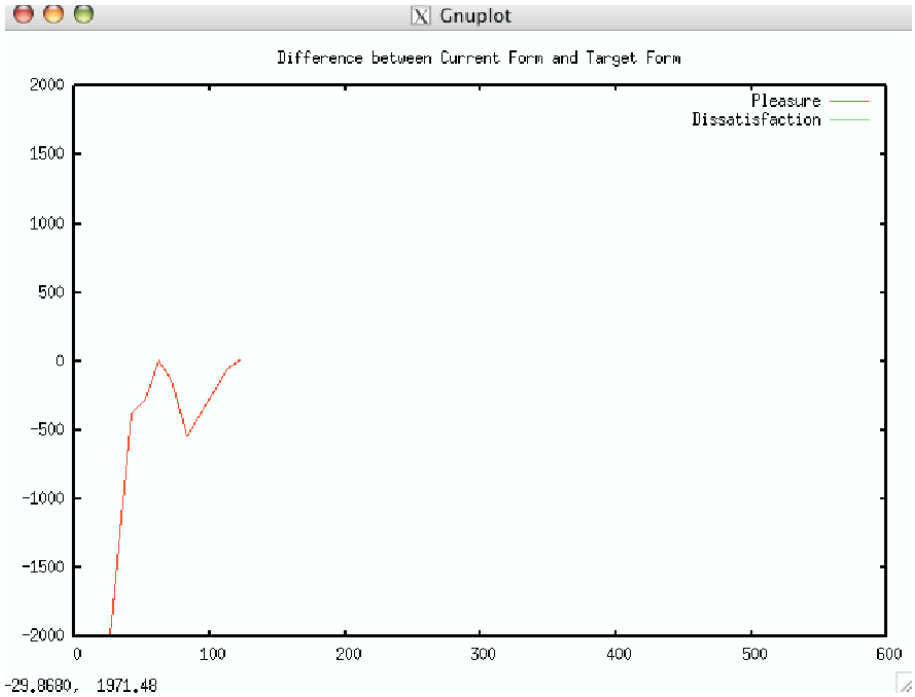

FIG. 9.13 – Une troisième évaluation de l'atteinte des objectifs.

9.4 Deuxième expérience : processus émotionnel

9.4.1 Objectif et hypothèse

L'objectif de cette expérience est de représenter une scène qui est altérée par un processus émotionnel.

Hypothèse : le robot est actif dans un environnement non connue pour exécuter une suite d'actions en fonction des objectifs passés en paramètres. Un objectif est représenté par un graphe de mots possédant des liens topologiques (section 6.6). Ces liens viennent d'une expérience passée dont le résultat a été évalué.

9.4.2 Approche

Dans notre cas applicatif, une entité est décrite comme une machine possédant un corps et un système auto-adaptatif qui n'est pas embarqué dans le robot pour l'instant. Les émotions sont présentes dans l'ontologie et sont liées à différents stimulus.

Rappelons le niveau de traitement des émotions :

- Les émotions sont représentées par des mouvements périodiques d'agrégations d'agents structurants.
- Ces agrégations expriment l'émotion concernant la représentation de quelque chose.
- Le type de la déformation représente le type de l'émotion.
- La déformation est une modification de l'émergence causée par la reconnaissance de certains objets (point focal).
- L'intensité de l'émotion est représentée par la fréquence des agrégations.

9.4.3 Objectifs du scénario

Pour cette expérience, l'objectif est pour l'Aibo de se diriger vers une balle, ce qui lui permet d'éprouver du plaisir. Le fait de se déplacer vers la balle est donc lié au plaisir. Si le robot éprouve du plaisir, cela voudra dire qu'il se dirigera vers son objectif. Un objectif est représenté par un graphe de mots possédant des liens topologiques (section 6.6). Ces liens viennent d'une expérience passée dont le résultat a été évalué.

9.4.4 Caractéristiques émotionnelles

Les caractéristiques émotionnelles du système pour cette expérience sont les suivantes : L'amour, la joie et la peine.

9.4.5 Accointances

Pour le mot Ball : capacités physiques, personnes, des lieux spécifiques, des couleurs.

Pour le mot Bone : capacités physiques, une émotion et une couleur.

Pour le mot Pleasure : des objets (ball, cushion, cover et bone), un verbe (to play), une couleur (pink) et une personne (Alain).

Pour le mot Dissatisfaction : un verbe (to work), un objet (generator), un état (fatigue) et une personne (Alain).

9.4.6 Monitoring et analyse

De nouveaux plans apparaissent dans cette expérimentation. Nous avons maintenant un plan concernant les émotions mixtes et un plan concernant les émotions simples [cf. figure 9.14]. Ces plans respectent la classification émotionnelle abordée dans le chapitre 6.2. Nous voyons que les connaissances Love, Sorrow et Joy sont activées, c'est pour cette raison que nous retrouvons Love dans le point focal de la figure 9.10.

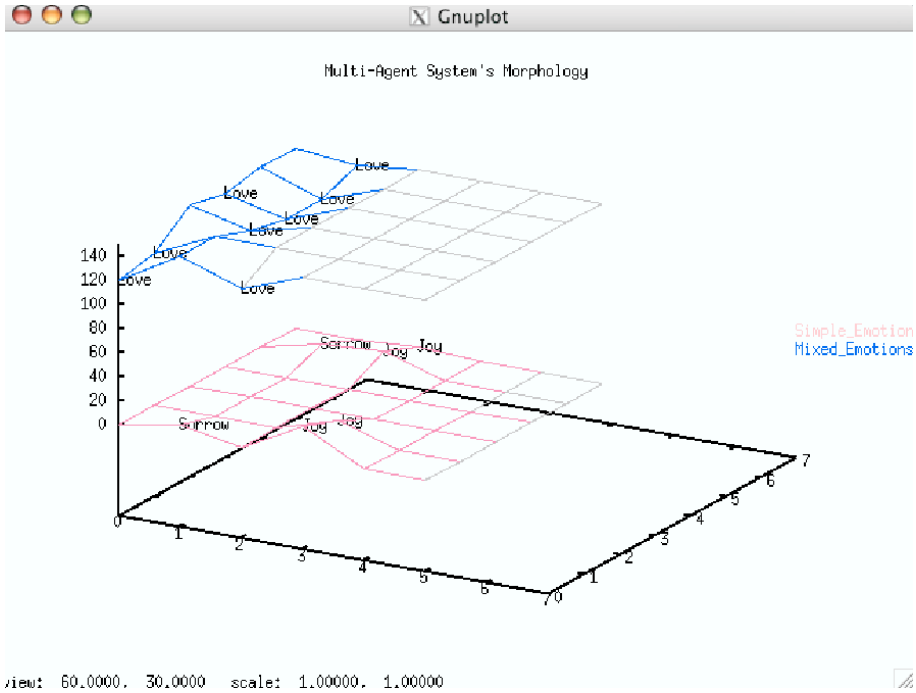

FIG. 9.14 – Les plans ontologiques concernant les émotions.

Chaque représentation émotionnelle du système, le plaisir et la non-satisfaction, s'altèrent entre elles. En fonction de ce que ressent le système, celui-ci va retarder ou non son avancé vers son objectif. La figure 9.15 montre qu'une représentation émotionnelle peut être plus importante qu'une autre, ici la plaisir par rapport a la non-satisfaction. Les deux représentations émotionnelles sont contraires, elles ne peuvent donc pas évoluer de la même manière. Love et Joy soutiennent le plaisir alors que Sorrow soutient la non-satisfaction. L'objectif de l'expérimentation qui est de se diriger vers une balle (ce qui procure du plaisir) est altéré, mais pas supprimé, par les émotions.

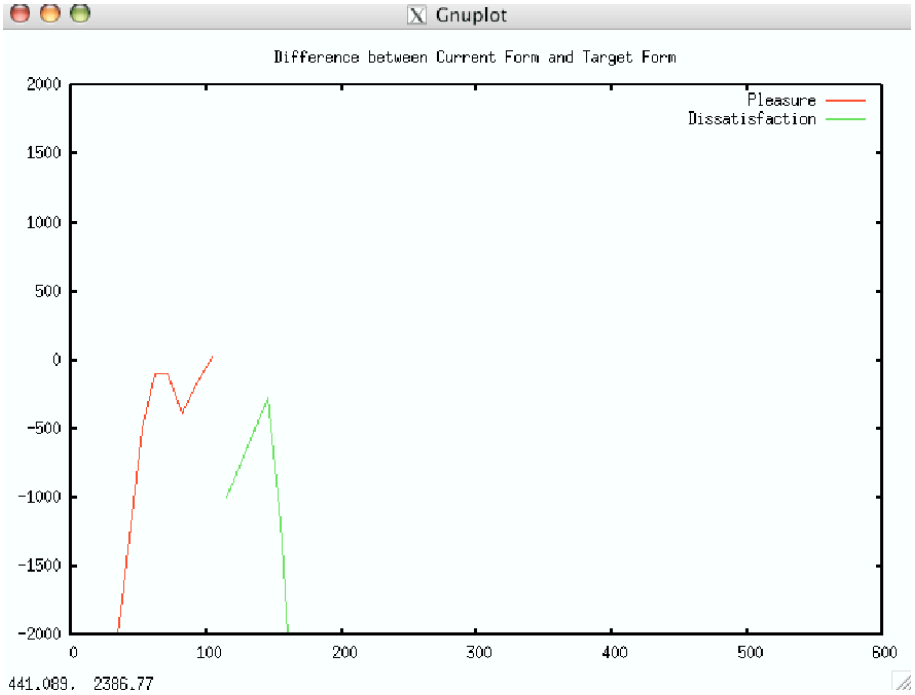

FIG. 9.15 – Divergence des représentations émotionnelles.

Le robot modifie son comportement lorsque le plaisir est altéré. Cette modification est radicale quand le plaisir s'écroule totalement. Le robot pense à jouer mais se laisse submerger par ses émotions. L'émotion fait partie intégrante de sa prise de décision.

9.5 Troisième expérience : fusion de deux systèmes auto-adaptatifs

9.5.1 Objectif et hypothèse

L'objectif de cette expérience est de réussir à fusionner deux systèmes afin d'atteindre une décision de groupe satisfaisante pour un contexte donné.

Hypothèse : deux robots sont actifs dans un environnement non connu par eux pour exécuter une suite d'actions correspondant à des objectifs passés en paramètres. On les nomme robot un et robot deux.

9.5.2 Approche

Cette expérience est cruciale, elle permet de fusionner deux systèmes auto-adaptatifs en temps-réel. Deux robots Aibos avec connexion Wifi sont en fonctionnement pour ce test. Nous commençons par effectuer, en ligne, une association entre les organisations d'agents au niveau morphologique, ce qui modifie les deux organisations en suivant trois cas précis :

- Un robot est maître et l'autre est esclave.
- Les deux robots génèrent une nouvelle expression morphologique intégrant un nouveau point focal.
- L'agrégation est impossible : les deux organisations n'ont rien en commun et chaque robot utilise son organisation propre.

Une agrégation peut être appliquée à *n* systèmes. La figure 9.16 montre qu'une nouvelle analyse est effectuée à partir de l'injection de connaissances dans les agents structurants et l'injection des statistiques morphologiques dans le système.

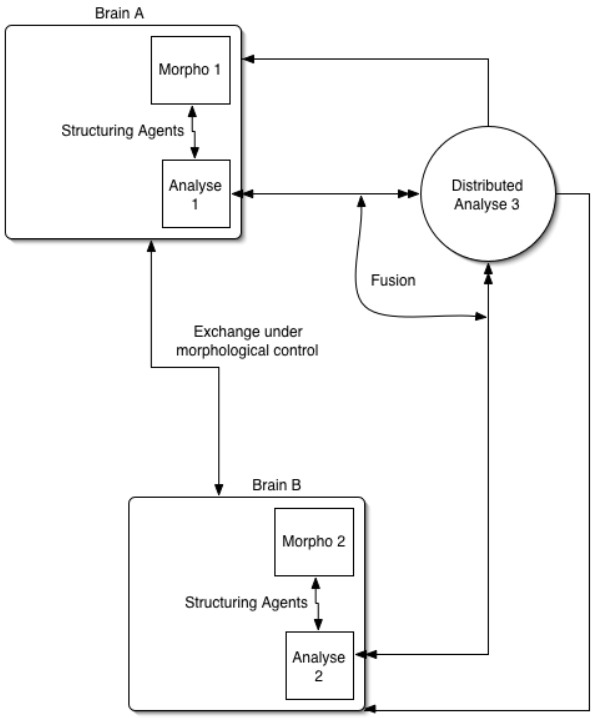

FIG. 9.16 – Les deux systèmes auto-adaptatifs communiquent par pattern de mémoire. La perception de l'un va compléter la perception de l'autre pour former une nouvelle analyse et prendre une décision de groupe.

9.5.3 Monitoring et analyse

Dans cette expérimentation, chaque robot va posséder ses propres plans ontologiques, ses propres point focaux, ainsi que ses propres courbes d'objectifs. Les deux robots évoluent dans un même environnement sauf que le robot numéro un à une vue précise de ce qui se passe dans l'environnement, alors que le robot numéro deux ne voit rien du tout et ne peux donc pas effectuer d'interprétation pour prendre des décisions.

Lors du lancement de l'expérimentation, les plans ontologiques des deux robots ne sont pas actifs (figure 9.17). Mais la figure 9.18 montre que le robot numéro un a commencé une activation de connaissances selon son environnement, alors que le robot numéro deux ne se rend compte de rien. Celui-ci aura une activation de connaissances propre quelques secondes après le premier robot[1]. Par cette activation de connaissances, le deuxième robot va générer un point focal propre tout en ayant les mêmes objectifs que son homologue (figure 9.19). De ce fait, comme le montre la figure 9.20, la direction des objectifs par les robots est suivie de manière différente. Le comportement est satisfaisant pour les deux robots (courbes tendant vers zéro) mais est tout de même sensiblement différents. Les objectifs ont été atteints de manières différentes, ceci permet la synergie de groupe.

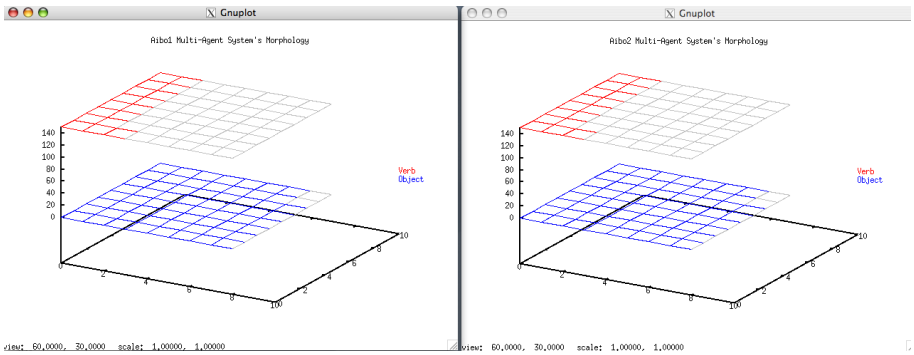

FIG. 9.17 – Au lancement des deux systèmes, l'activation des agents est inexistante des deux côtés.

[1] Il est difficile de donner une durée précise car tout dépend du robot, de l'état de la mémoire vive, de l'architecture de la mémoire virtuelle présente dans le système d'exploitation, sans compter le comportement de la machine virtuelle de Oz/Mozart. Il n'est pas possible, ici, d'effectuer des mesures avec les outils existants, il faudrait le faire avec le système auto-adaptatif même pour qu'il y ait une signification.

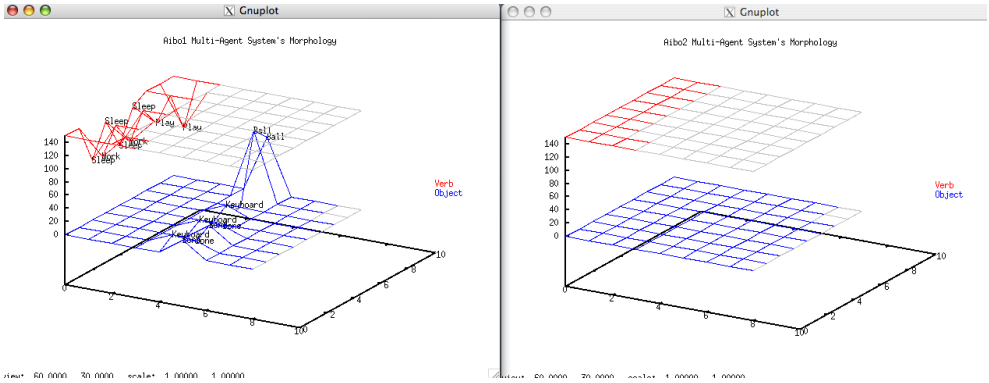

FIG. 9.18 – Le robot capable de se rendre compte de la scène génère une activation de ses connaissances. Le deuxième robot ne connaît toujours pas l'environnement dans lequel il évolue.

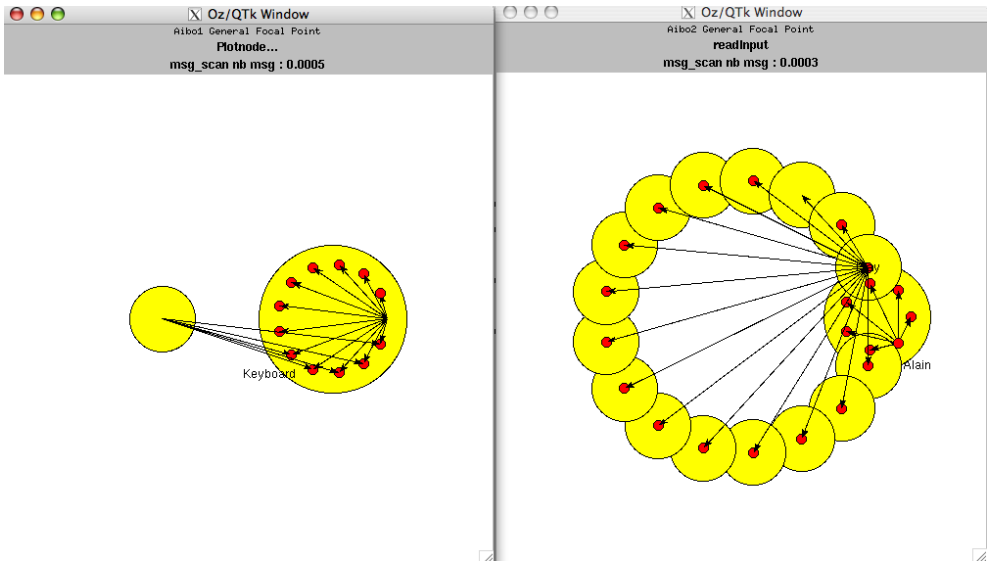

FIG. 9.19 – Bien que le deuxième robot utilise l'interprétation de la scène du premier robot, il génère un point focal différent de son homologue.

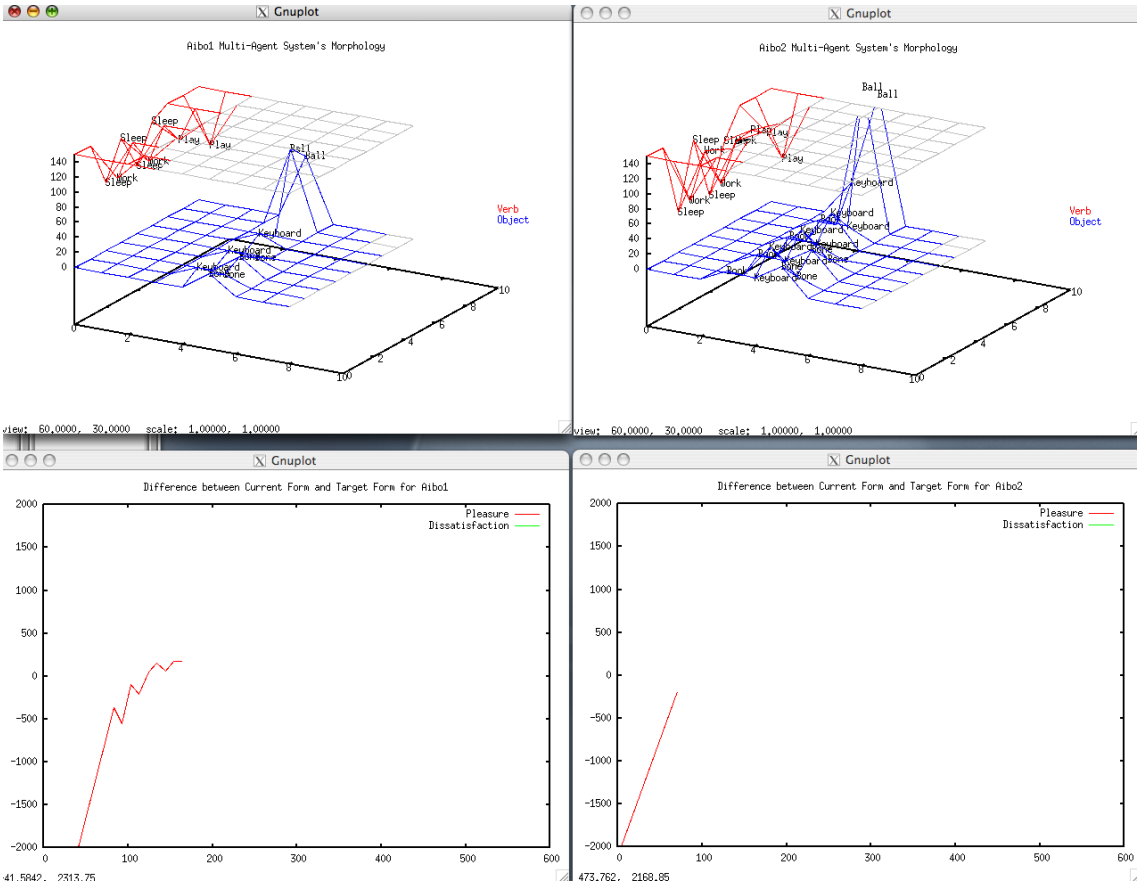

FIG. 9.20 – Les objectifs sont atteints de manière différentes.

9.6 Quatrième expérience : modification des paramètres en temps-réel

9.6.1 Objectif et hypothèse

L'objectif de cette expérience est de montrer que les paramètres du système auto-adaptatif peuvent être modifiés en temps-réel. En modifiant ces paramètres, le comportement du système change sensiblement. Ceci est visible sur le monitoring.

Hypothèse : le robot est actif dans un environnement non connu pour exécuter une suite d'actions en fonction des objectifs passés en paramètres. Un objectif est représenté par un graphe de mots possédant des liens topologiques (section 6.6). Ces liens viennent d'une expérience passée dont le résultat a été évalué.

9.6.2 Approche

Le système est lancé avec une configuration de base au niveau des connaissances. L'objectif est de passer du temps avec une personne (ici, Alain), ce qui procure du plaisir, comme vu dans l'expérience de la section 9.4. La configuration de base est modifiée en temps-réel pour réorienter le système. La connaissance "Work" a été augmentée au niveau de l'agentification, c'est-à-dire que le nombre d'agents s'occupant de cette connaissance a été modifié en temps-réel grâce à l'interface de l'ontologie.

9.6.3 Monitoring et analyse

Lors du lancement du système avec la configuration de base, les connaissances activées correspondent bien à l'objectif du système (figure 9.21). Nous avons les connaissances "Keyboard", "Sleep", "Book". Celles-ci sont liées fortement à la connaissance "Alain" dans l'ontologie de l'annexe D. Le système se dirige naturellement vers son objectif, ceci est visible sur le point focal présenté sur la figure 9.22. Nous voyons sur cette figure les liaisons entre les connaissances citées précédemment.

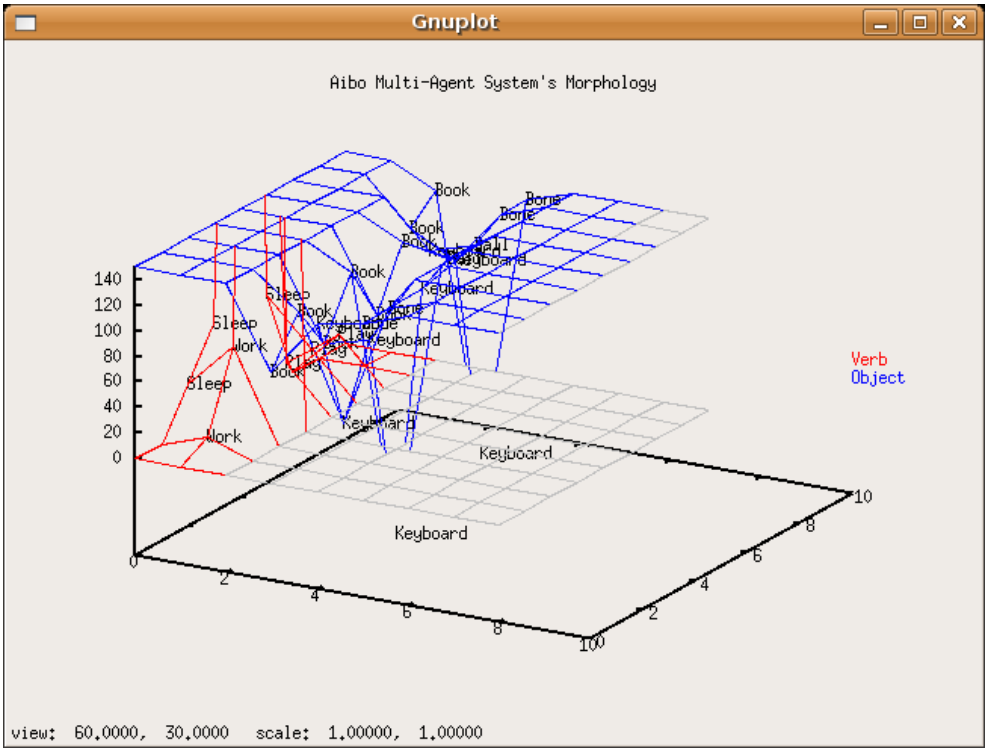

FIG. 9.21 – Plans ontologiques des objets et des verbes sans modification.

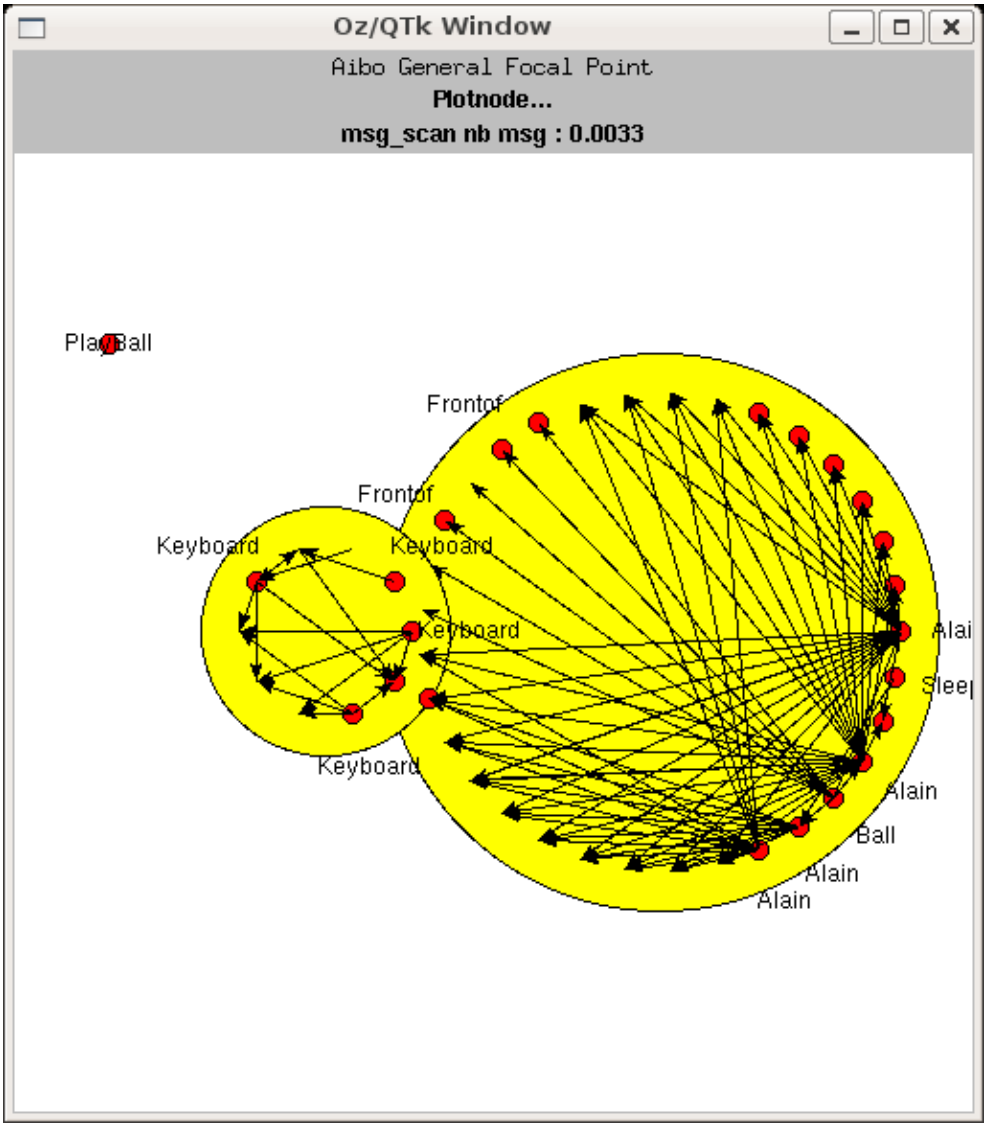

FIG. 9.22 – Point focal sans modification.

A l'instant de la modification sur la connaissance "Work", l'organisation du système est tout autre. La connaissance "Work" émerge plus facilement que les autres (figure 9.23). Ce qui perturbe grandement l'objectif principal du système qui est de passer du temps avec Alain, représenté dans l'ontologie par la connaissance "Alain". La figure 9.24 montre que le point focal du système s'est totalement transformé. Le système ne se dirige plus vers son objectif initial car les connaissances émergentes ne le permettent pas.

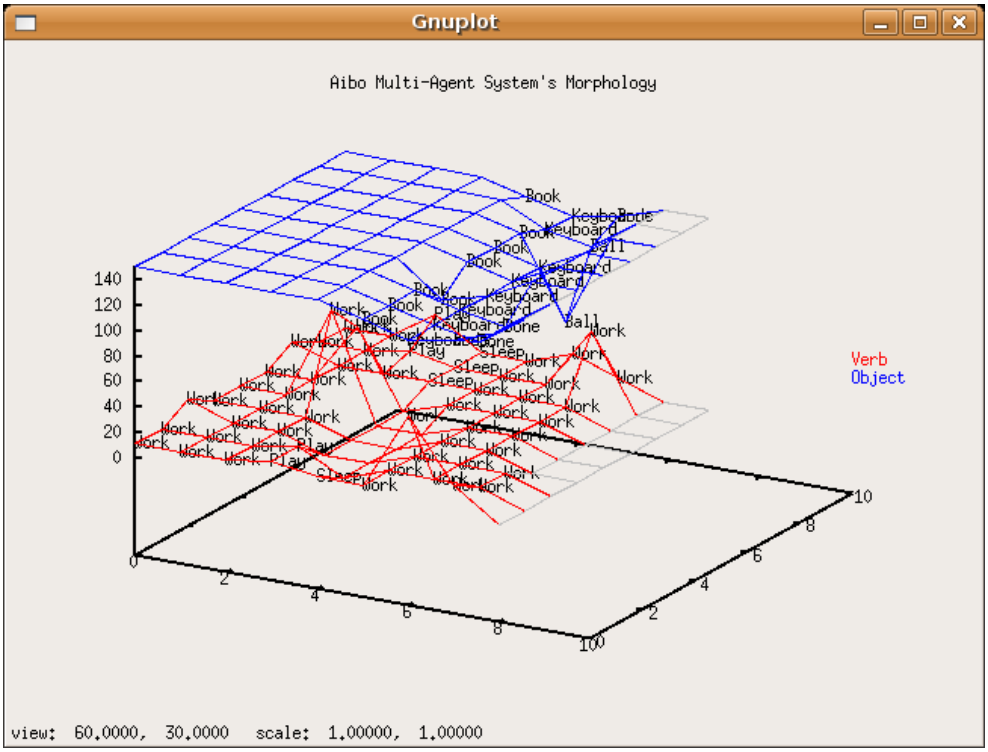

FIG. 9.23 – Plans ontologiques des objets et des verbes avec modification.

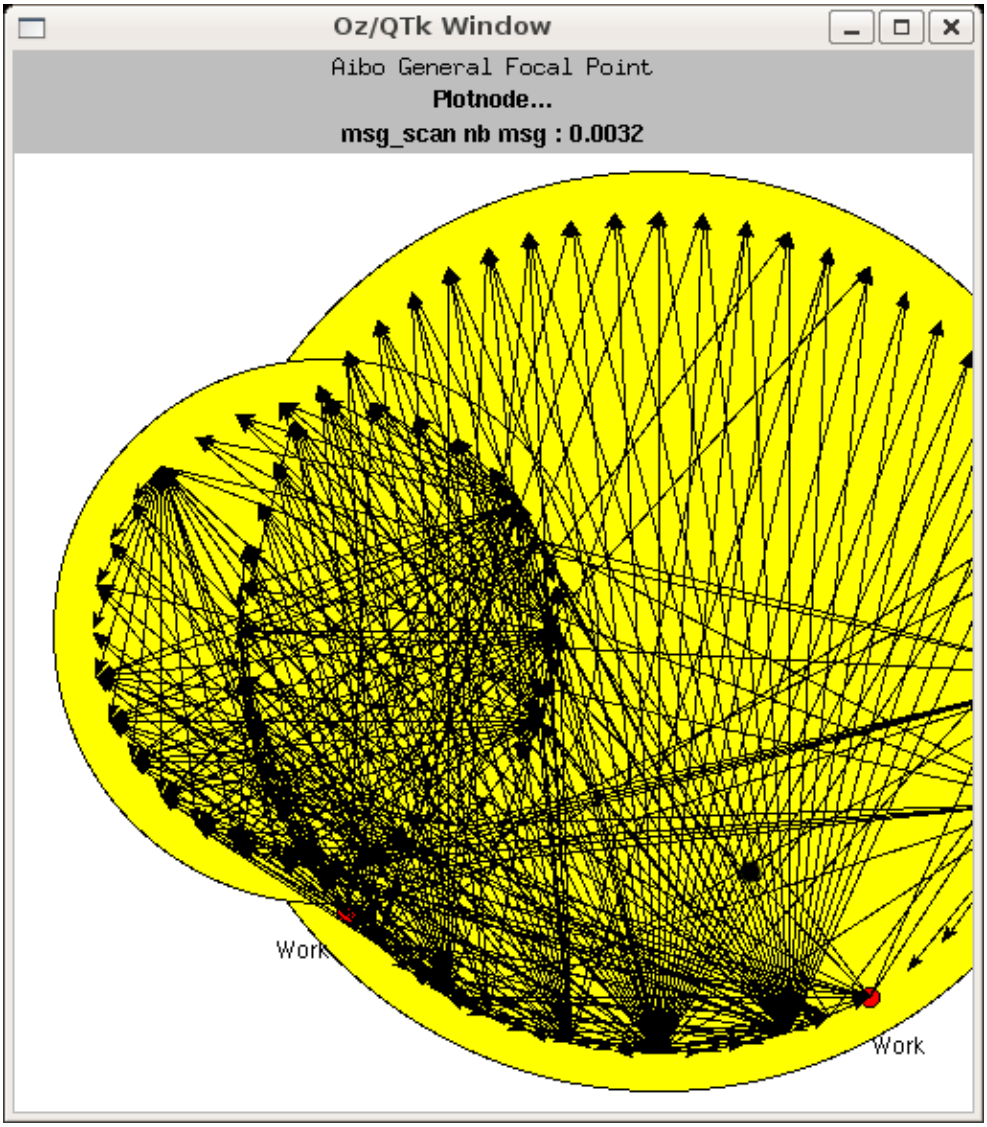

FIG. 9.24 – Point focal avec modification.

9.7 Cinquième expérience : génération de code html

9.7.1 Objectif et hypothèse

L'objectif de cette expérience est de montrer qu'il suffit de changer la connaissance et les expériences du système auto-adaptatif pour que celui-ci puisse être utilisé pour une tout autre activité que la conduite de robots.

Hypothèse : un utilisateur ne connaissant pas le développement HTML veut réaliser un site internet sans se préoccuper du code. Ce dernier va utiliser l'ontologie HTML fournit et passer l'objectif "page_generation" en paramètre (cet objectif est déjà présent dans l'ontologie).

9.7.2 Approche

Le système planifie dynamiquement des processus dans le but d'avoir un plan d'action adapté à chaque situation. De ce fait, il est possible de modifier l'ontologie du système auto-adaptatif pour lui donner un rôle approprié. Les expérimentations précédentes utilisaient une ontologie concernant un robot. Ici, nous changeons totalement d'ontologie pour orienter le système vers une génération de balises HTML dans le but de créer des pages de manière dynamique, sans avoir de développement supplémentaire. Cette expérience ne va pas très loin au niveau de la génération HTML, l'objectif est essentiellement de montrer que le système est générique, et qu'aucune programmation supplémentaire n'est demandée pour changer le rôle du système auto-adaptatif.

Pour cette expérimentation, une nouvelle classification a été faite dans l'ontologie (figure 9.25). Celle-ci est très simpliste, les balises HTML de bases y sont ajoutées (figure 9.26) :

- Le mot "nothing" permet de préciser au système que le fichier HTML est vide.
- L'objectif "page_generation" permet d'orienter l'organisation du système vers la génération de pages en se basant sur l'expérience présente dans l'ontologie (figure 9.27).

Vu que le système envoie des ordres (ici ce sont les balises HTML) sous forme de chaînes de caractères, elles sont enregistrées dans un fichier. Le mot "htmlpage" permet au système de connaître la composition d'une page HTML, il est relié au mot représentant l'objectif "page_generation" (figure 9.28).

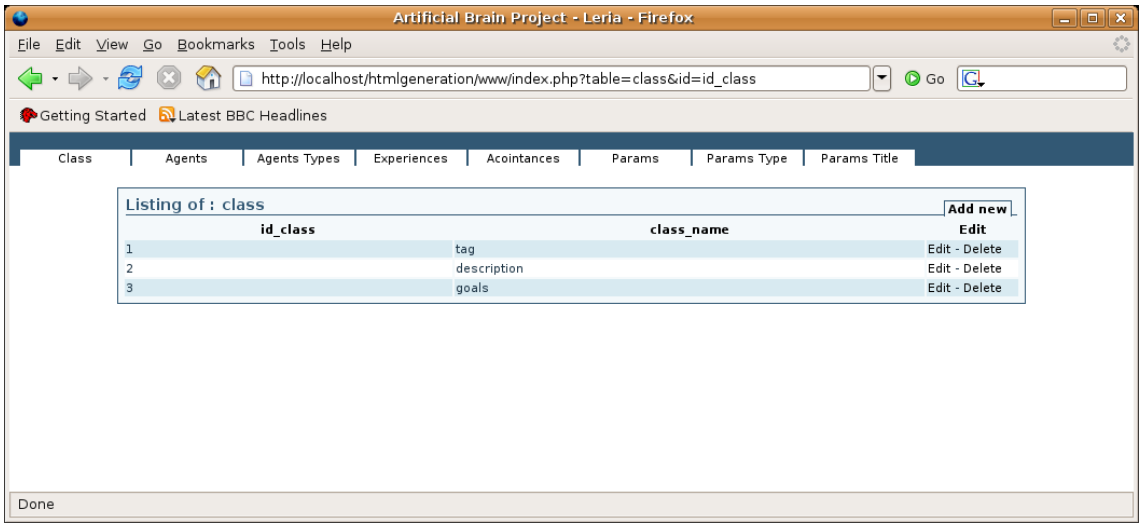

FIG. 9.25 – Eléments de classification pour le HTML.

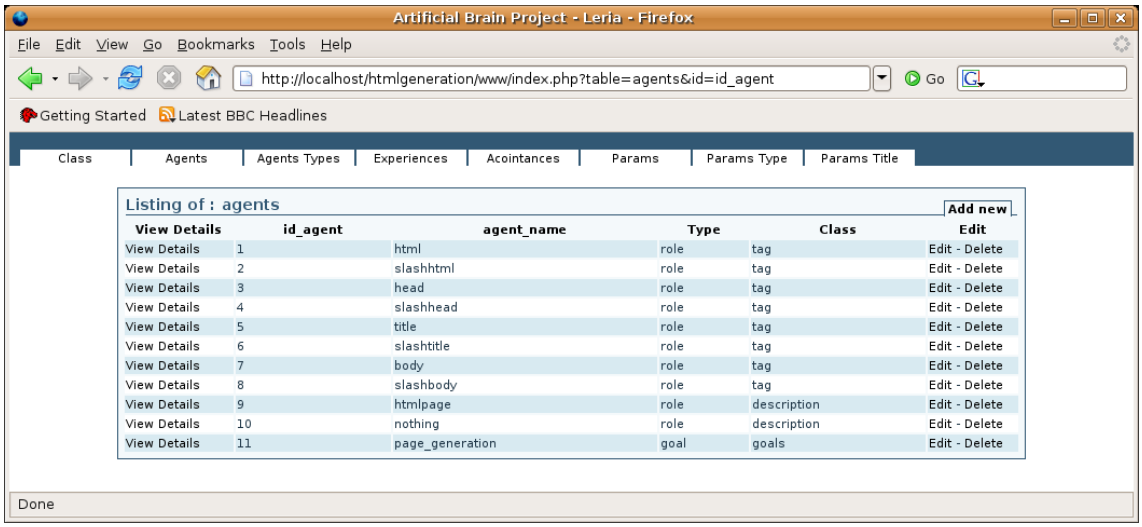

FIG. 9.26 – Liste des agents.

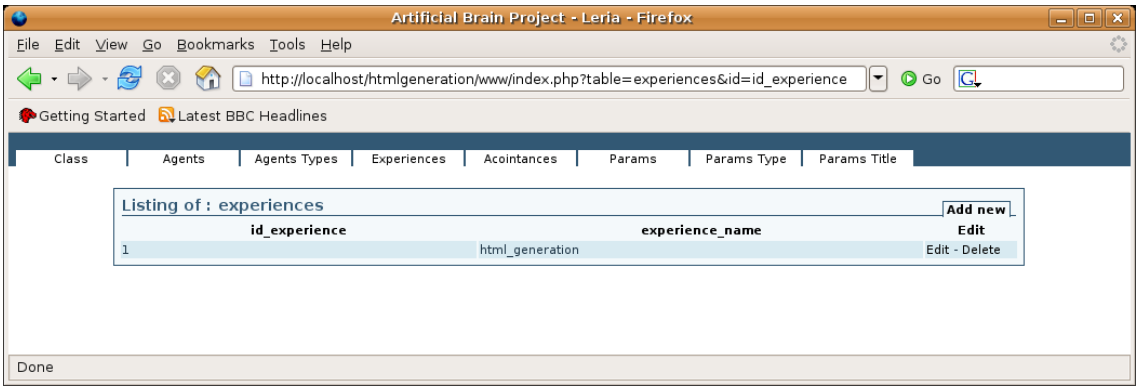

FIG. 9.27 – Expérience liée à la génération HTML.

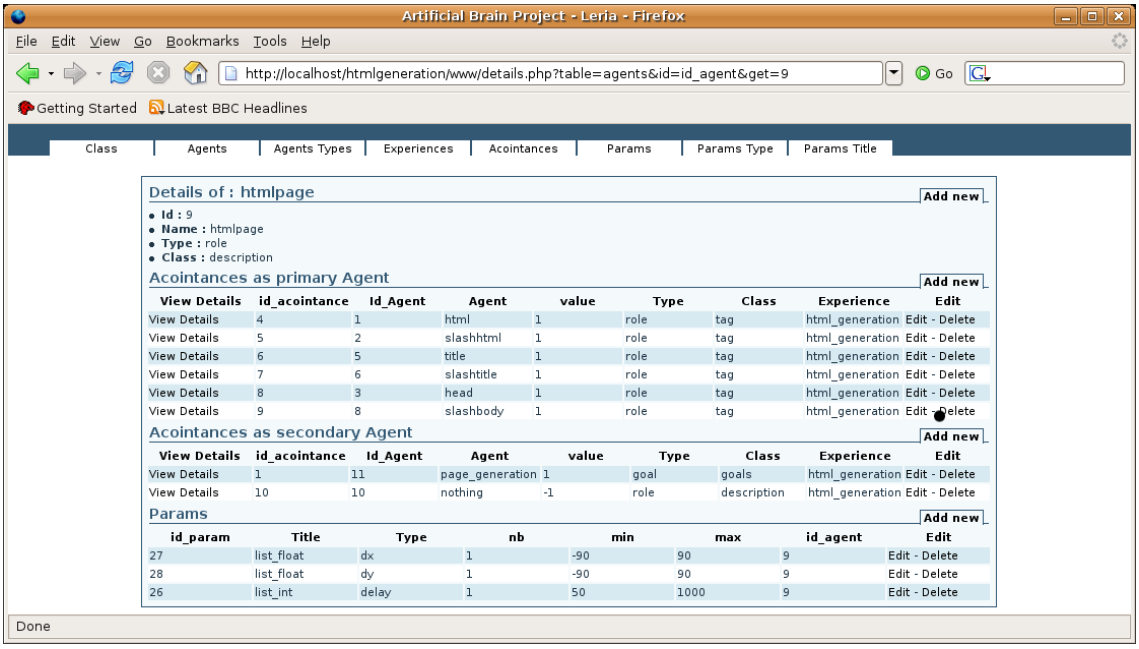

FIG. 9.28 – Description de l'agent concernant une page HTML.

9.7.3 Analyse du résultat

Le résultat est très rapide à obtenir car l'environnement n'est pas compliqué et il n'y a pas beaucoup d'informations à synchroniser. Ici, le système de monitoring n'a pas le temps d'afficher un suivi : la page HTML est générée immédiatement

(elle ne fait que quelques octets, figure 9.29). Les logs du système montrent que le graphe des balises HTML est bien traité (figure 9.30). Les messages “egraphDescr” et “egraph” indiquent que l’analyse morphologique est en cours de réalisation et que l’objectif de génération de page modifie l’organisation du système. La figure 9.31 montre le résultat obtenu. Ceci est un simple affichage, mais il est possible d’aller beaucoup plus loin en complétant l’ontologie et de donner la possibilité de préciser les objectifs.

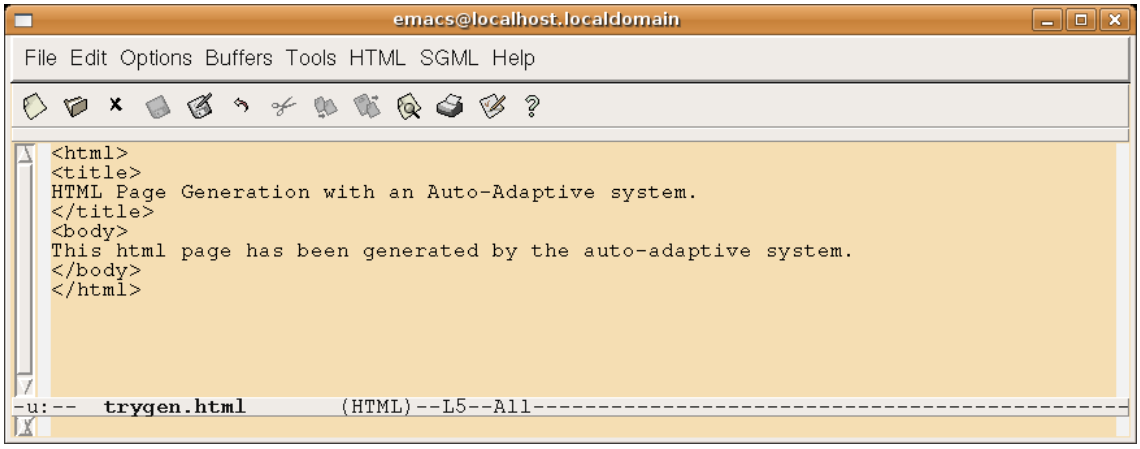

FIG. 9.29 – Le code de la page HTML.

FIG. 9.30 – Log de la génération HTML.

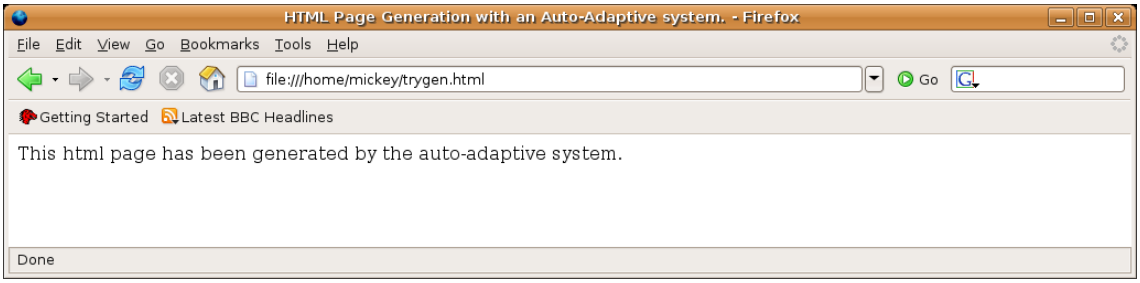

FIG. 9.31 – Le résultat de la génération HTML.

Chapitre 10

Développements en cours : acquisition on-line de connaissances nouvelles

10.1 Connaissances et apprentissage

L'apprentissage va permettre au système global contrôlant le robot d'évoluer dans la durée en acquérant de nouvelles connaissances et de nouvelles expériences afin de prendre des décisions de plus en plus précises et de plus en plus concrètes. Nous avons vu dans le chapitre sur l'état de l'art que la plupart des modèles de systèmes autonomes existant privilégient un apprentissage par modes, c'est à dire que le système doit passer obligatoirement sous un mode apprentissage avant de passer sur le mode traitement de l'information. Il serait intéressant pour un système autonome d'être capable, tout comme l'être humain ou l'animal, d'apprendre de lui-même simultanément à l'action qui est entreprise pour atteindre un objectif particulier. Le système, avec son architecture SMA, peut en effet :

1. utiliser ses connaissances agentifiées,
2. se voir augmenter ou modifier ses connaissances on-line par l'utilisateur,
3. augmenter ses connaissances de lui-même en générant de nouveaux agents, par conceptualisation.

Résumons ce qui a été décrit précédement. Le système global est un système auto-adaptatif ayant besoin d'un paramétrage spécifique[1] :

- Une liste de capteurs.
- Une liste d'effecteurs.
- Une ontologie correspondant à toute la connaissance disponible dans le système.
- Une liste d'expériences passées afin de donner à l'entité le moyen d'utiliser ses connaissances pour agir sur un environnement.

[1] Nous n'aborderons pas ici les paramètres techniques.

– Une liste d'objectifs.

Tous ces caractères pourront évoluer dans le temps. Il est donc primordial pour l'entité d'être capable de se servir d'un nouvel effecteur, par exemple dans le cas d'un ajout dynamique (la nouvelle capacité physique devra être liée à différentes connaissances pour être utilisée). Tous ces paramètres seront utiles pour l'apprentissage. Voici huit phases sur lesquelles nous nous basons pour l'apprentissage du système (ces phases ont été définies de manière pragmatique en analysant les résultats des expérimentations vues dans le chapitre 9) :

1. Constater qu'un élément présent dans une scène n'est pas connu.
2. Situer l'élément dans l'espace et le temps, et le lier au contexte courant.
3. Essayer de rapprocher l'élément à quelque chose de connu[1].
4. Parmi les éléments de cette liste de rapprochement, vérifier si certains éléments pourraient être utilisés pour atteindre l'objectif courant.
5. Si cette vérification se trouve positive, l'entité essaiera d'utiliser l'élément non connu en lui appliquant les expériences qui se sont avérées intéréssantes avec des éléments similaires.
6. Prendre en compte le retour sur les capteurs pour respecter la boucle systémique en vérifiant si l'entité se rapproche de l'objectif en cours.
7. Un identifiant temporaire est attribué à l'élément pour l'enregistrer dans l'ontologie et le prendre en compte dans l'expérience.
8. Un nom pourra être donné ultérieurement à cette nouvelle connaissance de manière directe par un superviseur humain ou de manière automatique en donnant un protocole de nommage au système.

Toutes ces phases possèdent une modélisation propre correspondant à une configuration spécifique du système.

10.1.1 Constat d'un élément non connu

Chaque élément reconnu dans une scène possède un taux de reconnaissance correspondent à un pourcentage grâce à l'usage d'un réseau de neurones déjà calibré. Lors de la reconnaissance des éléments, les agents s'activent laissant émerger la connaissance et l'expérience liée à ces éléments. La présence d'un élément non connu ne provoque aucune émergence, comme si l'élément n'était pas présent dans la scène. Voici les trois caractéristiques à ajouter au système afin de permettre à l'entité de constater qu'elle ne sait pas bien appréhender un élément et qu'elle va prendre cela en compte :

1. Une détection d'éléments connus ou inconnus, c'est-à-dire détecter la présence de quelque chose via les senseurs du robot.

[1] Le fait que l'élément soit vivant ou non pourra être défini grâce à une synergie de capteurs spécialisés ou non, comme un capteur de chaleur par exemple.

2. Un rôle représentant un élément non connu. Nous appellerons celui-ci “Unknown”.
3. Une expérience permettant de synchroniser la détection, l’émergence et le rôle “Unknown” pour constater la présence d’un élément non connu.

L’explication d’un rôle a été entreprise précédemment. Celui-ci est composé de plusieurs agents, de capacités de reconnaissance différentes, inter-connectés par un canal de communication constant. Cependant, ce rôle est particulier et bien plus complexe qu’un rôle standard. La reconnaissance d’images donnera des informations sur la présence d’un certain objet. A ce stade, il y aura une émergence dans le système. Si l’objet est connu, l’émergence des agents correspondant à la connaissance sera flagrante, c’est à dire elle sera supérieure à celle des agents liés aux autres connaissances. Dans le cas contraire, elle sera très faible et portera sur plusieurs objets présents dans l’expérience et la connaissance du système. C’est à dire que le système de reconnaissance confondra cet élément non connu avec certains éléments connus, ce qui provoquera une émergence minime sur tous ces éléments. Le rôle “Unknown” devra donc effectuer un contrôle sur le nombre d’émergences minimes se déroulant dans le système, ce qui résoud le dernier point vu plus haut : la synchronisation entre la détection, l’émergence, et le rôle. Le rôle “Unknown” pourra aussi se focaliser sur des capteurs spécifiques comme des capteurs de distance ou de mouvement afin d’accélérer la reconnaissance d’un élément non connu. La figure 10.1 présente le positionnement de ce rôle dans le système global. Bien que ce rôle soit spécifique, il fait tout de même partie de la liste des rôles.

Le rôle “Unknown” est bien spécifique mais n’a donc pas de place particulière dans le système. Ce rôle s’active lorsqu’une émergence minimale dépasse durant quelques secondes le nombre limite d’émergences minimales donné en paramètre. Ceci fonctionne dans le cas où un seul élément, qui est non connu, est présent dans la scène, ce qui n’est pas toujours le cas. Un élément non connu peut être en présence d’éléments connus. Une telle scène est schématisée sur la figure 10.2. Un élément non connu est détecté dans une scène si aucun élément connu ne peut être utilisé pour atteindre un objectif. De ce fait, l’émergence des éléments inutilisables est diminuée mais reste valide afin d’effectuer une comparaison avec l’élément non connu. Cette validité est obtenue grâce au fait que l’entité réduit la taille de la scène pour s’intéresser à l’élément non connu, le système n’a plus besoin d’inhiber certains éléments.

10.1.2 Situation spatiale et temporelle d’un élément non connu

Toute chose appréciée par le système possède une position dans l’espace et le temps. Nous suivons cette hypothèse pour que le système possède une expérience sur un élément non connu afin qu’il se rappelle de son utilisation. Comme vu précédemment, si un élément n’est pas présent dans une expérience, le système détectera sa présence mais ne saura pas comment l’utiliser. L’élément devra donc être présent

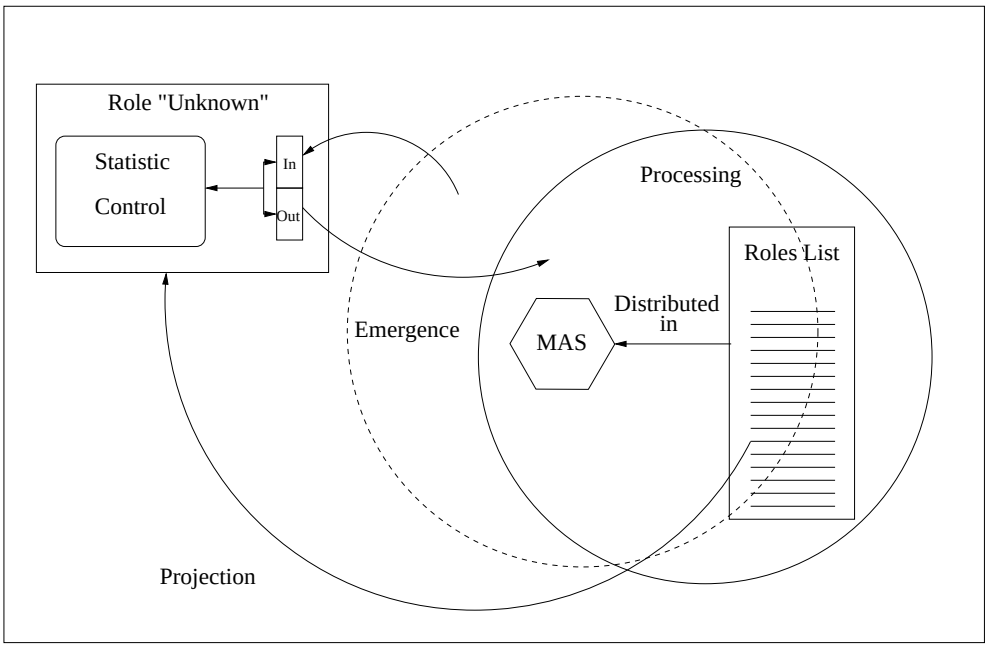

FIG. 10.1 – Positionnement du rôle “Unknown” dans le système.

dans au moins une expérience. Celle-ci regroupera le moment de la détection de l’objet ainsi que sa position dans l’espace. De ce fait, l’élément non connu devient connu mais n’est pas nommé, l’identification temporaire est exposée à la section 10.1.7.

10.1.3 Rapprochement d’un élément non connu à des éléments connus

Il est possible qu’un élément d’une scène soit inconnu du système, mais il est toutefois impossible que la morphologie de cet élément ne ressemble en aucun cas à certains éléments qui sont, eux, connus du système car tout objet possède des formes géométriques de bases (rectangle, triangle ...) qui peuvent amener à faire le rapprochement avec d’autres objets. Celui-ci effectuera donc un rapprochement d’objets connus en fonction de la reconnaissance, ce qui revient à mettre en évidence les objets ayant une émergence qui a de la signification. L’élément non connu sera proche de différents éléments connus.

10.1.4 Application d’expériences à un élément non connu

Afin de sélectionner les éléments connus pouvant être associés à l’élément non connu, le système applique différentes actions sur ce dernier pour en déduire sa méthode d’utilisation qui est la manière correcte de se servir de l’objet. Ces actions sont liées à l’expérience que possède le système sur les éléments présents dans sa

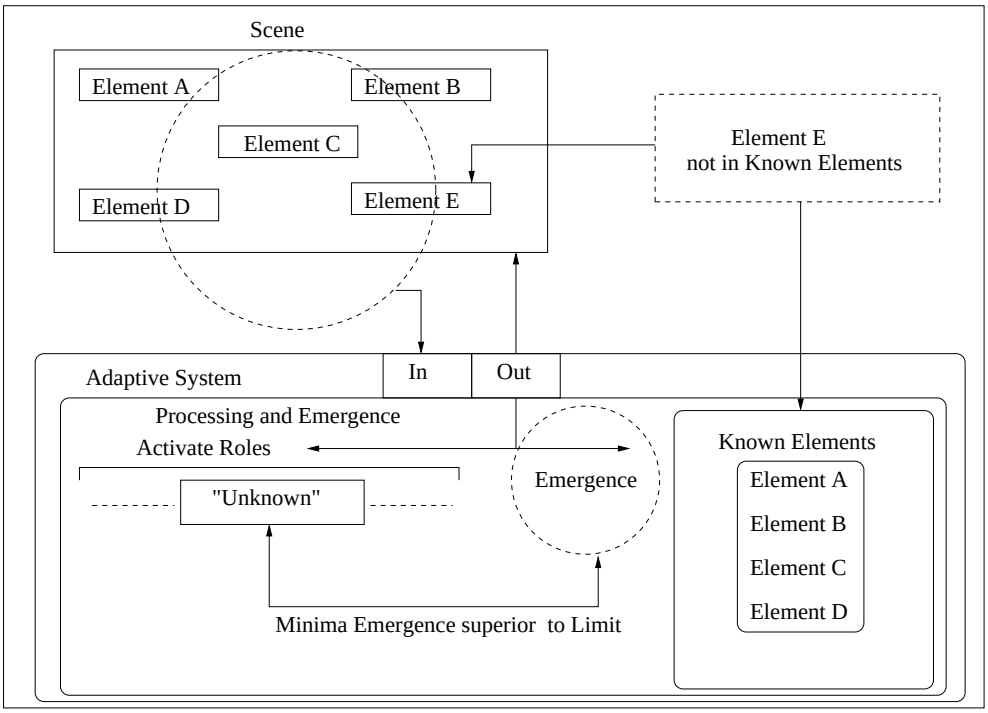

FIG. 10.2 – Activation du rôle “Unknown”.

connaissance. Les éléments connus retenus posséderont une méthode d’utilisation possible sur l’objet non connu. Les différents capteurs de mouvements et de reconnaissance permettront une émergence en ce qui concerne l’utilisation de l’élément non connu.

10.1.5 Test de besoin sur un élément non connu pour atteindre un objectif

En parallèle à l’application de l’expérience sur l’élément inconnu, le système vérifie si les différentes expériences sont susceptibles d’être utilisées afin d’aboutir à un objectif particulier. Si c’est le cas, et que tous les objets aux alentours ne sont pas utilisables ou pas utiles, le système essayera d’utiliser cet élément non connu pour l’instant. Ce dernier fait est important car par ces tests d’utilisation, le système se crée une expérience liée à l’élément non connu qui sera par la suite intégrée à la connaissance si cela s’avère utile.

10.1.6 Identification temporaire

Juste avant les tests d’expériences sur l’élément non connu, le système attribuera un identifiant à cet élément qui sera concaténé avec le mot “Unknown”. Cela

permettra au système d'être sûr qu'il utilise l'élément qu'il vient de découvrir. Celui-ci possèdera une morphologie agent particulière correspondant à plusieurs émergences traitées en même temps. Cette identification durera jusqu'à ce que le système pose un nom original à l'objet, soit par communication avec d'autres systèmes, soit par une communication directe avec l'utilisateur qui attribuera lui même un nom au nouvel élément.

10.1.7 Nommage

L'élément non connu va se transformer en élément connu car il va être inséré dans la connaissance du système en même temps que la nouvelle expérience acquise. Dans un premier temps, le système se servira de la concaténation des noms des éléments dont les expériences ont pu être appliquées à l'élément non connu. Cette mémorisation associée à la reconnaissance vidéo qui est implicite et à la valeur des différents capteurs permettra de réutiliser l'élément découvert récemment. De plus, il est possible de donner un autre nom à l'élément de manière dynamique, soit par une interface utilisateur, soit par comparaison de connaissances entre plusieurs systèmes.

10.1.8 Action et vérification du bon déroulement de l'action entreprise

Après la mémorisation de l'objet, une action pourra être entreprise afin de se rapprocher de l'objectif final. Si la modification de l'environnement et de la représentation de la scène, qui utilisent toutes les connaissances matérielles et non matérielles, se rapprochent de la forme de l'objectif, alors l'action aura été positive et le système continuera ses actions dans ce sens. Si l'action n'est pas positive, le système tentera d'utiliser le nouvel élément d'une manière différente.

10.2 Développements

Les éléments présentés dans ce chapitre sont en cour de développement grâce à l'acquisition d'une machine de calcul mise à disposition par L'Epitech, une école d'expertise informatique. Cette machine est un proliant ML 150 avec un Intel Xéon quadri-core possédant quatre méga-octets de mémoire cache et une fréquence de 1,86 GHz. Elle possède cinq giga-octet de mémoire vive et cinq cent giga-octet de disque dur. Le système Solaris 10 de Sun MicroSystem a été installé par un étudiant.

Cette machine permet aussi de développer un projet de vidéo surveillance global. Ce projet vise à utiliser plusieurs systèmes auto-adaptatifs (un par caméra et les deux robots Aibo) en y intégrant une nuées de réseaux de neurones permettant la reconnaissance d'éléments sur les images des caméras qui sont au nombre de dix. Ces éléments seront représentés par des connaissances dans l'ontologie qui

seront agentifiées pour être utilisé de manière dynamique. La communication entre les systèmes permettra de rendre compte à chacun de la situation des autres. Cette synchronisation permettra une prise de décision globale.

Chapitre 11

Conclusion et perspectives

Cette thèse approche plusieurs points importants dont les principaux sont le fait qu'un système multi-agent massif est capable de s'auto-organiser grâce à son contrôle morphologique et qu'il est possible de planifier des nuées de processus en temps-réel en fonction de différents objectifs. Les travaux présents dans ces écrits utilisent les résultats de recherches de différentes sciences comme l'informatique, la robotique, et les sciences de la complexité. L'analyse de ces différents résultats a mené à une proposition de solution utilisant leur lien synergique. Cette solution est un système auto-adaptatif pour le contrôle d'entités logicielles ou matérielles qui est en fait un planificateur dynamique de processus. Cette notion a été présentée dans un article scientifique [CC06b] faisant partie de ces travaux de recherches sous le nom de "Programmation Morphologique". La "Programmation Morphologique" est le fait de construire des graphes avec des mots ou des objets possédant des distances entre eux pour modifier la dynamique de la représentation de l'ontologie utilisée par le système auto-adaptatif pour orienter celui-ci dans son pilotage de plusieurs entités logicielles ou matérielles.

Les différentes expérimentations effectuées sur ce système ont montré quatre caractéristiques essentielles :

1. Les connaissances du système émergent comme une composition formant la scène, ce qui est la réalisation de l'émergence des entités actives. Celles-ci peuvent évoluer, le système s'adaptera en fonction de ses connaissances et de ses expériences passées représentées par les accointances entre les agents.
2. Il est possible d'insérer des émotions dans le système auto-adaptatif afin de transformer plus ou moins celui-ci en "machine émotionnelle". Les émotions prennent la forme de processus concurrents qui altèrent les objectifs rationnels du système selon les accointances qui existent entre les émotions et les connaissances. Cette concurrence est exploitée grâce au niveau d'énergie présent dans tous les agents du système.
3. Le système auto-adaptatif est capable de communiquer et de se synchroniser avec d'autres systèmes auto-adaptatifs de même type contrôlant des entités

hétérogènes. Ceci permet une décision de groupe et un partage de connaissances agentifiées, ce qui permet aux différentes entités d'évoluer en société. Ceci est possible en faisant transiter les représentations de scènes d'un système à l'autre de manière parallèle, le transit de connaissances et l'échange d'expériences est aussi possible.

4. Le système est un planificateur dynamique de processus. La "Programmation Morphologique", permet à l'utilisateur du système de ne pas avoir besoin de programmer pour automatiser une tâche vue comme une suite d'actions élémentaires. Celui-ci utilise la liaison qui existe entre les connaissances pour renseigner l'ontologie du système. Ainsi, il est possible d'insérer une grammaire simplifiée d'HTML pour pouvoir générer des pages de manière automatique et dynamique.

Le système auto-adaptatif découlant de ces recherches est générique. Il est capable de diriger un groupe d'entités hétérogènes afin de les faire tendre vers des objectifs spécifiques, mais non pas converger de manière optimale, ce qui est un autre problème. La planification dynamique de processus permet le développement par formes de connaissances et par formes d'objectifs en utilisant des graphes dynamiques. La géométrie est bien au coeur du fonctionnement du système.

Ce système auto-adaptatif permet de reprendre de nombreuses problématiques. Certains cas applicatifs peuvent être abordés comme les systèmes d'exploitations distribués, la robotique et la cybernétique et encore bien d'autres.

En ce qui concerne les systèmes d'exploitation distribués, un ordinateur est composé de matériel et de logiciel fonctionnant en synergie. Il existe des liens entre ces deux niveaux qui représentent l'expérience commune entre la connaissance et les interactions avec l'environnement de la machine. Créer un système d'exploitation global appréciant de lui-même ses activités permettrait d'utiliser la "Programmation Morphologique" pour diriger un système auto-adaptatif avec des objectifs et ainsi développer un système d'exploitation distribué sans Kernelland et sans Userland. Ce nouveau type de système d'exploitation serait appelé un système d'exploitation morphologique avec en interne un passage de messages massif permettant la communication entre les utilisateurs et le matériel. Ce système serait totalement autonome et pourrait s'installer lui-même sur un LAN et permettrait aux utilisateurs de créer des automates spécifiques sans langage de programmation.

Pour la robotique et la cybernétique, beaucoup de recherches sont en cours pour remplacer l'homme dans des situations difficiles ou dangereuses. Si nous créons une ontologie basée sur une situation spécifique et les capacités physiques des robots, nous pourrons diriger des groupes de robots hétérogènes dans un environnement hostile en profitant d'une décision de groupe efficace. Le système auto-adaptatif est capable de se connecter de lui-même aux différents capteurs des robots sans configuration technique longue. La "Programmation Morphologique",

quant à elle, permet de modifier certains paramètres en temps-réel pour diriger directement tous les robots sous la supervision du système autonome. Ceci veut dire que tous les robots sont continuellement connectés à un système de contrôle distribué sans aucune centralisation. Le système auto-adaptatif est capable de gérer de multiples types de capteurs pour copier le comportement de l'être humain dans le cas d'un Humanoïde par exemple.

Les résultats de ces recherches montrent que le système auto-adaptatif pourra être utilisé pour construire des systèmes encore plus complexes dans des domaines multiples. Tous ces systèmes pourront communiquer entre eux et trouver de nouvelles informations et connaissances sur Internet. Une attention particulière devra être portée sur la relation homme/machine afin que ce type de système reste intelligible et contrôlable.

Nous effectuons actuellement des développements supplémentaires traitant des ontologies très différentes, permettant de valider la notion de généricité de notre modèle et ouvrant sur la réalisation de la conscience artificielle : notion de self.

Table des figures

1.1 Boucle systémique 15
1.2 Architecture systémique 16

2.1 Modèle de base d'une cellule synthétique 21
2.2 Evolution du profil de concentration des deux types de molécules (Source : Santa Fe Institute). 22

3.1 Architecture d'un système spatial 30
3.2 Architecture du "Remote Agent" (Source : NASA). 31
3.3 Hiérarchie d'une constellation de satellites 32
3.4 Approche multi-agent(Sources : Axlog Ingénierie, ESA). 33
3.5 Architecture logicielle d'un agent (vision 1)(Sources : Axlog Ingénierie, ESA). 34
3.6 Architecture logicielle d'un agent (vision 2)(Sources : Axlog Ingénierie, ESA). 35
3.7 Modèle de planification par programmation par contraintes(Source : Axlog Ingénierie). 36

4.1 Courbe de difficulté du maintien de la température ambiante. . . . 41
4.2 Carte PCI du SmartTherm. 42

5.1 Une scène et ses différentes entités. 48
5.2 Une représentation de scène simple. 49
5.3 Une représentation de scène plus complexe. 50
5.4 Deux représentations de scène. 51
5.5 Deux représentations de scène communiquant entre elle. 52

6.1 L'ontologie est représenté par un graphe. Tous les éléments présents dans la base de connaissances possèdent des accointances avec d'autres éléments, ces accointances sont représenté par des liens. Ici, chaque connaissance est numéroté (K1..K14). 55
6.2 L'activation des agents entraîne une déformation des plans de classification ontologique. 56
6.3 Les agents respectent une classification ontologique pour former des groupes d'activités plus ou moins importants en fonction du contexte ou l'entité se trouve. En plus de la communication intra-groupe, tous les groupes communiquent entre eux en donnant la possibilité au système de les agréger. 57
6.4 Le graphe des émotions est attaché à celui de l'ontologie. Il est possible de décrire et de donner une orientation émotionnelle au système seulement avec des mots. 59

6.5 Le cadre en pointillé représente une section montrant différents liens entre les émotions, les verbes et les objets. Avec un peu plus d'expérience, beaucoup d'éléments dans l'ontologie peuvent être liés. Ici, il est plaisant pour le système de jouer à la balle, mais la peine est aussi une hypothèse possible car si la taille de la section change, une peine venant du passé peut apparaître. 61
6.6 L'architecture d'un agent reste simple car c'est l'organisation interne qui fait la puissance du système. Cette architecture doit être minimale afin que son utilisation soit la plus souple possible. . . . 62
6.7 Chaque agent doit effectuer certaines actions pour garder un niveau d'énergie suffisant pour exister. Tous les agents stockent les différentes actions à effectuer afin de respecter ses objectifs internes qui sont directement liés à l'ontologie. 63
6.8 Création d'un thread 65
6.9 Envoi d'un message entre deux threads 65
6.10 Thread d'un agent écoutant un port particulier 66
6.11 Pattern matching en Oz 66
6.12 Création d'agents de différents types 67
6.13 Les statistiques sur la communication des agents structurants permettent de construire des histogrammes afin d'avoir une vision externe du comportement du système pour sélectionner les formes aptes à la bonne réalisation des objectifs. 68
6.14 Toute l'ontologie du système est distribuée dans le système multi-agent massif. Dans cette ontologie, on retrouve aussi la connaissance correspondant aux différents mouvements d'actions à effectuer dans un environnement. 70
6.15 La forme géométrique découlant de l'activation de certains agents structurants permet l'orientation et la prise de décision du système afin de respecter les objectifs passés en paramètre. 71
6.16 La station de travail permettant l'évolution dans un environnement instable n'est pas embarquée dans le matériel. Cependant, la communication entre la station et le matériel reste continue pour traiter les informations en temps-réel. 72
6.17 Boucle systémique. 72
6.18 Une boucle systémique se forme entre l'entité et l'environnement pour former un tout indissociable. La communication entre la partie matérielle et la partie logicielle est continue et ininterrompue. Une action modifiera l'environnement ce qui provoquera une nouvelle action. 73
6.19 La totalité de l'ontologie est distribuée dans un système multi-agent massif où chaque agent utilise huit processus distincts afin d'évoluer dans son environnement restreint. 73

6.20 La construction d'une représentation courante est une activation des agents liés à la connaissance nécessaire pour la représenter. Les agents communiquent entre eux grâce à un passage de messages. Tous ces messages permettent de créer une forme particulière qui sera utilisée pour la prise de décision. 74

6.21 Le robot se fera une interprétation de scène qui lui permettra de comparer la forme géométrique créée avec ses objectifs. Si ces derniers sont différents, une décision est tout de même prise mais le système sera orienté en fonction des objectifs. 75

6.22 Organisation Aspectuelle et Représentation Morphologique 76

6.23 La morphologie des groupes d'agents structurants permet la création d'histogrammes qui seront comparé aux histogrammes générés à partir des objectifs passé en paramètre par l'utilisateur. Cela permet d'orienter le système vers certaines formes. 77

6.24 Les différents liens existants dans le système entre les agents et entre les groupes forment une ontologie distribuée en forme de graphe où chaque nœud est divisé en plusieurs processus. 78

6.25 Sélection d'un objectif dans une activation générale du Graphe Ontologique Global. Il faut que la forme de l'objectif corresponde à la forme générale du système. 84

7.1 Modèle conceptuel de données pour l'ontologie 90

7.2 Diagramme de classes concernant la gestion des capteurs. 91

7.3 Extrait du diagramme de classes concernant la création des entités. 93

7.4 Création d'une suite d'agent correspondant à un type spécifique. . 94

7.5 Modification en temps-réel des paramètres d'un agent Aspectuel. . 95

7.6 Structure de base d'un rôle traité par un agent Aspectuel. 99

7.7 Méthode de récupération des expériences par agent. 100

7.8 Liste des paramètres pour un rôle cognitif. 100

7.9 Liste des paramètres pour un rôle d'objectif. 100

7.10 Gestion interne des messages au niveau agent. 101

7.11 Auto-analyse statistique d'un agent au niveau de son envoi de messages. 102

8.1 Fenêtre de configuration de l'ontologie. Celle-ci permet d'insérer la connaissance du système correspondant à un environnement et ses contextes associés 104

8.2 Communication entre agents durant un traitement d'information d'un contexte environnemental spécifique. 107

8.3 Le système est capable de structurer les connaissances en *n* dimensions ontologiques. Il est donc possible d'effectuer une découpe du graphe tri-dimensionnel ontologique par plan afin de constater l'activation de certaines connaissances du système. Cette découpe peut se faire en fonction du classement employé dans l'ontologie (objet, verbes, couleurs, ...). 108
8.4 Il est possible à chaque instant de suivre la "représentation courante" de l'entité. Celle-ci évolue dans le temps et s'oriente vers des objectifs bien précis. 109
8.5 Le système fonctionne par analyse de formes géométriques que l'on appelle morphologie. Il est possible de donner au système des objectifs, et de suivre en temps-réel l'évolution de ceux-ci grâce à la différence entre la forme cible et la forme courante. 110
8.6 Le comportement de l'entité peut être suivi en temps réel. Il est important d'associer toutes les vues du monitoring afin de déceler des écart de comportement modifiables par un changement de paramètres. 111

9.1 Schéma global d'une expérience. 116
9.2 Dès l'exécution du système, des commandes sont envoyées au robot. Celui-ci les exécute directement. 119
9.3 Le système s'adapte en fonction des objectifs. Des messages de morphologies sont envoyés aux agents structurants. 120
9.4 La boucle systémique ne s'arrête jamais et le système s'auto-adapte en fonction de son environnement. Tant que le système fonctionnera, les agents structurants recevrons des messages de morphologies. 120
9.5 A l'initialisation du système, les plans ontologiques restent statiques. 121
9.6 Durant l'évolution du robot dans l'environnement, les plans ontologiques se déforment laissant apparaître une activation particulière de connaissances. 122
9.7 Une autre déformation des plan ontologiques. Ces plans dépendent de la classification employée dans l'ontologie. 123
9.8 Une représentation courante. 124
9.9 Une deuxième représentation courante. 125
9.10 Une troisième représentation courante. 126
9.11 Une première évaluation de l'atteinte des objectifs. 127
9.12 Une seconde évaluation de l'atteinte des objectifs. 128
9.13 Une troisième évaluation de l'atteinte des objectifs. 129
9.14 Les plans ontologiques concernant les émotions. 131
9.15 Divergence des représentations émotionnelles. 132
9.16 Les deux systèmes auto-adaptatifs communiquent par pattern de mémoire. La perception de l'un va compléter la perception de l'autre pour former une nouvelle analyse et prendre une décision de groupe. 133

9.17 Au lancement des deux systèmes, l'activation des agents est inexistante des deux côtés. 134
9.18 Le robot capable de se rendre compte de la scène génère une activation de ses connaissances. Le deuxième robot ne connaît toujours pas l'environnement dans lequel il évolue. 135
9.19 Bien que le deuxième robot utilise l'interprétation de la scène du premier robot, il génère un point focal différent de son homologue. 135
9.20 Les objectifs sont atteints de manière différentes. 136
9.21 Plans ontologiques des objets et des verbes sans modification. . . 138
9.22 Point focal sans modification. 139
9.23 Plans ontologiques des objets et des verbes avec modification. . . 140
9.24 Point focal avec modification. 141
9.25 Eléments de classification pour le HTML. 143
9.26 Liste des agents. 143
9.27 Expérience liée à la génération HTML. 144
9.28 Description de l'agent concernant une page HTML. 144
9.29 Le code de la page HTML. 145
9.30 Log de la génération HTML. 145
9.31 Le résultat de la génération HTML. 146

10.1 Positionnement du rôle "Unknown" dans le système. 150
10.2 Activation du rôle "Unknown". 151

C.1 Schéma global de l'Ontological Transcription Core. 180
C.2 Fonctionnement général Morphoviewer. 182

D.1 La classification de la connaissance présente dans l'ontologie. . . 183
D.2 Première page de la connaissance présente dans l'ontologie. . . . 184
D.3 Deuxième page de la connaissance présente dans l'ontologie. . . . 185
D.4 Troisième page de la connaissance présente dans l'ontologie. . . . 186
D.5 Quatrième page de la connaissance présente dans l'ontologie. . . . 186
D.6 Les expériences de l'ontologie. 187
D.7 Première page du listing des accointances. 187
D.8 Deuxième page du listing des accointances. 188
D.9 Troisième page du listing des accointances. 188

E.1 Environnement de développement du MOZART PROGRAMMING SYSTEM 191
E.2 Types primaires du langage OZ 192
E.3 Le OZ PANEL 198
E.4 Arbre de recherche pour le problème SEND MORE MONEZ 206
E.5 OZ EXPLORER 207

Bibliographie

[aES03] S. Solomon anb E. Shir. Complexity ; a science at 30. *Europhysics News*, 34(2) :409–411, 2003.

[Ald69] Brian Aldiss. Super-toys last all summer long. *Harper's Bazaar*, 1969.

[All97] K. T. Alligood. *Chaos : an introduction to dynamical systems*. Springer-Verlag New York, LLC., 1997.

[Asi72] Isaac Asimov. *Les robots*. j'ai lu, 1972.

[BAD+03] W. Bluethmann, R. Ambrose, M. Diftler, E. Huber, M. Goza, C. Lovchik, and D. Magruder. Robonaut : A robot designed to work with humans in space. In *Autonomous Robots*, volume 14, pages 179–207, 2003.

[Bar02] A. L. Barabasi. *Linked : the new science of networks*. Perseus ; New York, 2002.

[BBG+03] C. Breazeal, A. Brooks, J. Gray, G. Hoffman, C. Kidd, H. Lee, J. Lieberman, A. Lockerd, and D. Mulanda. Humanoid robots as cooperative partners for people. *International Journal of Humanoid Robots*, 2003.

[BDD00] A. Bechara, H. Damasio, and A.R. Damasio. Emotion, decision making and the orbitofrontal cortex. *Cerebral Crotex*, 10(3), 2000.

[BDF+98] D. Bernard, G. Dorais, C. Fry, E. Jr, B. Kanefsky, J. Kurien, W. Millar, N. Muscettola, P. Nayak, B. Pell, K. Rajan, N. Rouquette, B. Smith, and B. Williams. Design of the remote agent experiment for spacecraft autonomy, 1998.

[BDT99] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm intelligence*. Oxford University Press, New York, 1999.

[BJ02] J.P. Baquiast and C. Jacquemin. Automates intelligents : Dossier super-intelligences, 2002.

[BJ04] J.P. Baquiast and C. Jacquemin. Automates intelligents, la nasa fera dorénavant systématiquement appel aux robots intelligents, 2004.

[BP97] Douglas E. Bernard and Barney Pell. Designed for autonomy : Remote agent for the new millennium program. 1997.

[Bro01] R. Brooks. The relation between matter and life. *Nature*, 409 :409–411, 2001.

[Bro02] R. A. Brooks. *Flesh and Machines*. Pantheon Books, 2002.

[BS93] Rodney Brooks and Lynn A. Stein. Building brains for bodies. Technical Report AIM-1439, 1993.

[CAdN02] CNRS, ATILF, and Université de Nancy. Le trésor de la langue française informatisé. http ://atilf.inalf.fr/tlfv3.htm, 2002.

[Cam02] Mickaël Camus. Misure - mission managment system for uninhabited air vehicles and new techniques : Planification anytime. Technical report, {Epitech.} - Ecole pour l'Informatique et les nouvelles TECHnologies, sept 2002. Mémoire de fin d'étude.

[Cam05] J.C Campagne. *Morphologie et système multi-agent*. PhD thesis, Université Pierre et Marie Curie, 2005.

[Car00] A. Cardon. *Conscience artificielle et systèmes adaptatifs*. Eyrolles, 2000.

[Car04] A. Cardon. *Modéliser et concevoir une machine pensante*. Vuibert, 2004.

[CC05] M. Camus and A. Cardon. Towards an emotional decision-making. In *Second GSFC/IEEE WRAC 2005 : Workshop on Radical Agent Concept, NASA Goddard Space Flight Center*, 2005.

[CC06a] M. Camus and A. Cardon. An adaptive system to control robots : ontology distribution and treatment. *WSEAS International*, 2006.

[CC06b] Mickael Camus and Alain Cardon. Dynamic programming for robot control in real-time : Towards a morphology programming. In Hamid R. Arabnia, editor, *IC-AI*, pages 389–395. CSREA Press CSREA Press, 2006.

[CCC05] A. Cardon, J.C Campagne, and M. Camus. A self-adapting system generating intentional behavior and emotions. In *Second GSFC/IEEE WRAC 2005 : Workshop on Radical Agent Concept, NASA Goddard Space Flight Center*, 2005.

[CL98] A. Cardon and F. Lesage. Toward adaptative information systems : considering concern and intentionnality. In *KAW 98, Banff, Canada*, 4 1998.

[CNR01a] CNRS. Robea - programme national robotique et entités artificielles, 2001.

[CNR01b] AIST CNRS. Joint japanese-french robotics laboratory (jrl), 2001.

[CP01] C.Guettier and J.C. Poncet. Multi-levels planning for spacecraft autonomy. In *6th International Symposium on Artificial Intelligence and Robotic Applications for Space, Montreal*, 2001.

[Dam95] A. R. Damasio. *L'erreur de Descartes*. Odile Jacob, 1995.

[Dam05] A. R. Damasio. *Spinoza avait raison : joie et tristesse, le cerveau des émotions*. Odile Jacob, 2005.

[Dav04] Leonard David. Nasa studies robot servicing of hubble, 2004.

[Dev03] R. L. Devaney. *An Introduction to Chaotic Dynamical Systems, 2nd ed*. Westview Press, 2003.

[Doo97] K. Dooley. A complex adaptive systems model of organization change. *Nonlinear Dynamics, Psychology and Life Science*, 1(1) :69–97, 1997.

[Ecc93] J. C. Eccles. *Evolution du cerveau et création de la conscience*. Flammarion, 1993.

[EFSM99] H. Eriksson, R. Fergerson, Y. Shahar, and M. Musen. Automatic generation of ontology editors. In *Knowledge Acquisition Workshop*, Banff, Alberta, Canada, 1999.

[Esc71] M. C. Escher. *The Graphic Work of M.C. Escher*. Ballantine, 1971.

[GM95] M. Gell-Mann. What is complexity ? *Complexity*, 1(1), 1995.

[Gol99] J. Goldstein. Emergence as a construct : history and issues. *Emergence*, 1(1), 1999.

[GPB99] C. Guettier, J.C. Poncet, and E. Bornschlegl. Automatic planning for autonomous spacecraft constellation. In *2nd NASA International workshop on planning and scheduling for space*, 1999.

[Gru93] T. R. Gruber. Towards Principles for the Design of Ontologies Used for Knowledge Sharing. In N. Guarino and R. Poli, editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation*, Deventer, The Netherlands, 1993. Kluwer Academic Publishers.

[Gut90] M. Gutzwiller. *Chaos in Classical and Quantum Mechanics*. Springer-Verlag New York, LLC., 1990.

[Hei88] M. Heidegger. *Qu'est-ce qu'une chose ?* Gallimard, 1988.

[Hof99] D. R. Hofstadter. *Godel, Escher, Bach : An Eternal Golden Braid*. Basic Books, 1999.

[HSW93] M. Henz, G. Smolka, and J. Wurtz. Oz—a programming language for multi-agent systems. In *International Joint Conference on Artificial Intelligence*, Chambery, France, 1993.

[Jac04a] Christophe Jacquemin. Automates intelligents : Rabbit, prototype français de robot bipède marcheur et coureur, 2004.

[Jac04b] Christophe Jacquemin. Automates intelligents : Robots japonais de nouvelle génération : stratégies et opportunités, 2004.

[KMK+04] E .Kubinyi, A. Miklosi, F. Kaplan, M. Gacsi, J. Topal, and V. Csanyi. Social behaviour of dogs encountering aibo, an animal-like robot in a neutral and in a feeding situation behavioural processes. *Behavioural Processes*, 65(3) :231–239, 2004.

[KS01] Stanley Kubrick and Steven Spielberg. A.i., 2001.

[KW01] N. Kolb and I. Whishaw. *Cerveau et Comportement*. De Boeck Universite, 2001.

[Lar03] M. Larivey. *La puissance des émotions*. Editions de l'homme, 2003.

[LDD+02] L. Lafortune, M. Daniel, P. Doudin, F. Pons, and O. Albanese. *Pédagogie et psychologie des émotions : Vers la compétence émotionnelle*. Presses de l'Université du Quebec, 2002.

[LK00] J.S. Lerner and D. Keltner. Beyond valence : Toward a model of emotion-specific influences on judgment and choice. *Cognition an Emotion*, 14 :473–493, 2000.

[LS04] H. Liu and P. Singh. Conceptnet : A practical commonsense reasoning toolkit. *BT Technology Journal*, 2004.

[MB00] B. T. Messmer and H. Bunke. Efficient subgraph isomorphism detection : A decomposition approach. *IEEE Trans. Knowl. Data Eng.*, 12(2) :307–323, 2000.

[MH87] F. Murtagh and A. Heck. Multivariate data analysis. *Kluwer Academic Publishers, Dordrecht*, 1987.

[Min88] M. Minsky. *Society of Mind*. Simon and Schuster, 1988.

[Mit06] M. Mitchell. Complex systems : network thinking. Technical report, Santa Fe Institute, 2006. Working paper.

[Moo90] F. Moon. *Chaotic and Fractal Dynamics*. Springer-Verlag New York, LLC., 1990.

[MS06] J. Macia and R. V. Sole. Synthetic-turing protocells : vesicles self-reproduction through symmetry-breaking instabilities. Technical report, Santa Fe Institute, 2006. Working paper.

[New82] A. Newell. The knowledge level. *Artificial Intelligence*, 18(1) :87–127, 1982.

[NOK+02] T. Nagata, M. Ohono, J. Kubokawa, H. Sasaki, and H. Fujita. A multi-agent approach to unit commitment problems. In *2002 IEEE Power Engineering Society Winter Meeting. Conference Proceedings (Cat. No.02CH37309)*, volume 1, pages 64–9, 1 2002.

[Ott02] E. Ott. *Chaos in Dynamical Systems*. Cambridge University Press New York, 2002.

[otUoC06] The Regents of the University of California. Emergent nature. Technical report, University of California, Santa Barbara, 2006. Working paper.

[PAF+03] D. Purves, G. J. Augustine, D. Fitzpatrick, L. C. Katz, A. S. LaMantia, J. O. McNamara, and M. S. Williams. *Neurosciences*. De Boeck Universite, 2003.

[PBC+96] B. Pell, D. Bernard, S. Chien, E. Gat, N. Muscettola, P. Nayak, M. Wagner, and B. Williams. A remote agent prototype for spacecraft autonomy, 1996.

[Pon02] J.C. Poncet. Présentation du projet misure, 2002.

[PV04] O. PV. Le cerveau et l'amour. *La Recherche*, (380) :32–39, 11 2004.

[RH04] P.V. Roy and S. Haridi. *Concepts, Techniques, and Models of Computer Programming*. MIT Press, 2004.

[Rio02] Y. Rio. Architecture et systèmes spatiaux, 2002.

[Rit06] T. Ritchey. Modelling complex socio-technical systems using morphological analysis. Technical report, Swedish Morphological Society, 2006. Working paper.

[SBM97] A. Sahai, S. Billiart, and C. Morin. Astrolog : A distributed and dynamic environment for network and system management, 1997.

[SC01] L. A. Segel and I. R. Cohen. *Design principles for the immune system and other distributed autonomous systems*. Oxford University Press, New York, 2001.

[See99] R. Seel. Complexity and organisation development - an introduction. *http ://www.new-paradigm.co.uk/complex-od.htm*, 1999.

[Sim01] H. A. Simon. *The science of the artificial*. MIT Press, 2001.

[SK02] R. Squire and E. R. Kandel. *La mémoire*. De Boeck Universite, 2002.

[SMRM06] R. V. Sole, A. Munteanu, C. Rodriguez, and J. Macia. Synthetic protocell biology : from reproduction to computation. Technical report, Santa Fe Institute, 2006. Working paper.

[Spr03] J. C. Sprott. *Chaos and Time-Series Analysis*. Oxford University Press, 2003.

[Tho89] R. Thom. *Modèles mathématiques de la morphogénèse*. Christian Bourgois, 1989.

[Tho99] R. Thom. *Paraboles et catastrophes*. Flammarion, 1999.

[Tho01] Michel H. Thoby. Myriade : Cnes micro-satellite progam. In *Fifteenth Annual AIAA/USU Conference on Small Satellites*, 2001.

[Tra02] Thomas P. Trappenberg. *Fundamentals of Computational Neuroscience*. OXFORD Unibersity Press, 2002.

[Tur52] A. Turing. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society of London*, B(641) :37–72, 1952.

[Wal92] M. Mitchell Waldrop. *Complexity : The Emerging Science at the Edge of Order and Chaos*. Simon and Schuster, 1992.

[WJXyJcXf03] Z. Wei-Jun, S. Xiang-yang, W. Ji-cai, and H. Xiao-feng. Study on the simulation mechanism for crisis decision-making based on multi-agent technique. *Systems Engineering and Electronics*, 25 :959–1011, 8 2003.

[Wol02] S. Wolfram. *A new kinf of science*. Landmark, 2002.

Quatrième partie

Annexes

Annexe A

Définitions de l'autonomie

(Trésors de la langue Française)

[...]
AUTONOMIE, subst. fém.
A. [En parlant d'un pays, d'une province, d'une commune, ou encore d'une nation, d'un peuple, gén. p. oppos. à une collectivité intégrante ou intégrée]

1. Fait de se gouverner par ses propres lois (p. anal. avec le droit dont jouissaient, sous les Romains, certaines villes grecques). P. ext. Fait, pour une collectivité, de s'administrer elle-même :

   (a) Chaque ville tenait fort à son autonomie ; elle appelait ainsi un ensemble qui comprenait son culte, son droit, son gouvernement, toute son indépendance religieuse et politique. FUSTEL DE COULANGES, La Cité antique, 1864, p. 259.

   (b) Les Alsaciens-Lorrains subissent un régime d'exception dans l'Empire allemand et ils demandent à être traités comme le sont les Bavarois, les Saxons, les Wurtembourgeois ; voilà l'autonomie qu'ils réclament et que d'ailleurs on leur refuse. Mais il n'est nullement question de leur faire une situation autonome entre la France et l'Allemagne. Non, ce n'est pas d'une telle autonomie, d'une indépendance quelconque qu'il est question entre les Alsaciens-Lorrains et les Allemands. BARRÈS, Mes cahiers, t. 8, 1909-10, p. 26.

   SYNT. Autonomie administrative, douanière, économique, fiscale, politique ; autonomie d'une commune, des colonies, des forces ; les autonomies corporatives, locales ; perdre, réclamer, refuser, retrouver l'autonomie ; tenir à son autonomie ; jouir de l'autonomie ; être un champion de l'autonomie ; être jaloux de son autonomie ; besoin, volonté d'autonomie.
   Spéc. Autonomie des ports. „Régime administratif d'un port qui reçoit

par décret la personnalité morale et s'administre lui-même. Ex. : Bordeaux, Le Havre. (L. 12 juin 1920)“ (CAP. 1936).

2. Fait, pour une collectivité, d'assumer et de vivre son particularisme, son individualité morale, culturelle :

   (a) Le secret, c'est que la Provence a ses dieux, qu'elle retrouvera son autonomie, sa personnalité, sa langue. BARRÈS, Mes cahiers, t. 13, 1921, p. 113.

B. P. anal.

1. En parlant d'une pers.
   – Faculté de se déterminer par soi-même, de choisir, d'agir librement :

     (a) ... le désir d'autonomie est d'autant plus prononcé que la dépendance a été plus ressentie, mais il est aussi d'autant plus redouté par le sujet comme dangereux ou coupable que s'est formée en lui une secrète valorisation de sa soumission. Le révolutionnaire hésitant ou déchiré naît parfois de ce conflit... MOUNIER, Traité du caractère, 1946, p. 442.

   – Liberté, indépendance morale ou intellectuelle :

     (a) C'est te déclarer libre, indépendant, sans maître, sans obligation ; c'est par conséquent t'affranchir de la loi morale. Le principe de ton quiétisme, c'est donc un refus d'obéissance, c'est une sécession et en quelque sorte une révolte. Si dur, si triste, si pénible que soit l'isolement, il flatte néanmoins notre instinct d'antivasselage ; il nous crée une autonomie altière et entière, il nous fait souverains, souverains sans sujets, sans puissance, sans grandeur, mais n'ayant à s'humilier devant rien ni devant personne. AMIEL, Journal intime, 1866, p. 442.

     SYNT. Autonomie
     intellectuelle, morale ; l'autonomie de la conscience, de la pensée, de la raison, de la volonté ; affirmer son autonomie.
   – PÉDAG. „Organisation scolaire telle que les écoliers participent, dans une mesure plus ou moins grande, au choix des matières enseignées et à la discipline générale de l'école, de manière à apprendre à se gouverner eux-mêmes“ (PIÉRON 1963). L'autonomie des écoliers.
   – PHILOSOPHIE :

     (a) Dans ce paradoxe d'une « détermination imprévisible » ne pourrait-on reconnaître la synthèse de nécessité et de contingence que les grands métaphysiciens identifièrent toujours à la liberté du sage, et qu'ils appellent « autonomie » parce qu'elle associe la spontanéité à la loi ? JANKÉLÉVITCH, Le Je-ne-sais-quoi et le presque-rien, 1957, p. 211.

   – PSYCHOLOGIE :

(a) Quand la psychanalyse a remplacé un conflit inconscient et névrotique par un conflit conscient et humain, elle n'a pas, remarque Dalbiez, apporté une solution à ce dernier conflit, que seul le malade peut résoudre. Mais elle a haussé d'un plan les conditions de l'action et donné de meilleurs cartes à l'autonomie du moi contre les tyrannies du surmoi. MOUNIER, Traité du caractère, 1946, p. 730.

2. Spéc. [En parlant de la struct. ou du fonctionnement d'organismes naturels ou créés par l'industr. hum.]
   – ANAT. et PHYSIOL. L'autonomie biologique de la cellule :
     (a) Chacun de ces éléments possède son autonomie, c'est-à-dire sa vie propre. Ils ont chacun des propriétés spéciales (...) il y a aussi subordination et harmonie dans les mécanismes vitaux, sans quoi il ne pourrait pas y avoir d'unité ni d'ensemble dans la vie de l'organisme total. C. BERNARD, Principes de méd. exp., 1878, p. 11.
   – P. ext. Autonomie biologique de l'être :
     (a) Adieu, vous que j'aimais. Ce n'est point ma faute si le corps humain ne peut résister trois jours sans boire. Je ne me croyais pas prisonnier ainsi des fontaines. Je ne soupçonnais pas une aussi courte autonomie. On croit que l'homme peut s'en aller droit devant soi. On croit que l'homme est libre... On ne voit pas la corde qui le rattache au puits, qui le rattache, comme un cordon ombilical, au ventre de la terre. S'il fait un pas de plus, il meurt. SAINT-EXUPÉRY, Terre des hommes, 1939, p. 237.
   – TECHNOL. (aéron., astronaut.). Autonomie cinétique ; autonomie de manœuvre, de vol ; autonomie d'un avion, d'un navire, d'un véhicule. PRONONC. : [] ou [-]. Également [-] (PASSY 1914).
   – ÉTYMOL. ET HIST. 1. 1596 « fait de se gouverner d'apr. ses propres lois » (HULSIUS, Dict. françois-alemand et alemand-françois d'apr. Behrens ds Z. fr. Spr. Lit. t. 23, 2e part., p. 12) attest. isolée ; p. ext. av. 1815 « indépendance » contexte philos. (VILLERS, Kant, p. 138 ds LITTRÉ : Descartes effaça le honteux jurer sur la parole du maître, qui ôtait toute autonomie à la raison). 2. 1762 hist. anc. (Ac.). Empr. au gr. « droit de se régir par ses propres lois, indépendance, autonomie (en parlant d'un État) » (THUCYDIDE 3, 46 ds BAILLY).

   STAT. Fréq. abs. littér. : 303. Fréq. rel. littér. : XIXe s. : a) 19, b) 198 ; XXe s. : a) 240, b) 1 021. BBG. AQUIST. 1966. BACH.-DEZ. 1882. BARR. 1967. BASTIN 1970. BATTRO 1966. CAP. 1936. DUB. Pol. 1962, p. 68, 91, 105, 132, 134, 136, 187. FOULQ.-ST-JEAN 1962. FRANCK 1875. GOBLOT 1920. JULIA 1964. LAFON 1969. LAL. 1968. Lar. comm. 1930. LEMEUNIER 1969. LITTRÉ-ROBIN 1865. Méd. Biol. t. 1 1970. MIQ. 1967. PIÉRON 1963. RÉAU-ROND. 1951. ROMEUF t. 1 1956.

[...]

Annexe B

Définitions de la morphologie

(Trésors de la Langue Française)

[...]
MORPHOLOGIE, susbt. fém. MORPHOLOGIE, susbt. fém.

A. 1. Science, étude des formes. Goethe n'est pas géomètre (...) Il ne sent pas que l'Algèbre est aussi une Morphologie, et une génération en quelque sorte organique du nombre, dont elle définit les espèces, les transformations, la structure (VALERY, Variété IV, 1938, p.114). Tous les minéralogistes de l'époque [début du XIXe s.] prêtaient beaucoup d'attention à la description et à la classification des formes cristallines (morphologie cristalline) (Hist. gén. sc., t.3, vol.1, 1961, p.346). V. forme ex. 54.

2. En partic.
a) BIOL. Etude de la forme extérieure et de la structure des êtres vivants. Morphologie biologique, organique, somatique; morphologie générale, comparative; morphologie abdominale, crânienne, dentaire; morphologie florale, végétale. L'embryologie est le fondement le plus sûr de la morphologie et de l'anatomie comparée (CAULLERY, Embryol., 1942, p.127). L'on sait que la forme est une résultante biologique, et dans la mesure où la morphologie tend à devenir expérimentale et dynamique, beaucoup la considèrent comme une physiologie de la forme (DELAY, Psychol. méd., 1953, p.152):

1. De la morphologie statique, qui étudie les formes au repos (typologie générale, physiognomonie, chirologie), à la morphologie dynamique qui s'attache à leurs mouvements, la continuité s'établit de plus en plus. MOUNIER, Traité caract., 1946, p.207.

b) LINGUISTIQUE

[P. oppos. à syntaxe]
GRAMM. TRADITIONNELLE. Etude des différentes catégories de mots et des formes qu'ils présentent dans une langue (flexion et dérivation). En un sens très large, étude de tous les procédés de formation qui mettent en jeu des morphèmes et principalement des suffixes (morphologie suffixale); en un sens plus restreint, étude des éléments fléchis ou désinences (morphologie flexionnelle) (MAR. Lex. 1933, p.122):

2. ... les limites entre la lexicologie et la morphologie, et surtout entre la syntaxe et la morphologie sont très flottantes (...). En réalité, si la morphologie est l'étude des marques, il y a une morphologie du lexique et une morphologie de la syntaxe, et, la phonologie mise à part, seules les deux divisions: lexique et syntaxe restent en présence. PERROT, Ling., 1953, p.121.
LING. MOD. Description de la structure interne des mots et étude des règles qui régissent cette structure (d'apr. Lang. 1973).
[P. oppos. à lexique et phonologie] Description des règles régissant la structure interne des mots et de celles qui régissent la combinaison des syntagmes en phrases. Synon. morphosyntaxe (d'apr. Lang. 1973). c) SOCIOL. Morphologie sociale. Etude des formes matérielles, c'est-à-dire des structures apparentes des faits sociaux, principalement de leurs conditions géographiques et démographiques (d'apr. FOULQ. Sc. soc. 1978):

3. L'étude des sociétés urbaines, et jusque dans ses aspects les plus incontestablement sociologiques (c'est-à-dire relevant de la morphologie sociale), doit être envisagée en elle-même et pour elle-même et non par rapport aux préoccupations générales de la sociologie...

Traité sociol., 1967, p.294.
d) SC. DE LA TERRE. Synon. de géomorphologie. Initiation aux diverses techniques; stages de climatologie, glaciologie et de morphologie littorale (Colloque géogr. appl., 1962, p.137). Morphologie climatique (Encyclop. Sc. Techn. t.6 1971, p.342). Partie de la géomorphologie qui s'intéresse aux formes d'érosion (on dit aussi climatomorphologie) (d'apr. GEORGE 1970, s.v. géomorphologie). Morphologie structurale (Encyclop. Sc. Techn. t.6 1971, p.342). Partie de la géomorphologie qui s'intéresse aux formes du relief liées à la structure de l'écorce terrestre (d'apr. Encyclop. Sc. Techn., t.6 1971, p.342).

B. P. méton. Objet de cette étude, forme, structure. Le manuscrit de

Cassel offre plus de 500 exemples [du] schéma [de la courante] et permet d'en étudier la morphologie (ECORCHEVILLE, Suite orch., 1906, p.61). Cette civilisation parle le grec; entendons par là un attique d'une morphologie simplifiée, mais d'un vocabulaire enrichi par les apports dialectaux et ceux de la langue parlée (L'Hist. et ses méth., 1961, p.503). Les changements qui affectent la morphologie et la phonologie d'une langue ne peuvent (...) pas être interprétés comme les conséquences d'une action directe du cadre social sur la langue (...). De tels changements ne peuvent résulter que de causes elles-mêmes linguistiques (Traité sociol., 1968, p.272). V. origine II B ex. de Verne:

4. Les méthodes traditionnelles d'étude de la morphologie des cristaux sont encore largement utilisées de nos jours, mais surtout pour identifier les substances cristallines les plus variées grâce aux mesures des angles que font entre elles les faces d'un cristal et qui sont caractéristiques de chaque espèce. Encyclop. Sc. Techn. t.4 1970, p.87.

En partic. Forme générale du corps humain. Les sécrétions internes des glandes génitales modifient la morphologie (APERT ds Nouv. Traité Méd. fasc. 8 1925, p.373). Les morphologies les plus variées permettent la pratique du football (J. MERCIER, Football, 1966, p.78).

Prononc. et Orth.: []. Att. ds Ac. 1935. Etymol. et Hist. 1. 1822 science qui étudie la forme et la structure des organismesÈ (BLAINVILLE, De l'Organisation des animaux... Tome premier, contenant la morphologie et l'aistésologie [titre]); 2. 1868 gramm. (LITTRE); 3. 1935 géol. (Ac.). Formé des élém. morpho-* et -logie*, terme créé d'abord en all. par Goethe dans un sens gén., v. Blockhaus Enzykl., s.v. Morphologie. Fréq. abs. littér.: 62. Bbg. BRONDAL (V.). Déf. de la morphologie. In: [Mél. Ginneken (J. van)]. Paris, 1937, pp.43-50. GOUGENHEIM (G.). Morphologie et fonction gramm. Journal de Psychol. 1959, t.56, pp.417-426. MANCZAK (W.). Phonétique et morphologie hist. du fr. Warszawa, 1962, 203 p. MARTINET (A.). Qu'est-ce que la morphologie? In: [Mél. Frei (H.)]. Cah. F. Sauss. 1969, no26, pp.85-90. SECHEHAYE (A.). Considérations sur la morphologie. Cah. F. Sauss. 1944, t.4, pp.53-64.
[...]

Annexe C

Les logiciels développés

Deux logiciels ont été développés pour cette thèse. Aucun composant ou autre plate-forme de développement “Agents” n’ont été utilisés. Il a fallu créer une plate-forme de développement pour les systèmes auto-adaptatifs pour valider les hypothèses posées. Voici les deux logiciels en question :

- L’Ontological Transcription Core (OTC).
- Le morphoviewer.

C.1 Le noyau de transcription ontologique

C.1.1 Présentation

OTC (Ontological Transcription Core) est un logiciel basé sur un système multi-agent massif connecté en temps-réel à un système matériel ou logiciel afin de les manager. Ce système est une plate-forme de développement et une couche d’intégration permettant une autonomie de décision justifiée.

C.1.2 Description

Voici le fonctionnement général de l’OTC (figure C.1) :

- Domaine : le système peut gérer un certain nombre de domaines présents dans le monde réel. Il est capable de manager, de superviser ou de contrôler.
- Connaissance : une fois le domaine choisi, une ontologie exprimera ses caractéristiques.
- Transcription : l’ontologie est automatiquement chargée avec ses structures dans l’OTC sans programmation supplémentaire, générant une organisation d’agents.
- Objectifs : nous définissons les objectifs en fonction des problèmes d’une situation ainsi que les actions que nous devons exécuter dans certains contextes. Nous pouvons donner ces objectifs et les modifier en temps-réel.
- Exécution : le système est une boucle infinie entre :
  - le flux d’entrée,

  - la génération d'une représentation d'une situation courante avec des propositions de décisions, sélectionnées parmis les évènements passés, puis chargées dans la mémoire du système,
  - la génération d'un monitoring sophistiqué,
  - et les actions de l'utilisateur.
- Monitoring : le monitoring est un monde virtuel permettant le chargement de connaissance dans le système pour agir sur les paramètres et la visualisation de la représentation interactive courante.

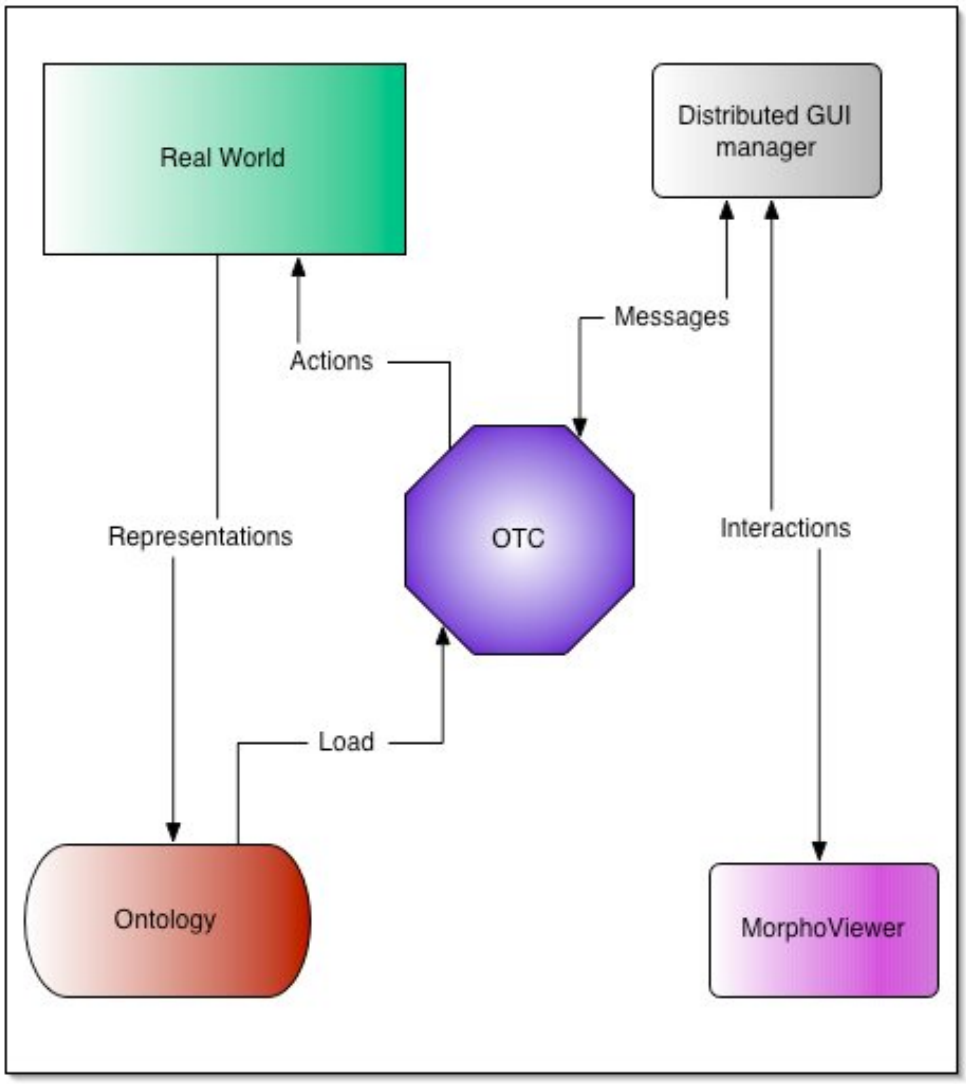

FIG. C.1 – Schéma global de l'Ontological Transcription Core.

C.2 Morphoviewer

C.2.1 Présentation

Le morphoviewer est un monde virtuel abstrait permettant de :
- charger de la connaissance dans le système pour agir sur les paramètres,
- de visionner la représentation interactive courante.

Le rôle du morphoviewer est la production et la mise en évidence des émergences présentes dans le système.

C.2.2 Description

Voici le fonctionnement général du morphoviewer (figure C.2), un monitoring avec deux interfaces de mise à jour :
- une mise à jour concernant l'ontologie,
- une mise à jour concernant le point focal.

Le point focal est composé de sphères. Les sphères sont spécifiques à la conception du noyau et peuvent être reliées symétriquement. L'importance des sphères suit un ordre définit formant l'ordre émergent de la représentation en constante modification selon les entrées.

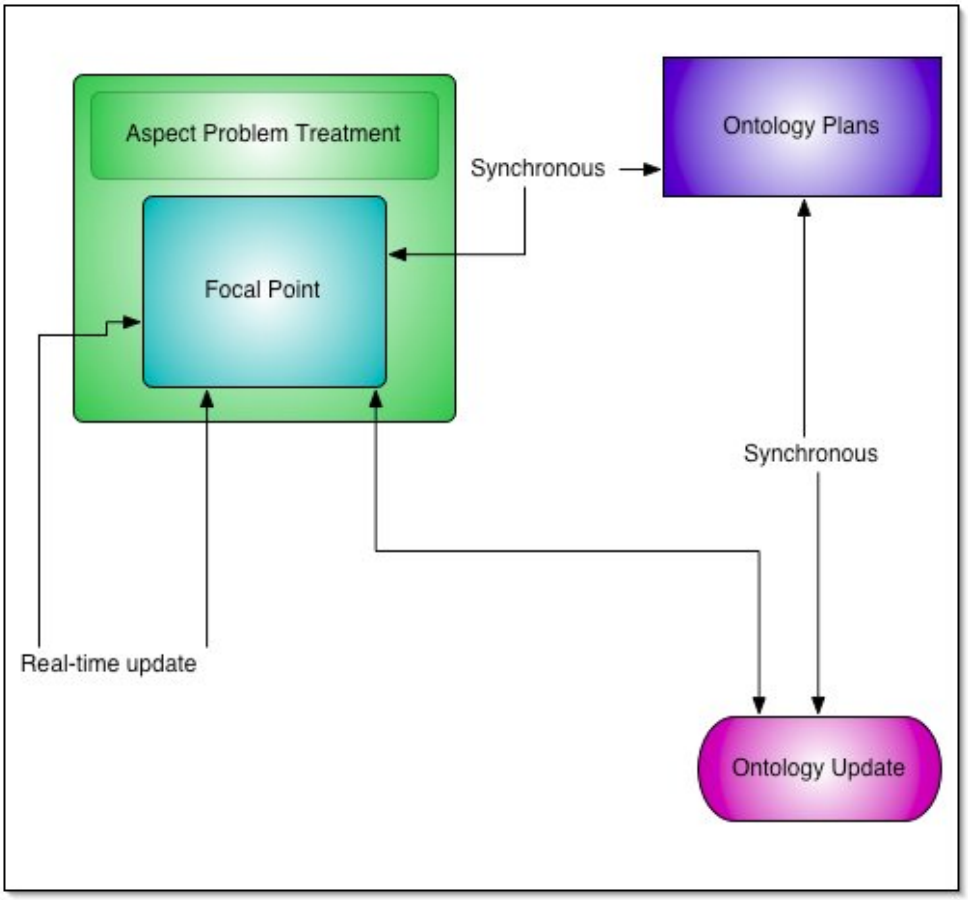

FIG. C.2 – Fonctionnement général Morphoviewer.

Annexe D

Les ontologies utilisées

Listing of : class

Add new

id_class	class_name	Edit
1	Capacities	Edit - Delete
2	Object	Edit - Delete
3	Verb	Edit - Delete
4	Colour	Edit - Delete
5	State	Edit - Delete
6	People	Edit - Delete
7	Simple_Emotions	Edit - Delete
8	Mixed_Emotions	Edit - Delete
9	Space	Edit - Delete
10	Experience	Edit - Delete

FIG. D.1 – La classification de la connaissance présente dans l'ontologie.

Listing of : agents Add new

View Details	id_agent	agent_name	Type	Class	Edit
View Details	1	Ears	role	Capacities	Edit - Delete
View Details	2	Backward	role	Capacities	Edit - Delete
View Details	3	Left	role	Capacities	Edit - Delete
View Details	4	Right	role	Capacities	Edit - Delete
View Details	5	Stop	role	Capacities	Edit - Delete
View Details	6	Stand	role	Capacities	Edit - Delete
View Details	7	Sit	role	Capacities	Edit - Delete
View Details	8	Tailrl	role	Capacities	Edit - Delete
View Details	9	Earl	role	Capacities	Edit - Delete
View Details	10	Earr	role	Capacities	Edit - Delete
View Details	11	Tailr	role	Capacities	Edit - Delete
View Details	12	Taild	role	Capacities	Edit - Delete
View Details	13	Tailup	role	Capacities	Edit - Delete
View Details	14	Bear	role	Capacities	Edit - Delete
View Details	15	Mouth	role	Capacities	Edit - Delete
View Details	16	Excited	role	Capacities	Edit - Delete
View Details	17	Dance1	role	Capacities	Edit - Delete
View Details	18	Dance2	role	Capacities	Edit - Delete
View Details	19	Dande3	role	Capacities	Edit - Delete
View Details	20	Dance4	role	Capacities	Edit - Delete
View Details	21	Dance5	role	Capacities	Edit - Delete
View Details	22	Head	role	Capacities	Edit - Delete
View Details	23	Turnbody	role	Capacities	Edit - Delete
View Details	24	Bow	role	Capacities	Edit - Delete
View Details	25	Insect	role	Capacities	Edit - Delete
View Details	26	Mouth2	role	Capacities	Edit - Delete
View Details	27	Kick	role	Capacities	Edit - Delete
View Details	28	Aibone	role	Capacities	Edit - Delete
View Details	29	Pass	role	Capacities	Edit - Delete
View Details	30	Throw	role	Capacities	Edit - Delete

1 | 2 | 3 | 4 >>

FIG. D.2 – Première page de la connaissance présente dans l'ontologie.

Listing of : agents Add new

View Details	id_agent	agent_name	Type	Class	Edit
View Details	31	Lick	role	Capacities	Edit - Delete
View Details	32	Kiss	role	Capacities	Edit - Delete
View Details	33	Pick	role	Capacities	Edit - Delete
View Details	34	Greet	role	Capacities	Edit - Delete
View Details	35	Happy	role	Capacities	Edit - Delete
View Details	36	Sad	role	Capacities	Edit - Delete
View Details	37	Sleepy	role	Capacities	Edit - Delete
View Details	38	Sleepwithaibone	role	Capacities	Edit - Delete
View Details	39	Led	role	Capacities	Edit - Delete
View Details	40	Led1	role	Capacities	Edit - Delete
View Details	41	Led2	role	Capacities	Edit - Delete
View Details	42	Led3	role	Capacities	Edit - Delete
View Details	43	Led4	role	Capacities	Edit - Delete
View Details	44	Led5	role	Capacities	Edit - Delete
View Details	45	Led6	role	Capacities	Edit - Delete
View Details	46	Led7	role	Capacities	Edit - Delete
View Details	47	Led8	role	Capacities	Edit - Delete
View Details	48	Led9	role	Capacities	Edit - Delete
View Details	49	Led10	role	Capacities	Edit - Delete
View Details	50	Led11	role	Capacities	Edit - Delete
View Details	51	Throwaibone	role	Capacities	Edit - Delete
View Details	52	Thirlaibone	role	Capacities	Edit - Delete
View Details	53	Joyaibone	role	Capacities	Edit - Delete
View Details	54	Danceaibone	role	Capacities	Edit - Delete
View Details	55	Walkball	role	Capacities	Edit - Delete
View Details	56	Playball	role	Capacities	Edit - Delete
View Details	57	Kickball	role	Capacities	Edit - Delete
View Details	58	Forwardball	role	Capacities	Edit - Delete
View Details	59	Sadball	role	Capacities	Edit - Delete
View Details	60	Holdaibone	role	Capacities	Edit - Delete

<< 1 | 2 | 3 | 4 >>

FIG. D.3 – Deuxième page de la connaissance présente dans l'ontologie.

Listing of : agents Add new

View Details	id_agent	agent_name	Type	Class	Edit
View Details	61	Aibonenoise	role	Capacities	Edit - Delete
View Details	62	Putaibone	role	Capacities	Edit - Delete
View Details	63	Slap	role	Capacities	Edit - Delete
View Details	64	Bug	role	Capacities	Edit - Delete
View Details	65	Weddingmarch	role	Capacities	Edit - Delete
View Details	66	Amazingrace	role	Capacities	Edit - Delete
View Details	67	Blackwhite	role	Capacities	Edit - Delete
View Details	68	Cuddlytoy	role	Object	Edit - Delete
View Details	69	Bone	role	Object	Edit - Delete
View Details	70	Ball	role	Object	Edit - Delete
View Details	71	Keyboard	role	Object	Edit - Delete
View Details	72	Book	role	Object	Edit - Delete
View Details	73	Generator	role	Object	Edit - Delete
View Details	74	Cushion	role	Object	Edit - Delete
View Details	75	Cover	role	Object	Edit - Delete
View Details	76	Cd	role	Object	Edit - Delete
View Details	77	Cassette	role	Object	Edit - Delete
View Details	78	Teddybear	role	Object	Edit - Delete
View Details	79	Play	role	Verb	Edit - Delete
View Details	80	Work	role	Verb	Edit - Delete
View Details	81	Dance	role	Verb	Edit - Delete
View Details	82	Sleep	role	Verb	Edit - Delete
View Details	83	Office	role	Verb	Edit - Delete
View Details	84	Pink	role	Colour	Edit - Delete
View Details	85	Fatigue	role	State	Edit - Delete
View Details	86	Alain	role	People	Edit - Delete
View Details	87	Jean-charles	role	People	Edit - Delete
View Details	88	Lionel	role	People	Edit - Delete
View Details	89	Mickael	role	People	Edit - Delete
View Details	90	Pleasure	role	Simple_Emotions	Edit - Delete

<< 1 | 2 | 3 | 4 >>

FIG. D.4 – Troisième page de la connaissance présente dans l'ontologie.

Listing of : agents Add new

View Details	id_agent	agent_name	Type	Class	Edit
View Details	91	Dissatisfaction	role	Simple_Emotions	Edit - Delete
View Details	92	Joy	role	Simple_Emotions	Edit - Delete
View Details	93	Pain	role	Simple_Emotions	Edit - Delete
View Details	94	Tenderness	role	Simple_Emotions	Edit - Delete
View Details	95	Sympathy	role	Simple_Emotions	Edit - Delete
View Details	96	Love	role	Mixed_Emotions	Edit - Delete
View Details	97	Frontof	role	Space	Edit - Delete
View Details	98	Behind	role	Space	Edit - Delete
View Details	99	Where	role	Space	Edit - Delete
View Details	100	Side	role	Space	Edit - Delete
View Details	101	Far	role	Space	Edit - Delete
View Details	102	Near	role	Space	Edit - Delete
View Details	103	pleasure	goal	Simple_Emotions	Edit - Delete
View Details	104	dissatisfaction	goal	Simple_Emotions	Edit - Delete
View Details	105	PlayBall	classification	Experience	Edit - Delete

<< 1 | 2 | 3 | 4

FIG. D.5 – Quatrième page de la connaissance présente dans l'ontologie.

Listing of : experiences Add new

id_experience	experience_name	Edit
1	Play	Edit - Delete
2	PlayBall	Edit - Delete
3	Play_with_peoples	Edit - Delete
4	Work on computer	Edit - Delete

FIG. D.6 – Les expériences de l'ontologie.

Listing of : acointances Add new

id_acointance	Experience	agent1	agent2	value	Edit
1	Play	Alain	Ears	1	Edit - Delete
2	Play	Alain	Tailrl	1	Edit - Delete
3	Play	Alain	Ball	1	Edit - Delete
4	Play	Alain	Sleep	-1	Edit - Delete
5	Play	Alain	Work	-1	Edit - Delete
6	Play	Ball	Frontof	1	Edit - Delete
7	Play	Ball	Jean-charles	1	Edit - Delete
8	Play	Ball	Ears	1	Edit - Delete
9	Play	Ball	Tailrl	1	Edit - Delete
10	Play	Ball	Led2	1	Edit - Delete
11	Play	Ball	Mouth2	1	Edit - Delete
12	Play	Ball	Sleep	-1	Edit - Delete
13	Play	Sleep	Cushion	1	Edit - Delete
14	Play	Sleep	Cover	1	Edit - Delete
15	Play	Sleep	Work	-1	Edit - Delete
16	Play	Sleep	Play	-1	Edit - Delete
17	Play	Work	Keyboard	1	Edit - Delete
18	Play	Work	Book	1	Edit - Delete
19	Play	Work	Generator	1	Edit - Delete
20	Play	Play	Ears	1	Edit - Delete
21	Play	Play	Tailrl	1	Edit - Delete
22	Play	Play	Ball	1	Edit - Delete
23	Play	Play	Sleep	-1	Edit - Delete
24	Play	Play	Work	-1	Edit - Delete
59	Play	pleasure	Play	1	Edit - Delete
60	Play	pleasure	Cushion	1	Edit - Delete
61	Play	pleasure	Pink	1	Edit - Delete
62	Play	pleasure	Cover	1	Edit - Delete
63	Play	pleasure	Bone	1	Edit - Delete
25	PlayBall	Pink	Ball	1	Edit - Delete

1 | 2 | 3 >>

FIG. D.7 – Première page du listing des accointances.

Listing of : acointances Add new

id_acointance	Experience	agent1	agent2	value	Edit
26	PlayBall	Pink	Bone	1	Edit - Delete
27	PlayBall	Pink	Keyboard	-1	Edit - Delete
28	PlayBall	Pink	Alain	1	Edit - Delete
29	PlayBall	Ball	Frontof	1	Edit - Delete
30	PlayBall	Ball	Jean-charles	1	Edit - Delete
31	PlayBall	Ball	Ears	1	Edit - Delete
32	PlayBall	Ball	Tailrl	1	Edit - Delete
33	PlayBall	Ball	Led2	1	Edit - Delete
34	PlayBall	Ball	Mouth2	1	Edit - Delete
35	PlayBall	Ball	Sleep	-1	Edit - Delete
58	PlayBall	pleasure	Ball	1	Edit - Delete
68	PlayBall	PlayBall	Ball	1	Edit - Delete
36	Play_with_peoples	Sympathy	Love	1	Edit - Delete
37	Play_with_peoples	Sympathy	Alain	1	Edit - Delete
38	Play_with_peoples	Sympathy	Jean-charles	1	Edit - Delete
39	Play_with_peoples	Sympathy	Lionel	0.77	Edit - Delete
40	Play_with_peoples	Sympathy	Mickael	1	Edit - Delete
41	Play_with_peoples	Love	Alain	1	Edit - Delete
42	Play_with_peoples	Love	Jean-charles	1	Edit - Delete
43	Play_with_peoples	Frontof	Ball	1	Edit - Delete
44	Play_with_peoples	Frontof	Play	1	Edit - Delete
45	Play_with_peoples	Frontof	Work	-1	Edit - Delete
46	Play_with_peoples	Bone	Sit	1	Edit - Delete
47	Play_with_peoples	Bone	Tailrl	1	Edit - Delete
48	Play_with_peoples	Bone	Led9	1	Edit - Delete
49	Play_with_peoples	Bone	Kiss	1	Edit - Delete
50	Work on computer	Keyboard	Work	1	Edit - Delete
51	Work on computer	Keyboard	Bear	1	Edit - Delete
52	Work on computer	Keyboard	Play	0.85	Edit - Delete
53	Work on computer	Keyboard	Mickael	1	Edit - Delete

<< 1 | 2 | 3 >>

FIG. D.8 – Deuxième page du listing des accointances.

Listing of : acointances Add new

id_acointance	Experience	agent1	agent2	value	Edit
54	Work on computer	Keyboard	Kick	1	Edit - Delete
55	Work on computer	Book	Bear	1	Edit - Delete
56	Work on computer	Book	Work	1	Edit - Delete
57	Work on computer	Book	Alain	1	Edit - Delete
64	Work on computer	dissatisfaction	Work	1	Edit - Delete
65	Work on computer	dissatisfaction	Generator	1	Edit - Delete
66	Work on computer	dissatisfaction	Alain	1	Edit - Delete
67	Work on computer	dissatisfaction	Fatigue	1	Edit - Delete

<< 1 | 2 | 3

FIG. D.9 – Troisième page du listing des accointances.

Annexe E

Le Oz/Mozart

E.1 Présentation

Le MOZART PROGRAMMING SYSTEM (www.mozart-oz.org) est un environnement de développement pour applications distribuées et autonomes. Ce système est le résultat de plusieurs recherches effectuées dans le design des langages, les moteurs d'inférence basés sur les contraintes, les algorithmes distribués et les interfaces homme-machine.

Le Mozart est basé sur le langage OZ qui supporte différents paradigmes. Ce système a été développé par un consortium nordique incluant l'UCL Université Catholique de Louvain , l'UNIVERSITÄT DES SAARLANDES , et la SWEDISH INSTITUTE OF COMPUTER SCIENCE (SICS) .

E.2 Caractéristiques

Le OZ est un langage de très haut niveau permettant le prototypage rapide d'une application ayant pour finalité d'être la plus autonome possible. Cette finalité est atteinte en général grâce à l'utilisation de moteurs d'inférence et de contraintes, ce que permet le OZ/MOZART .

Les principales caractéristiques du MOZART PROGRAMMING SYSTEM sont donc :

- la programmation distribuée,
- la programmation logique,
- la programmation par contraintes,
- la programmation orientée objet,
- la compatibilité multi-plateformes,
- un environnement de développement.

E.3 L'environnement de développement

le MOZART PROGRAMMING SYSTEM met à disposition un bon nombre d'outils à l'utilisateur. Parmi ces outils, on peut remarquer l'utilisation de EMACS ou XEMACS . Pour ce faire, un module LISP spécifique a été développé.

La sélection de l'éditeur de texte se fait grâce à la variable d'environnement *OZEMACS*. De cette manière, lorsque l'utilisateur lance l'environnement de développement, c'est EMACS qui se charge en mémoire avec le module OZ approprié.

Le répertoire binaire du MOZART PROGRAMMING SYSTEM mettant à disposition tous ses outils prend cette forme :

```
$>ll
total 76
drwxr-xr-x    2 mc       users        4096 déc 21  2001 .
drwxr-xr-x    8 mc       users        4096 mai 15 14:49 . .
lrwxrwxrwx    1 mc       users           2 mar 11 14:26 oldpickle2text -> oz
-r-xr-xr-x    1 mc       users        2710 nov 21  2001 oz
-r-xr-xr-x    1 mc       users       13162 déc 21  2001 ozc
-r-xr-xr-x    1 mc       users        3168 déc 21  2001 ozd
lrwxrwxrwx    1 mc       users           2 mar 11 14:26 ozengine -> oz
-r-xr-xr-x    1 mc       users       23645 déc 21  2001 ozl
-r-xr-xr-x    1 mc       users        1516 déc 21  2001 ozplatform
lrwxrwxrwx    1 mc       users           2 mar 11 14:26 oztool -> oz
-r--r--r--    1 mc       users        5632 nov 21  2001 ozwrapper.bin
-r--r--r--    1 mc       users        6656 nov 21  2001 ozwrapperw.bin
lrwxrwxrwx    1 mc       users           2 mar 11 14:26 pickle2text -> oz
lrwxrwxrwx    1 mc       users           2 mar 11 14:26 text2pickle -> oz
$>
```

Voici les rôles des principaux outils :
- **oz** : l'interpréteur du langage OZ ,
- **ozc** : le compilateur permettant de générer du bytecode,
- **ozd** : le debugger du MOZART PROGRAMMING SYSTEM ,
- **oztool** : qui est en définitif **oz** et qui permet de créer des librairies dynamiques en C ou en C++.

La figure E.1 montre l'environnement de développement de base.

E.4 Le Oz Kernel Language

Cette partie du langage représente l'essentiel du langage OZ . C'est sur cette base que le langage a été construit. Grâce à ce KERNEL LANGAGE, d'autres paradigmes ont pu être intégré à OZ . Tous les paradigmes supportés par le langage OZ

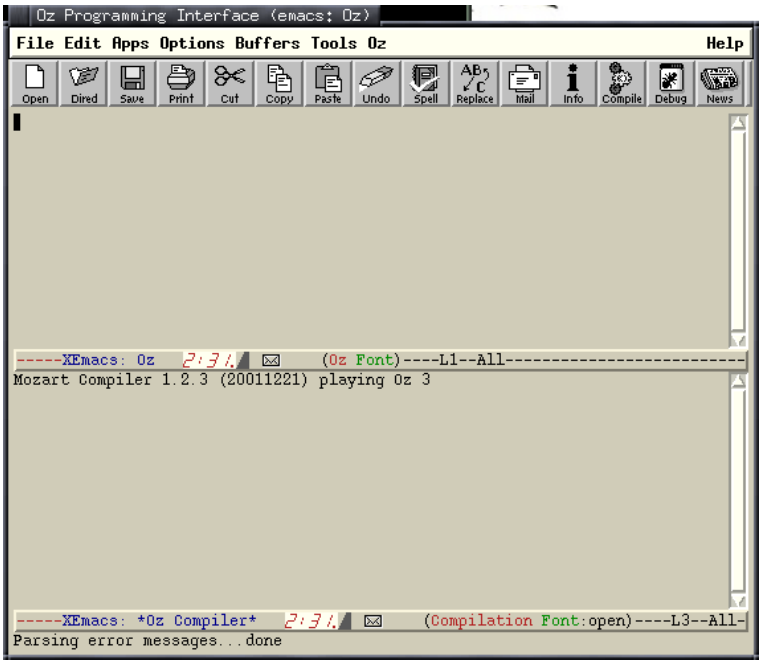

FIG. E.1 – Environnement de développement du MOZART PROGRAMMING SYSTEM

est en fait traduit en “Kernel Language” avant d’être byte-compilé.

Le OZ KERNEL LANGAGE :

```
<Statement> ::= <Statement1> <Statement2>
| X = f(l1:Y1 ... ln:Yn)
| X = <number>
| X = <atom>
| X = <boolean>
| {NewName X}
| X = Y
| local X1 ... Xn in S1 end
| proc {X Y1 ... Yn} S1 end
| {X Y1 ... Yn}
| {NewCell Y X}
| {Exchange X Y Z}
| {Access X Y}
| if B then S1 else S2 end
| thread S1 end
| try S1 catch X then S2 end
| raise X end
```

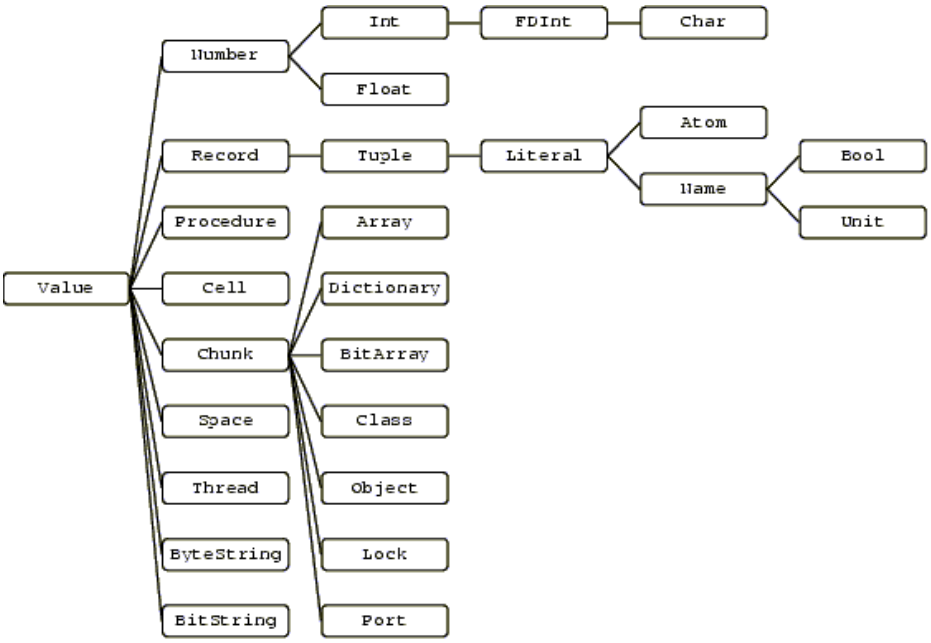

FIG. E.2 – Types primaires du langage OZ

E.5 Les types

La déclaration de type n'existe pas en OZ/MOZART . L'abstraction est telle que l'utilisation des variables se fait de la même manière que dans un langage de script du type PERL, PYTHON ou RUBY.

Il est très facile de limiter l'accès aux variables. Un code de ce type :

```
local X Y Z in S end
```

permet d'introduire trois variables X, Y et Z qui seront visibles seulement dans le code S. Une variable commence obligatoirement par une majuscule, de plus, lors de la déclaration, aucune valeur n'est associée à ces variables, elles peuvent prendre un type alpha-numérique.

Alors qu'un code de ce type :

```
declare X Y Z in S
```

permet l'accès aux variables X, Y et Z depuis tout le script présent dans le fichier. La figure E.2 présente les types primaires du langage OZ .

E.5.1 Initialisation

Il y a différentes manières d'initialiser une variable. En scripting, le symbole "=" est utilisé, nous avons donc :

```
X = 1
```

qui permet d'initialiser la variable X. Mais attention, après cette initialisation, X ne pourra pas prendre une autre valeur que 1. Cette variable est considérée comme une constante. Il est donc impossible de la placée dans une boucle pour modifier sa valeur à chaque itération.

E.6 Les différents paradigmes supportés

E.6.1 Type de développement

Le paradigme de base en OZ est le fonctionnel comme le SCHEME ou le LISP. Voici comment appeler une procédure *P*, *R* étant la variable récupérant la valeur retournée par *P* :

```
{P X1 ... Xn R}
```

P peut être utilisé aussi comme un appel de fonction,

```
{P X1 ... Xn}
```

de cette manière, le résultat est traité directement et n'est pas stocké.
Il existe plusieurs méthodes syntaxiques pour écrire une fonction en OZ . La fonction

```
fun {F X1 ... Xn} S E end
```

ou *S* représente le corps de la fonction et *E* la valeur de retour, est équivalente à cette procédure :

```
proc {F X1 ... Xn R} S R=E end
```

E.6.2 Les spécificités du langage

Les structures de contrôles sont basiques, par exemple, le bloc *if* s'écrit comme ceci :

```
if B then S1 else S2 end
```

La spécificité majeure est dans le traitement du *and* et du *or*, qui est tout à fait différent du C++ ou du JAVA. En effet, le langage OZ utilise *andthen* et *orthen*.*andthen* est écrit de la sorte :

```
fun {AndThen BP1 BP2}
      if {BP1} then {BP2}
      else false end
end
```

ou

```
proc {AndThen BP1 BP2 B}
   B = if {BP1} then {BP2}
       else false end
end
```

Pour le *orthen*, il suffit d'ajouter un test sur le deuxième paramètre pour savoir si l'expression est vraie.

Au niveau des boucles, il n'existe que le for, qui s'écrit tout simplement :

```
for I in 1..B do
   S
end
```

ou I représente la variable incrémentée, B la limite, et S le corps du programme.

E.6.3 Scripting

Le MOZART PROGRAMMING SYSTEM donne la possibilité d'écrire des scripts, ce qui accélère le développement si le but est de créer un petit programme de test qui n'aura pas beaucoup de lignes de codes, ni de suivi de qualité. Pour un gros projet, il est bien sûr déconseillé d'utiliser le scripting.

Au niveau de la syntaxe, il y a plusieurs moyens de procéder. Pour déclarer des variables qui ne seront pas visibles dans le script, il faut procéder de la sorte :

```
local G H I in
   S
end
```

ou G, H et I sont des variables visibles seulement dans le corps S.
Il est possible aussi de déclarer une fonction ou une procédure comme une variable :

```
local
   G = proc{$}
       C
       end
   H = fun{$}
       D
```

```
       E
       end
   I
     in
   S
end
```

Ici, G est une procédure avec un corps C et H est une fonction de corps D et E comme valeur de retour.

Pour qu'une variable soit visible dans tout le script, il suffit de procéder comme ceci :

```
define L in
 local
    G = proc{$}
        C
        end
    H = fun{$}
        D
        E
        end
    I
      in
    S
 end
 local T U in
    K
 end
end
```

L sera visible même dans le local de corps K.

E.6.4 Les modules

Les modules sont un système de package permettant de stocker un ensemble de procédures ou de valeurs[1] qui ont été écris précédemment.
Un module possède un nombre de procédures privées qui ne sont pas visibles hors du module, et un nombre de procédures d'interfaces qui permettent de fournir les services externes du module. Il existe une syntaxe spécifique pour écrire un module. Le fichier du programme commence par *functor*, celui-ci spécifie les composants dans module. Le MOZART PROGRAMMING SYSTEM possède un MODULE MANAGER qui transforme un *functor* en un module utilisable.

[1] ces valeurs peuvent être des classes ou des objets.

Voyons maintenant comment les modules procèdent pour atteindre l'ensemble de leurs procédures.En OZ , il est très simple d'écrire un programme permettant de stocker plusieurs procédures dans une liste :

```
declare List in
local
   proc {Append ... }  ... end
   proc {MergeSort ...} ... end
   proc {Sort ... } ... {MergeSort ...} ... end
   proc {Member ...} ... end
 in
   List = 'export'(append: Append
                   sort: Sort
                   member: Member
                   ... )
end
```

Pour accéder à la procédure *Append* hors du module *List*, il suffit d'utiliser *List* comme un Record, c'est à dire *List* : $List.append$. Le problème avec ce programme est que toutes ces procédures sont inutilisables hors du module. Pour qu'elles le soient, il faut écrire le module de la sorte (avec le mot clé $functor$) :

```
functor
export
  append:Append
  sort:Sort
  member:Member
  ...
define
   proc {Append ... }  ... end
   proc {MergeSort ...} ... end
   proc {Sort ... } ... {MergeSort ...} ... end
   proc {Member ...} ... end
end
```

Le mot clé *export* permet d'atteindre les procédures désirées. Par la suite, la compilateur *ozc* sera utilisé pour créer le module. L'extension $.ozf$ représente les modules, ces derniers s'obtiennent avec l'option $-c$ du compilateur, pour notre programme (admettons que le fichier s'appelle list.oz) :

```
ozc -c List.oz
```

Ce qui générera list.ozf, qui pourra être importé dans un autre module.
Soit list.ozf se trouvant dans /home/XXX/, ce module pourra être chargé et utilisé de cette manière :

```
functor
import
   MList at '/home/XXX/list.ozf'
define
   {MList.append ...}
   .
   .
   .
end
```

import permet de charger les modules demandés par le nouveau module ou le script. Une fois que tous les modules sont chargés, le code entre *define* et *end* est exécuté.

E.7 Le processing

OZ/MOZART à une méthode bien à lui pour créer de nouveau processus. Cette méthode donne aux développeurs une souplesse de programmation qui est très agréable, et surtout, très efficace. En effet, la création de thread est intégrée directement à la syntaxe du langage. En OZ , un nouveau thread s'écrit de la sorte :

```
thread S end
```

ou S est le code à exécuter.
Grâce au module Thread, qui est chargé automatiquement, il est très facile de manipuler les threads et de les faire communiquer entre eux. Voici un petit exemple d'utilisation de thread :

```
declare X0 X1 X2 X3 in
thread
   local Y0 Y1 Y2 Y3 in
      {Browse [Y0 Y1 Y2 Y3]}
      Y0 = X0+1
      Y1 = X1+Y0
      Y2 = X2+Y1
      Y3 = X3+Y2
      {Browse completed}
   end
end
{Browse [X0 X1 X2 X3]}
```

Browse est un outil d'affichage, et sert ici à constater le résultat. Le programme tel qu'il est affichera des valeurs vide. Mais si ces valeurs sont ajouté (avant le dernier Browse) : $X0 = 0$, $X1 = 1$, $X2 = 2$ et $X3 = 3$, alors le code du thread s'exécutera au fur et à mesure des initialisations.

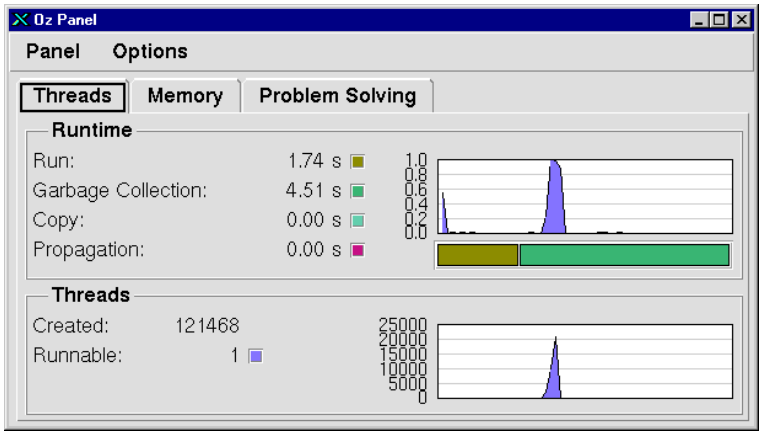

FIG. E.3 – Le OZ PANEL

Le MOZART PROGRAMMING SYSTEM possède un monitoring pour suivre l'évolution des threads présents dans un programme, celui-ci s'appelle le OZ PANEL. Ce programme crée un nombre exponentiel de thread :

```
fun {Fib X}
   case X
   of 0 then 1
   [] 1 then 1
   else thread {Fib X-1} end  + {Fib X-2} end
end
```

La figure E.3 montre le OZ PANEL scrutant le programme {Fib 26 X}, *X* étant la variable de récupération.

E.8 La programmation orienté objet

E.8.1 Les classes

OZ/MOZART permet de programmer en objet. Une classe, en OZ , est un type *chunk* qui contient :

- la collection de méthodes dans une table de méthodes,
- une description des attributs que chaque instance de classe possédera,
- une description des $features$ qui sont des variables autre que les atomes ou les noms, et qui sont accessibles par le $featur-name$,
- une description de la manière dont les objets de la classe doivent se comporter.

Il est très facile d'écrire une classe en script OZ ,

```
declare Foo
```

```
local
   Attrs = [val]
   MethodTable = m(browse:MyBrowse init:Init inc:Inc)
   proc {Init ...}
       A
   end
   proc {Inc ...}
       B
   in
       C
   end
   proc {MyBrowse ...}
       D
   end
in
   Foo = {NewChunk c(methods:MethodTable attrs:Attrs)}
end
```

La classe *Foo* a bien été créée en respectant la description d'une classe en langage OZ . Cette classe possède trois méthodes, qui sont en réalité des procédures, possédant respectivement un corps *A*, *B* et *C*. Les attributs sont accessibles grâce à l'atome *val*, les méthodes sont en fait un Record de procédures.

E.8.2 Les objets

Pour la créations des objets, le système reste le même, ils peuvent être créés grâce au langage de base.En OZ un objet correspond :

- à la création d'un Record du nombre d'attribut présent dans la classe,
- à créer une nouvelle cellule à chaque élément du Record,
- à appeler la méthode d'initialisation de la classe,
- à retourner l'objet ou plutôt la procédure correspondante.

La création d'une classe en objet prend cette forme :

```
proc {NewObject Class InitialMethod ?Object}
   State Obj
in
   State = {MakeRecord s Class.attrs}
   {Record.forAll State proc {$ A} {NewCell _ A} end}
   proc {Obj M}
      {Class.methods.{Label M}}
   end
   {Obj InitialMethod}
   Object = Obj
end
```

La transformation de la classe *Foo* en objet prendrait donc cette forme :

```
declare F
{NewObject Foo init(...) F}
{F inc(..)}
{F browse(...)}
```

Une fois la classe transformée, il est possible d'accéder directement aux méthodes.

E.8.3 La syntaxe

Bien qu'il soit facile de programmer en objet avec le langage de base, OZ supporte une syntaxe bien plus souple et efficace pour créer des objets. De cette façon, une classe *X* s'écrit de cette manière,

```
class X ... end
```

les attributs, comme ceci,

```
attr A1 ... AN
```

et les méthodes, comme cela,

```
meth E() S end
```

avec *S* le corps de la méthode.
La classe *Foo* aurait donc pu prendre cette forme :

```
class Foo
   attr val
   meth init(...)
      A
   end
   meth inc(...)
      C
   end
   meth browse(...)
      D
   end
end
```

Il est tout à fait possible d'intégrer une classe dans un module, il suffit de ne pas oublier de l'exporter.

E.8.4 L'héritage

Le langage OZ supporte l'héritage et l'héritage multiple. Le mot clé qui est utilisé pour exprimer cela est $from$.Une classe A qui hérite d'une classe B s'écrit

```
class A from B ... end
```

OZ/MOZART ne supportera pas certains héritages comme par exemple A qui hérite de B et B qui hérite de A,

```
class A from B ... end
class B from A ... end
```

ou encore,

```
class A1 meth m(...) ... end end
class B1 meth m(...) ... end end
class B from B1 end
class A from A1 end
class C from A B end
```

Dans ce type de cas, la compilateur générera une erreur. L'héritage multiple est supporté, mais à une certaine condition, en effet, toutes les méthodes des classes présentes dans l'héritage doivent avoir un nom différent, de cette manière,

```
class A meth m(...) ... end end
class B meth m(...) ... end end
class C from A B end
```

ne sera pas du tout accepté par le compilateur.

E.9 La programmation logique

Le langage OZ donne aussi la possibilité de faire de la logique, beaucoup de problèmes en intelligence artificielle peuvent être résolus par de la programmation logique. A cet effet, OZ/MOZART fournit un espace de travail pour la programmation logique qui permet de stocker et de traiter toutes les variables OZ de n'importe quel type, que ce soit du integer, du cell, du Record ou du List, pour donner la possibilité d'effectuer des opération logiques sur celles-ci et d'utiliser les disjonctions ainsi que les conditions.

E.9.1 Les disjonctions

Dans toute disjonctive nous parlons de clause et de fait. En OZ une clause est une suite de fait appelé G suivie d'un corps appelé $S1$:

```
G then S1
```

La suite de fait prend cette forme :

```
X1 ... Xn in S0
```

ou les variables Xi existe bien et sont quantifiables.
Une disjonctive en OZ prend cette forme :

```
or
   G1 then S1
[] G2 then S2
   ...
[] GN then SN
end
```

Le mot clé *or* crée automatiquement un nouveau thread s'exécutant dans un espace père *SP*.

- N espaces sont créés de SP_1 à SP_n, avec N nouveaux threads pour exécuter G_1 à G_n,
- le processus père reste bloqué jusqu'à ce qu'un ou plusieurs de ces fils aient échoué,
- si tous les processus enfant échouent, le processus père génère une exception dans son espace,
- il n'y a que l'espace qui n'a pas échoué qui reste valide.

E.9.2 Les conditions

Une condition logique en OZ/MOZART prend cette forme :

```
cond X1 ... XN in S0 then S1 else S2 end
```

ou X_i sont de nouvelles variables et S_i sont des corps.X1 ... XN in S0 then S1 est la clause de la condition, et S2 est l'alternative.
Le mot clé *cond* permet le création d'un nouveau thread père *SP*,

- le thread père est bloqué,
- un espace SP_1 est créé avec un simple processus exécutant cond X1 ... XN in S0,
- l'exécution du processus père reste bloqué jusqu'à ce que SP_1 est empilé ou désempilé,
- si SP_1 désempile, le processus père continue avec S2,
- si SP_1 empile, l'espace SP_1 est mergé avec l'espace *SP* (le processus père), et le processus père continue avec S1.

E.9.3 Les programmes non déterministe

Le langage OZ permet d'écrire des programmes non déterministe comme PROLOG. Ce type de développement est tout de même différent de la méthode employé

par le PROLOG.Alors que ce dernier possède une stratégie de recherche par défaut basé sur l'unification et le backtracking, OZ lui, laisse le développeur diviser sa stratégie de recherche avec différentes méthode.

Pour cela, le mot clé *dis* est utilisé, ce qui permet de présenter un choix.

```
proc {Append Xs Ys Zs}
   dis
      Xs = nil Ys = Zs then skip
   [] X Xr Zr in
      Xs = X|Xr Zs = X|Zr then
      {Append Xr Ys Zr}
   end
end
```

Ce qui correspond en PROLOG à :

```
append([], Ys, Ys).
append([X|Xr], Ys, [X|Zr]) :- append(Xr, Yr, Zr).
```

E.10 La programmation par contraintes

Le langage OZ permet aussi de faire de la programmation par contraintes. Celle-ci est gérée par différent modules. Essentiellement, OZ/MOZART donne la possibilité de résoudre des problèmes combinatoires qui sont traités avec plusieurs variables appartenant à un domaine fini d'entier positif.

Les deux techniques principales en programmation par contraintes sont :

- la propagation des contraintes, qui est un mécanisme d'inférence obtenue grâce aux propagateur qui accumule l'information dans un espace de contrainte.
- la distribution des contraintes, qui découpe le problème en plusieurs sous-problème pour appliquer ensuite la propagation des contraintes.

Ces deux méthodes donnent plus de chance au développeur de trouver une solution à un problème combinatoire.

Un exemple de problème combinatoire est le problème SEND MORE MONEY, qui consiste à trouver un chiffre à chaque lettre sachant que S et M sont différents de zéro et que l'équation

```
SEND + MORE = MONEY
```

soit satisfaite. L'unique solution de ce problème est **9567 + 1085 = 10652**

E.11 Les contraintes et les domaines finis

Un domaine fini est une suite d'entier positif, il se note $m\sharp n$ pour un domaine de ce type $[m,..,n]$. Une contrainte est une formule de prédicat logique, voici un exemple typique :
$X = 67,\ X \in D\sharp 9,\ X = Y,\ X^2 - Y^2 = Z^2,\ X + Y + Z < U,\ X + Y \neq 5.Z,\ X_1,...,X_9$ sont par paire distinct

Une contrainte de domaine prend cette forme $x \in D$, ou D est un domaine fini. Les contraintes de domaine peuvent être exprimées comme ceci : $x = n$, ce qui est équivalent à $x \in n\sharp n$.

Une contrainte basique prend cette forme : $x = n$, $x = y$ ou $x \in D$, ou D est un domaine fini. Un problème ayant un domaine fini possède un ensemble fini P de contraintes quantifiables ou P contient une contrainte de domaine pour toutes les variables utilisées par les contraintes dans P. La solution d'un problème en domaine fini est la variable qui satisfait toutes les contraintes dans P.

E.12 La propagation de contraintes

Cette propagation est tout simplement une règle d'inférence qui rétréci le domaine de variable, soit l'équation $X < Y$, soit le domaine de contrainte $X \in 23\sharp 100$ et $Y \in 1\sharp 33$. La propagation de contrainte pourra rétrécir le domaine à $X \in 23\sharp 32$ et $Y \in 24\sharp 33$.

E.13 Application en OZ

Pour résoudre un problème combinatoire en OZ , il faut utiliser le module **FD**, comme Finite Domain. Résolvons le problème SEND MORE MONEY. Il s'agit d'attribuer une variable pour chaque lettre et de leur associer un domaine de chiffres possibles.
La distribution des contraintes, consiste à une stratégie $first - fail$, les variables sont ordonnées par ordre alphabétique des lettres, la stratégie essaie la valeur possible pour chaque variable.

```
proc {Money Root}
   S E N D M O R Y
in
   Root = sol(s:S e:E n:N d:D m:M o:O r:R y:Y)
   Root ::: 0#9
   {FD.distinct Root}
   S \=: 0
   M \=: 0
```

```
                    1000*S + 100*E + 10*N + D
       +            1000*M + 100*O + 10*R + E
       =: 10000*M + 1000*O + 100*N + 10*E + Y
       {FD.distribute ff Root}
end
```

Ce script déclare dans un premier temps les variables locales pour chaque lettre, ensuite

- *Root* est un Record qui contient les variables représentant les lettres (contrainte basique),
- le champs d'action de *Root* est une suite d'entiers appartenant au domaine 0♯9 (contrainte basique),
- la suite d'entier de *Root* fonctionne par pair distinct (FD.distinct, contrainte non basique),
- les variables *S* et *M* doivent être différentes de zéro (contrainte non basique),
- les chiffres associés aux lettres doivent satisfaire l'équation SEND + MORE = MONEY (contrainte non basique).

La signalisation des contraintes se fait différemment selon le type de contraintes, basiques ou non. Les contraintes basiques sont signalées automatiquement dans l'espace de contrainte, c'est le cas pour les variables comme *Root*.Les contraintes non basiques, elles, ont besoin de mots clés ou de méthode pour se signaler.
La dernière ligne, {FD.distribute ff Root} permet la distribution du champs d'action de *Root* avec la stratégie $first - fail$ (le ff).Cette ligne aurait pu s'écrire

```
{FD.distribute ff [D E M N O R S Y]}
```

La recherche de la solution se fait grâce à un autre module :*Search*. Tout les cas vont être testé pour que la solution optimal soit retourner.

```
{Browse {SearchAll Money}}
```

permet d'afficher le résultat retourner par la module *Search* qui sera :

```
[sol(d:7 e:5 m:1 n:6 o:0 r:8 s:9 y:2)]
```

Pour comprendre le processus de recherche pour ce problème, il faut plus que la solution, la figure E.4 montre tout le déroulement de la recherche avec toutes les nodes de l'arbre.

E.14 OZ Explorer

LeMOZART PROGRAMMING SYSTEM met à disposition un outil permettant de visualiser la recherche de solutions d'un problème combinatoire. Celui-ci peut être utilisé pour visualiser la solution du problème SEND MORE MONEY. En effet, au lieu d'utiliser le module *Search*, c'est le module *Explorer* qui est de mise. A la place de

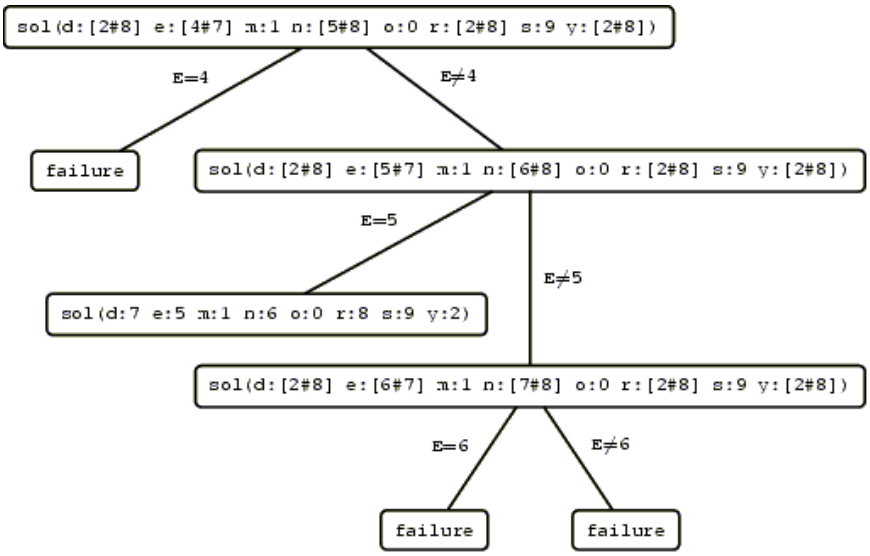

FIG. E.4 – Arbre de recherche pour le problème SEND MORE MONEZ

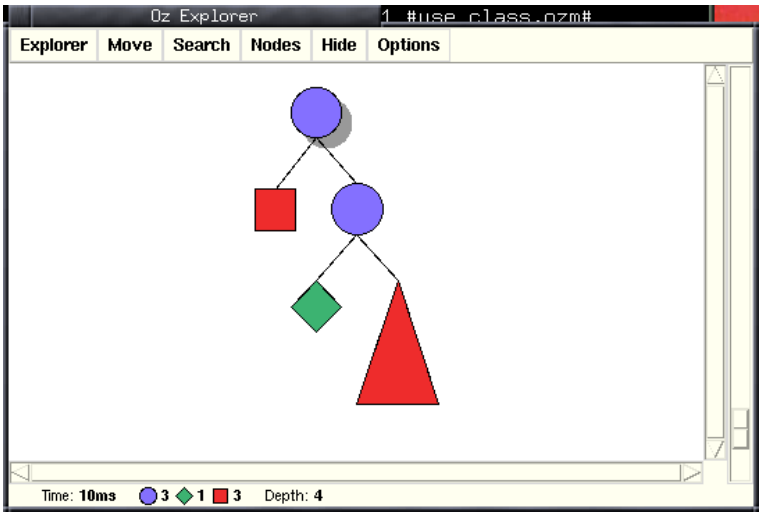

FIG. E.5 – OZ EXPLORER

```
{Browse {SearchAll Money}}
```

il suffit de mettre

```
{ExploreAll Money}
```

ce qui donne la figure E.5 ou le losange représente la solution du problème, les carrés et les triangles les solutions non retenues, celles qui ont échoué à la satisfaction des contraintes.

Glossaire

Analyse	Nom de l'agent récupérant les statistiques des agents de morphologie pour les stocker sous forme de graphe, 15, 77, 88, 93, 95, 97, 98
Aspectuel	Nom désignant la structure générique employée pour créer un agent dans le système multi-agent, 56, 68, 69, 77, 88, 94, 95
Morphologie	Nom de l'agent produisant les statistiques basées sur la communication des agents structurants, 15, 69, 82, 88, 93, 95, 97, 98, 103, 105, 106, 118, 150, 151, 209
Scène	Une scène est une sélection d'une partie des entités actives du système ce que nous appelons un focus sur une partie représentée d'un environnement. Cette partie active et sélectionnée est accessible grâce aux différents capteurs que possède le système évoluant dans l'environnement et à ses connaissances, 47–49, 72, 74, 78, 106, 115, 118, 119, 129, 134, 135, 148–150, 155
Structurant	Terme utilisé pour désigner un agent dont le rôle principal sera de gérer une connaissance, 69, 70, 75, 77