Architecture de COntrôle/COmmande dédiée aux systèmes Distribués Autonomes (ACO^{2}DA) : application à une plate-forme multi-véhicules

Mehdi Mouad

N° d'ordre : D. U : 2437

E D S P I C : 643

UNIVERSITE BLAISE PASCAL - CLERMONT II

ECOLE DOCTORALE
SCIENCES POUR L'INGENIEUR DE CLERMONT-FERRAND

Thèse

Présentée par

MEHDI MOUAD

pour obtenir le grade de

DOCTEUR D'UNIVERSITE

SPECIALITE: Vision pour la robotique

Architecture de COntrôle/COmmande dédiée aux systèmes Distribués Autonomes (ACO^2DA) : Application à une plate-forme multi-véhicules

Soutenue publiquement le 31 janvier 2014

COMPOSITION DU JURY

- Nazim AGOULMINE — Président du jury
- Lounis ADOUANE — Encadrant
- Zahia GUESSOUM — Rapporteur
- Djamel KHADRAOUI — Encadrant
- Cyril NOVALES — Examinateur
- Pierre VIEYRES — Rapporteur
- Philippe MARTINET — Directeur de thèse

Table des matières

Table des figures

Remerciements

À un saint homme, mon père Driss.
Un homme débordant d'amour, de gentillesse, de droiture et de tolérance, qui a marqué tous ceux qui ont croisé son chemin. Tu me manques plus chaque jour. Merci pour l'ami, et le père que tu étais pour moi.
Puisse DIEU te combler de son amour.

À ma mère Tahra, merci pour tout tes sacrifices !
Que DIEU te garde.

Cette thèse a été réalisée dans un cadre de partenariat entre deux laboratoires, l'Institut Paslal à Clermont-Ferrand au sein de l'équipe MACCS (Modeling Autonomy and Control in Complex Systems), ainsi que le Centre de Recherche Public Henri Tudor au Luxembourg département SSI (Service Science & Innovation). Elle n'a pu être menée à bout sans l'aide de certaines personnes auxquelles je veux dire merci.

Ainsi, je tiens à remercier en premier lieu mon directeur de thèse Mr. Philippe Martinet, Professeur à l'Ecole Centrale de Nantes, pour sa confiance et la liberté d'action qu'il m'a accordé. Je tiens à remercier tout particulièrement mes deux Co-Directeurs de thèse, Lounis Adouane, Maître de Conférences à Polytech'Clermont-Ferrand, et Djamel Khadraoui, Docteur Ingénieur au CRP Henri Tudor, pour leur confiance, leur soutien et la liberté d'action et de décision qu'ils m'ont accordé. Je les remercie pour la qualité de leur encadrement et surtout pour leurs qualités humaines.

J'exprime également ma gratitude à Madame Zahia Guessoum, Maître de Conférences HDR à l'Université Pierre et Marie Curie, et à Monsieur Pierre Vieyres, Professeur à l'Université d'Orléan, pour avoir accepté de rapporter sur cette thèse. Je suis reconnaissant envers messieurs Nazim Agoulmine, Professeur à l'Université d'Evry, pour m'avoir fait l'honneur de présider le jury de soutenance, et Cyril Novales, Maître de Conférences à l'Université d'Orléan, pour avoir accepté de prendre part au jury.

Ce travail est le fruit d'une collaboration scientifique et humaine riche, et c'est dans ce sens que j'adresse de sincères remerciements à Pierre Schmidt, Benjamin Gateau et Yannik Zedda pour tout le temps et les efforts que nous avons investis dans la partie simulation.

Ces travaux à l'Institut Pascal et au CRP Henri Tudor m'ont donné l'occasion de pouvoir apprécier l'amitié de certaines personnes que je ne saurais oublier : Brahim Batouch, Wided Guedria, Ahmed Benzerrouk, Omar Ait-Aider, Youcef Mezouar, Guy Guemcam, Hedi Ayed, Jean-Charles Quinton, Emerson Olaya.

Je tiens à remercier tous ceux que j'ai pu connaître et apprécier pendant ces années de thèse : Alwadoud Karim, Mohammed Ben Abdellah, Abdessalam Yassine, Saad El Khomssi, Ahmed Bakari, Mohammed Aznabet, Youssef Joui, Rachid Aboutayeb, Sylvain Rabou, Alexy Catanzaro, Nouredine, Fouzia, Abderrahman, Saffah, Bouchra, Dounia, Naserdine Dahhan, Salahedine, Omar, Abdelhamid, Abdelaziz, Miloudi, Abdellah et Amine.

Je ne saurais terminer sans rendre hommage aux membres de ma famille qui ont patienté tout ce temps et qui m'ont soutenu et supporté : mon père Driss, ma mère Tahra, mon épouse Leila, mon frère Ahmed, ma sœur Houda son mari Adil et enfin Lilya et Dina.

Résumé : La complexité associée à la coordination d'un groupe de robots mobiles est traitée dans cette thèse en investiguant plus avant les potentialités des architectures de commande multi-contrôleurs dont le but est de briser la complexité des tâches à exécuter. En effet, les robots mobiles peuvent évoluer dans des environnements très complexes et nécessitent de surcroît une coopération précise et sécurisée pouvant rapidement devenir inextricable. Ainsi, pour maîtriser cette complexité, le contrôleur dédié à la réalisation d'une tâche est décomposé en un ensemble de comportements/contrôleurs élémentaires (évitement d'obstacles et de collision entre les robots, attraction vers une cible, planification, etc.) qui lient les informations capteurs (provenant des capteurs locaux du robot, etc.) aux actionneurs des différentes entités robotiques. La tâche considérée dans cette thèse correspond à la navigation d'un groupe de robots mobiles dans des environnements peu ou pas connus en présence d'obstacles (statiques et dynamiques). La spécificité de l'approche théorique consiste à allier les avantages des architectures multi-contrôleurs à ceux des systèmes multi-agents et spécialement les modèles organisationnels afin d'apporter un haut niveau de coordination entre les agents/robots mobiles. Le groupe de robots mobiles est alors coordonné suivant les différentes normes et spécifications du modèle organisationnel. Ainsi, l'activation d'un comportement élémentaire en faveur d'un autre se fait en respectant les contraintes structurelles des robots en vue d'assurer le maximum de précision et de sécurité des mouvements coordonnés entre les différentes entités mobiles. La coopération se fait à travers un agent superviseur (centralisé) de façon à atteindre plus rapidement la destination désirée, les événements inattendus sont gérés quant à eux individuellement par les agents/robots mobiles de façon distribuée. L'élaboration du simulateur ROBOTOPIA nous a permis d'illustrer chacune des contributions de la thèse par un nombre important de simulations.

Mots-clés : Systèmes multi-robots mobiles, Systèmes multi-agents, Évitement d'obstacles, Architecture de contrôle/commande, Modèles organisationnels multi-agents, Simulateur ROBOTOPIA.

Abstract : The difficulty of coordinating a group of mobile robots is adressed in this thesis by investigating control architectures which aim to break task complexity. In fact, multi-robot navigation may become rapidly inextricable, specifically if it is made in hazardous and dynamical environment requiring precise and secure cooperation. The considered task is the navigation of a group of mobile robots in unknown environments in presence of (static and dynamic) obstacles. To overcome its complexity, it is proposed to divide the overall task into a set of basic behaviors/controllers (obstacle avoidance, attraction to a dynamical target, planning, etc.). Applied control is chosen among these controllers according to sensors information (camera, local sensors, etc.). The specificity of the theoretical approach is to combine the benefits of multi-controller control architectures to those of multi-agent organizational models to provide a high level of coordination between mobile agents-robots systems. The group of mobile robots is then coordinated according to different norms and specifications of the organizational model. Thus, activating a basic behavior in favor of another is done in accordance with the structural constraints of the robots in order to ensure maximum safety and precision of the coordinated movements between robots. Cooperation takes place through a supervisor agent (centralized) to reach the desired destination faster ; unexpected events are individually managed by the mobile agents/robots in a distributed way. To guarantee performance criteria of the control architecture, hybrid systems tolerating the control of continuous systems in presence of discrete events are explored. In fact, this control allows coordinating (by discrete part) the different behaviors (continuous part) of the architecture. The development of ROBOTOPIA simulator allowed us to illustrate each contribution by many results of simulations.

Key-words : Multi-mobile robots systems, Multi-agent systems, Obstacle avoidance, Control and Management Architecture, Multi-agent organizationnal models, ROBOTOPIA simulator.

Introduction générale

La robotique telle qu'elle est connue aujourd'hui est une science interdisciplinaire comprenant de vastes champs de recherche : la vision, la planification, le contrôle/commande du mouvement, la locomotion, le design, etc. Cependant, l'un des problèmes les plus importants reste la coopération, la planification et la coordination de mouvements au sein d'une architecture de contrôle/commande dans un contexte multi-robots. L'étude des systèmes multi-robots est devenue une préoccupation majeure dans le milieu de la recherche en robotique, car quelles que soient les capacités d'un robot unique, il reste spatialement limité. Bien que le champ des systèmes multi-robots étend la recherche sur les systèmes mono-robot, c'est aussi une discipline à part entière car ils peuvent accomplir des tâches plus complexes que des robots simples ne peuvent pas réaliser.

L'étude des systèmes multi-robots implique l'étude de la coopération et de la coordination entre les robots. La coopération des systèmes multi-robots est définie comme la capacité d'un système à coopérer afin d'accomplir une tâche spécifique. En revanche, la coordination est définie comme l'ensemble des mécanismes et des interactions utilisées pour obtenir une telle coopération. Dans ce cadre, de nombreux enjeux sont engendrés. En effet, la coopération de plusieurs robots signifie qu'ils doivent partager des ressources communes, communiquer, et, pour chaque entité, tenir compte de l'action des autres afin d'accomplir la sienne. En outre, la littérature nous offre une large gamme d'architectures qui permettent le contrôle et la commande des systèmes robotiques, on peut alors les classifier par localité du contrôle, centralisé ou distribué, ou encore par le type du contrôle, cognitif ou réactif. On trouve aussi les architectures dites hybrides, ce sont un mixte entre deux ou plusieurs types d'architectures cités auparavant.

La problématique scientifique couverte par cette thèse se situe dans le cadre des architectures de contrôle multi-robots, couvrant les disciplines de : la planification et re-planification, la coordination multi-robots, l'organisation et la supervision multi-agents. Plus particulièrement, l'objectif de nos travaux de thèse vise à l'élaboration d'une architecture de contrôle/commande pour un système multi-robots évoluant dans des environnements encombrés (présence d'obstacles).

Nous voulons aboutir à une structure de contrôle qui soit ouverte et flexible. En d'autres termes, nous voulons que cette architecture puisse être aisément adaptée et modifiée selon différents contextes d'exécution désirés. L'approche théorique adoptée consiste à s'orienter vers un contrôle/commande hybride combinant à la fois les aspects centralisés, distribués, cognitifs et réactifs. Ainsi, nous facilitons le passage et la combinaison de ces différents aspects au sein de l'architecture proposée selon le contexte d'exécution choisi.

Dans ce contexte, nous proposons d'investiguer le domaine des systèmes multi-agents et spécialement les modèles organisationnels multi-agents, afin d'apporter un haut niveau de coordination entre les agents/robots. Cette approche répondra alors à la contrainte d'ouverture exigée grâce à la facilité de formalisation du problème avec les contraintes dans les différentes spécifications du modèle organisationnel utilisé. En effet, pour confier une tâche plus complexe au système multi-robots, en plus de la spécification de l'organisation, il suffirait d'ajouter les nouveaux comportements/contrôleurs correspondant aux nouvelles tâches dans la base des comportements/contrôleurs déjà disponibles. Nous aborderons aussi l'aspect planification et re-planification de trajectoires dans un contexte multi-robots, ainsi que la navigation en formation. Nous avons illustré les résultats obtenus à travers l'élaboration d'un simulateur multi-agents permettant la simulation des différents aspects d'exécution de l'architecture de contrôle/commande proposée.

Ce manuscrit est organisé en sept chapitres, qu'on peut regrouper en trois principaux thèmes :

— **Problématique de Recherche et État de l'Art,** qui regroupe deux chapitres, le premier est dédié à un état de l'art sur la robotique mobile. Les différentes architectures de contrôle utilisées en robotique mobile y sont classifiée, ainsi que la présentation des systèmes multi-robots, leur classification et les différentes applications et approches utilisées dans ce domaine. Dans le deuxième, une attention particulière est accordée aux systèmes multi-agents, leurs relation avec le domaine de la robotique ainsi que les différents modèles organisationnels et institutionnels existants. En suite, les travaux proposés dans cette thèse sont positionnés par rapport aux travaux existants.

— **Contribution de la thèse : Architecture de contrôle proposée,** il est composée de trois chapitres, le premier est consacré à la présentation de l'architecture de contrôle proposée. Les différents blocs constituant le niveau de contrôle/commande de cette dernière sont expliqués de façon détaillée. Le deuxième détail les composantes du niveau d'organisation de notre architecture de contrôle, et enfin le troisième fait le lien entre ces deux chapitres et présente une illustration du fonctionnement global

de l'architecture proposée dans ses différents modes de fonctionnement. Dans cette partie les contributions apportées sont exposées ainsi que le mécanisme de coopération entre les entités robotiques.

— **Validation et résultats,** ici on trouve deux chapitres, dans le premier on fait une présentations du logiciel de simulation développé ROBOTOPIA, les principales théories sur lesquelles il se base, et on détail ses différentes composantes ainsi que les règles qui le régissent. Le deuxième chapitre est dédié à la présentation des différentes expérimentations effectuées, les différents scénarios d'exécution ainsi que les résultats obtenus.

Le manuscrit se termine par une conclusion générale reprenant les différentes contributions apportées ainsi que les perspectives envisagées.

Chapitre 1

Vers des systèmes multi-robots autonomes

Ce chapitre est consacré à un état de l'art sur les systèmes multi-robots mobiles. Nous commencerons par un historique des robots mobiles, nous nous intéresserons ensuite aux systèmes multi-robots, enfin nous nous focaliserons principalement sur les architectures de contrôle/commande qui leurs sont dédiées leurs types et leur classification.

1.1 Introduction

Le but de notre travail de recherche est l'élaboration d'une architecture de contrôle/commande d'un groupe de robots autonomes non-holonomes opérant dans des environnements non structurés, de façon souple et fiable.

Tout d'abord, nous tenons à préciser le sens d'une architecture de contrôle/commande. En effet, les robots sondent leur environnement, et agissent en se déplaçant pour réaliser leurs objectifs. Les capteurs et actionneurs sont le corps physique qui permet respectivement l'évaluation de l'état proprioceptif et extéroceptif du système, et effectuer une action (mouvement). Entre les données générées par les capteurs et les commandes envoyées aux actionneurs, il y a le logiciel de contrôle/commande appelé architecture. C'est en son sein que les décisions sont prises, ces décisions auront une incidence sur l'état du robot, et devront être obtenues dans un but spécifié par l'utilisateur, et doivent tenir compte de la situation du robot, et éviter les dangers potentiels.

Dans ce contexte, plusieurs types de traitements sont impliqués dans l'architecture de contrôle (gestion capteurs, le filtrage des données, la planification, le calcul de lois de commande, etc.). La façon dont les traitements sont implémentés, et les liens de communication et de contrôle entre eux sont représentés dans l'architecture de contrôle. Les approches de structurelles des architectures de contrôle peuvent être classés en trois grandes catégories :

— les Architectures Réactives versus Cognitives.
— les Architectures Centralisées versus Distribuées.
— les Architectures Hybrides.

Les deux premières catégories se concentrent respectivement sur la vitesse de réponse et la clarté de la structure en termes de niveau de performance (rapidité et coût en énergie). La tendance actuelle est aux architectures dites hybrides, car elles intègrent la nécessité à la fois des aspects réactifs et cognitifs. Mais avant de traiter les architectures de contrôle robotiques, intéressons-nous aux robots, leurs origines, leurs propriétés, leurs caractéristiques et leurs rôles.

1.2 Historique de la robotique mobile

C'est quoi un Robot

Le terme "robot" a été introduit en 1920 par l'écrivain tchèque Karel Capek dans sa pièce de théâtre RUR [1]. Ce terme, provenant du tchèque ROBOTA "travail

1. Rossum's Universal Robots

forcé", désignait à l'origine une machine androïde capable de remplacer l'homme dans toutes ses tâches. En 1942, le mot «robotique» est apparu pour la première fois dans un roman intitulé "Habillage" par le scientifique et écrivain américain Isaac Asimov.

Selon l'Association Japonaise de Robots Industriels (AJRI), les robots sont répartis dans les classes suivantes :

- Classe 1 : robot de manipulation manuelle : un robot à plusieurs degrés de liberté actionné par l'opérateur.
- Classe 2 : robot à séquence fixe : robot de manipulation qui effectue les étapes successives d'une tâche en fonction d'une méthode prédéterminée, qui est difficile à modifier.
- Classe 3 : robot à séquence variable : le même type de robot de manipulation que dans la classe 2, mais les étapes peuvent être modifiées facilement.
- Classe 4 : robot playback : l'opérateur humain effectue la tâche manuellement en conduisant ou en commandant le robot, qui enregistre les trajectoires. Cette information est rappelée lorsque cela est nécessaire, et le robot peut effectuer la tâche en mode automatique.
- Classe 5 : robot à commande numérique : l'opérateur humain alimente le robot avec un programme de mouvement plutôt que de lui enseigner la tâche manuellement.
- Classe 6 : robot intelligent : un robot avec des moyens de comprendre son environnement, et la capacité de mener à bien une tâche malgré les changements dans les conditions ambiantes dans lesquelles celui-ci doit exécuter sa tâche.

La Robotics Institute of America (RIA) ne considère que les machines qui sont au moins de la classe 3 comme étant des robots : " Un robot est un manipulateur reprogrammable, polyvalent conçu pour déplacer des matériaux, des pièces, des outils ou des dispositifs spécialisés, par le biais de mouvements variables programmés pour l'exécution de diverses tâches."

La plupart des robots mobiles en usage dans l'industrie aujourd'hui sont des manipulateurs industriels ("robots d'assemblage") qui opèrent au sein d'un espace de travail limité, et n'ont pas la possibilité de se déplacer. Ici nous nous intéresserons aux robots mobiles qui peuvent changer leur situation par le biais de la locomotion. Un des types les plus communs de robot mobile est le robot unicycle (cf. figure 1.1) à deux roues directionnelles en industrie.

Ce genre de robots fonctionnent dans des environnements spécialement modifiés (boucles d'induction, balises et d'autres marqueurs), et effectuent des tâches de transport le long d'itinéraires fixes. L'inconvénient majeur est qu'ils sont rigides et la modification de l'itinéraire est coûteuse en temps et énergie, donc tout changement imprévu peut conduire à l'échec de la mission. L'alternative à ce

FIGURE 1.1 – Le robot unicycle KIVA à deux roues directionnelles [Systems 13]

genre de robots serait de rendre plus flexible leurs mouvement en les dotant d'intelligence pour qu'ils puissent prendre des décisions et communiquer entre eux, en l'occurrence les rendre plus autonomes.

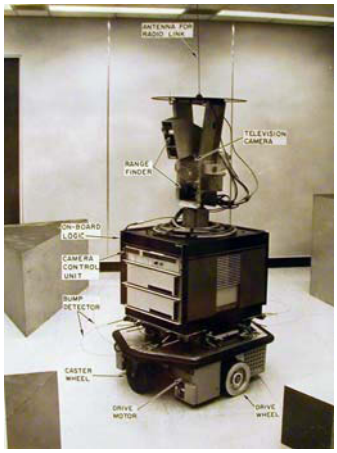


FIGURE 1.2 – Le robot "Shakey" du Stanford Research Institute [SRI 67]

Au début de la robotique, les robots étaient principalement solidaires d'un bâti statique, ce qui n'empêchaient pas qu'ils aient un espace de travail étendu. Le besoin progressif de disposer de robots qui se déplacent (mobiles) s'est imposé par la suite comme une évidence à atteindre [2] (cf. figure 1.2), car à une époque où la flexibilité et l'adaptabilité sont des maîtres mots, cette mobilité ne pouvait être que très souhaitable. D'ailleurs, cette mobilité trouve un grand écho dans différents domaines d'utilisation tels que l'agriculture, les travaux publics ou l'exploration spatial [Adouane 05]. Pour rendre cela possible nous nous intéresserons à trois importants aspects : l'autonomie, l'intelligence et la relation robot-tâche-environnement.

A. L'autonomie

2. Un des premiers robots mobiles au nom de Shakey est apparu en 1967.

Il y a deux principales définitions de l'autonomie [Webster 81] :
— comportement basique initialement programmé sans aide extérieur.
— avoir le pouvoir de l'auto-gouvernance.

Les robots qui contiennent des contrôleurs et des blocs d'alimentation sont autonomes dans le premier cas, mais ça reste une faible autonomie. Cependant, pour faire face à des situations imprévues et s'adapter à des environnements changeants, la puissance de l'auto-gouvernance («forte autonomie») est nécessaire. L'autonomie dans ce cas signifie que la machine est capable de déterminer sa ligne de conduite par son propre raisonnement, plutôt que d'en suivre un prédéfini par des instructions. La forte autonomie exige la capacité de planifier et d'apprendre de l'expérience.

Un robot mobile autonome, a alors la capacité de se déplacer dans son environnement pour effectuer un certain nombre de tâches différentes, et est aussi capable de s'adapter aux changements de son environnement, apprendre de l'expérience et de modifier son comportement en conséquence, et de construire des représentations internes de son monde qui peut être utilisée pour des processus de raisonnement comme la navigation.

B. L'intelligence

Nous pouvons lire dans les manuels de vulgarisation scientifique et les magazines à propos des machines intelligentes, et des robots intelligents une définition utilisable et claire de l'intelligence. L'intelligence se réfère au comportement, et l'aune à laquelle nous la mesurons est très dépendante de notre propre éducation, notre compréhension et notre point de vue. Alan Turing a écrit en 1947 :

"La mesure dans laquelle nous considérons que quelque chose se comporte d'une manière intelligente est déterminée autant par notre propre état d'esprit et de formation que par les propriétés de l'objet considéré. Si nous sommes en mesure d'expliquer et de prédire son comportement, nous pouvons tenter d'imaginer son intelligence. Par conséquent, le même objet peut être vu comme intelligent par un homme, mais le jugement d'un autre peut dire qu'il ne l'est pas, le deuxième homme ayant découvert les règles de son comportement."

L'observation de Turing décrit l'intelligence comme un objectif inaccessible, parce qu'au moment où nous comprenons le fonctionnement d'une machine, ou pouvons prédire son comportement, il n'est plus question d'intelligence. Cela signifie que tant nous progressons dans notre compréhension de la conception et l'analyse des systèmes artificiels, l'intelligence évolue avec nous, tout en restant ainsi hors de notre portée.

Cependant, notre définition de l'intelligence est liée au comportement humain. Nous nous considérons comme intelligent, et donc une machine qui fait ce que nous faisons doit être considérée comme intelligente aussi. Pour la robotique mobile,

ce point de vue a des conséquences intéressantes. Les humains ont toujours pensé que les jeux comme les échecs requièrent de l'intelligence, alors que se déplacer dans un environnement sans aucun accroc majeur est tout simplement ordinaire, ne nécessitant pas d'intelligence. Si l'intelligence est "ce que les humains font, à peu près tout le temps", alors ce comportement ordinaire est la clé de l'intelligence des robots ! Il s'est avéré que construire des robots qui se déplacent sans problème est beaucoup plus difficile que de construire des machines qui jouent aux échecs !

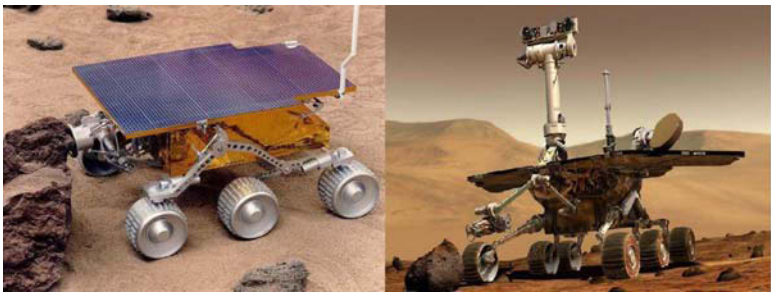


FIGURE 1.3 – Robot mobile "Sojourner" et "Opportunity" utilisés pour les missions Pathfinder et Viking de la NASA [NASA 12]

Un exemple illustrant les différents types d'autonomie, peut être donné au travers du robot "Sojourner" et "Opportunity" (cf. figure 1.3) qui est le premier robot mobile à avoir roulé sur le sol martien. Le Sojourner dispose entre autres d'une batterie est d'un panneau solaire qui lui assure une autonomie énergétique. Il possède aussi une caméra et des capteurs lasers pour pouvoir percevoir son environnement. Quant à son aspect décisionnel, Sojourner a été principalement télé-opéré depuis la Terre. Sachant que la transmission radio des commandes mettent en moyenne 11 minutes pour parcourir les 192 millions de kilomètres séparant Mars de la Terre, Sojourner a été conçu pour fonctionner aussi d'une manière semi-autonome [Volpe 97] appelé *Supervised Autonomous Control*, où les ordres généraux (comme atteindre un objectif) sont donnés au robot depuis la Terre, cependant celui-ci navigue d'une manière complètement autonome entre les roches du sol martien [Adouane 05].

C. La tripartite Robot Tâche Environnement

Le contrôle des robots mobiles nécessite la juxtaposition de trois phases : la perception, la décision, et l'action. La perception construit un modèle même simple de l'environnement du robot, la décision utilise ce modèle pour générer des consignes de mouvement, et finalement l'action transforme ces consignes en commandes adéquates pour les effecteurs du robot. Un contrôle bien mené nécessite donc la maîtrise de ces trois phases [Adouane 05].

Comme l'intelligence, le comportement d'un robot ne peut pas être évalué indépendamment de l'environnement du robot et de la tâche que le robot doit effectuer. Le robot, la tâche et l'environnement dépendent les uns des autres et s'influencent mutuellement (cf. figure 1.4).

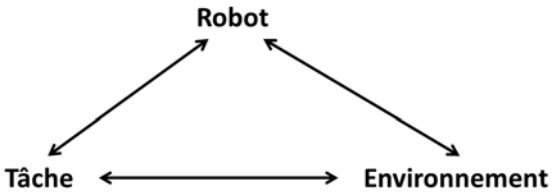


FIGURE 1.4 – La relation entre le Robot, la Tâche et l'environnement.

Le célèbre exemple de Tim Smithers de ce fait est l'araignée [Smithers 94] : très compétente à la survie dans les endroits exigus, mais totalement incompétente dans l'eau ! Si on fait l'analogie avec la robotique mobile, on remarque que dans ce cas le robot semble être intelligent dans une situation, mais incompétent dans l'autre. Par conséquent, cela signifie qu'un robot à usage général ne peut pas exister (de la même manière qu'un être vivant polyvalent n'existe pas). La fonction d'un robot et son fonctionnement sont définis par son propre comportement au sein d'un environnement spécifique, en tenant compte d'une tâche spécifique.

On a pu rencontrer lors de notre investigation de l'état de l'art sur la robotique, divers types de robots dédiés à divers application (cf. figure 1.5). On peut citer les robots volant ou drones qui sont utilisés pour l'exploration de zones dangereuses, ou dans les applications de surveillance et d'attaque militaires. Les robots sous-marins pour l'exploration, la cartographie sous-marine et aussi la surveillance de la température, la pression et le PH des rivières pour l'analyse de la pollution et la mesure des risques pour la vie aquatique, on trouve aussi les robots marcheurs humanoïdes, etc.

Après avoir présenté succinctement les principales caractéristiques inhérentes à une entité robotique élémentaire (structure, autonomie, contrôle), nous passons à présent au grand champ d'investigation qui est celui de la robotique mobile collective, appelé aussi systèmes multi-robots coopératifs. Sa classification, ses mécanismes, et ses différents courants conceptuels et méthodologiques seront traités dans ce qui suit.

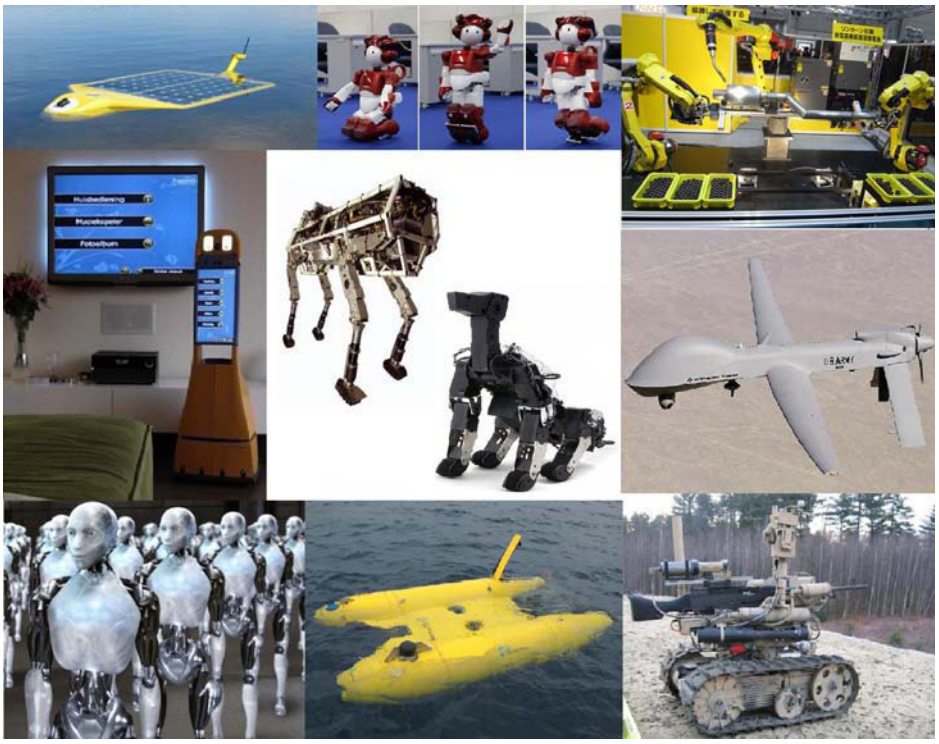


FIGURE 1.5 – Exemples de différents types de robots rencontrés dans l'état de l'art.

1.3 Les systèmes multi-robots

Les systèmes multi-robots sont actuellement une des préoccupations majeures de la recherche en robotique. En effet, les systèmes multi-robots peuvent accomplir des tâches qu'un seul robot ne peut pas effectuer, notamment parce qu'un seul robot est spatialement limité. En 1989, Brooks et Flynn du laboratoire MIT's Artificial Intelligence ont proposé ce qu'ils ont appelé une idée radicale dans le domaine de l'exploration du système solaire : remplacer un grand robot explorateur par une collection de petits robots mobiles appelés "rovers" [Brooks 89]. Ils ont estimé que le coût par kilogramme des rovers serait fortement réduit. Diminuer un peu la fiabilité pour chaque rover mobile serait acceptable, du moment que l'échec d'un rover unique ne mettrait pas en péril toute la mission. En effet, la mission pourrait être prévue avec une estimation de la fiabilité inférieur à 100%. Après l'atterrissage, soit ensemble ou en petits groupes, les rovers se disperseront en couvrant une larges surface. Bien qu'au jour d'aujourd'hui l'application des systèmes multi-robots n'est pas très répondue notamment dans l'espace (par exemple le robot Curiosity développé par la NASA cf. figure 1.3). Cette idée radicale présente la principale motivation pour l'utilisation de systèmes multi-

robots, en effet, à plusieurs ils peuvent se disperser plus rapidement et couvrir un large espace pour les missions de reconnaissance, ou bien transporter des objets dispersés dans une zone de stockage. Ils peuvent même être utilisés dans des missions sous-marines comme la cartographie des fonds des mers, la récupération des balises échoués ou encore les boites noires suite au crache d'avions.

Les premières études sur les systèmes multi-robots ont étés ceux de Fukuda (robotique cellulaire [Fukuda 89]), d'autres chercheurs du MIT travaillant dans le domaine des sociétés robotisées pouvant aller jusqu'à vingt robots mobiles : [Brooks 90a], [Mataric 92]. D'autres efforts de pionniers dans l'étude des systèmes multi-robots ont été réalisés par les universités de Stanford [Caloud 90] et Carnegie Mellon [Noreils 90].

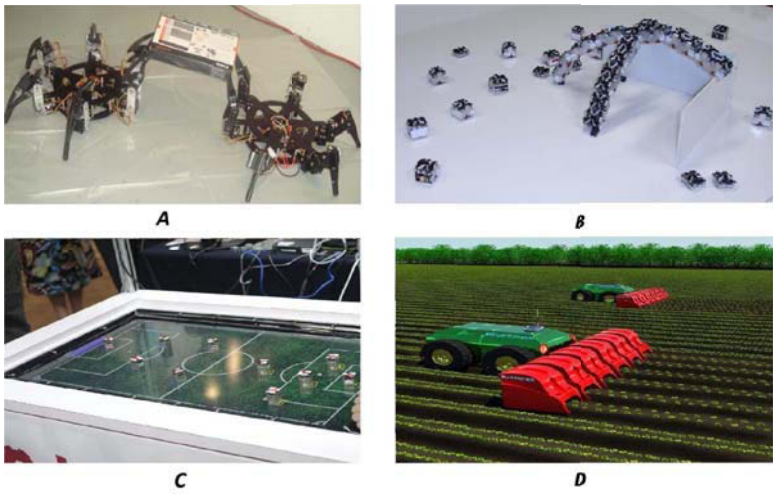


FIGURE 1.6 – Exemples de différents types de tâches coopératives multi-robots : A. Coopération de robots pour soulever un objet [Agheli 13], B. Robots coopérant par assemblage [Saska 11], C. Coopération de robots footballeurs [Camacho 06], D. Coopération de robots agricoles [Robots 13]

Plus récemment, plusieurs robots mobiles autonomes ont été proposées pour une utilisation dans des missions de sauvetage [Davids 02], [Yamauchi 99], d'exploration [Burgard 05], [Weisbin 00], [Rooker 07], dans des applications militaires [Maxwell 13], de divertissement avec des équipes de football robotisées [Camacho 06], [Kim 04], et dans l'agriculture [Robots 13] (cf. figure 1.6).

Un système multi-robots peut être défini comme un ensemble de robots opérant dans le même espace de travail [Farinelli 04]. En outre, le comportement coopératif est organisé comme suit : Compte-tenu de certaines tâches spécifiées par un designer, un système multi-robots affiche un comportement coopératif si,

en raison de certains mécanismes sous-jacents (i.e., le mécanisme de coopération ou de coordination), il y a une augmentation dans l'efficacité totale du système.

Ainsi, de nombreuses motivations nous ont permis de nous orienter vers les systèmes multi-robots :

1. l'amélioration du temps d'exécution d'une tâche, voire la qualité d'exécution de celle-ci. Nous prenons comme exemple la tâche de pousser un objet lourd par un groupe de robots, ce qui consiste à diviser la force globale nécessaire à cette tâche, sur les différentes entités robotiques qui peuvent détecter les points optimaux où appliquer leurs efforts et pousser ensemble dans la direction du mouvement souhaitée [Adouane 05]. Ou encore l'utilisation d'un groupe de robots mobiles pour explorer un environnement dans le but d'élaborer une cartographie de cet environnement [Burgard 05], chaque robot explore alors une partie différente de la carte, et établit une carte partielle qui servira à construire la carte globale.
2. avec un groupe de robots mobiles, on bénéficie d'une distribution des capteurs et des actionneurs sur les entités robotiques, ce qui rend le système plus tolérant à certains défauts de fonctionnement. Par exemple, la panne d'un robot n'entraîne pas l'arrêt systématique de la mission, du fait de la redondance des entités robotiques.
3. l'utilisation d'un groupe de robots peut s'avérer une solution plus simple et moins chère que la conception d'un seul robot dont la structure et le contrôle peuvent vite devenir complexes et encombrants.

Utiliser un groupe de robots mobiles paraît donc une idée séduisante. Cependant, cela entraîne aussi l'apparition de nouvelles contraintes liées à l'interaction de plusieurs entités dans le même environnement. La prise en compte de ces aspects est indispensable pour contrôler un Système Multi-Robots (SMR).

1.3.1 Les tâches coopératives multi-robots

Pour qu'une tâche soit qualifiée de coopérative, les robots chargés de l'accomplir doivent coordonner leurs actions en interagissant dans le même environnement, et en partageant impérativement des objectifs et des ressources communs : pousser ou porter le même objet, cartographier un environnement, etc. sont des exemples de tâches coopératives dans lesquels chaque robot doit tenir compte des actions des autres en accomplissant la sienne [Adouane 05].

La tâche coopérative peut être considérée comme l'objectif à atteindre par le système multi-robots. Cet objectif change selon les applications désirées ou les types de robots utilisés.

L'utilisation d'un SMR doit alors améliorer l'efficacité d'exécution de cette tâche par rapport à la qualité d'exécution offerte par un seul robot. Elle peut aussi rendre la tâche possible, alors que celle-ci est irréalisable par un seul (par exemple, porter ou pousser un objet trop lourd pour un seul robot, explorer une zone trop vaste, etc.). Cela signifie que dans tous les domaines où la robotique mobile peut être utile, la robotique coopérative peut apporter de nouvelles possibilités intéressantes. On peut alors citer l'exploration spatiale [Huntsberger 03], le maintien de formation de robots ou d'avions militaires [Balch 99], le transbordement dans les ports et aéroports [Alami 98b], le convoi de robots pour le transport urbain de passagers [Bom 05], sauvetage et navigation sous-marins [Bahr 09], etc.

Une partie des travaux sur la robotique coopérative cherche à reproduire les activités des animaux, insectes, etc. En effet, l'homme a toujours eu des inspirations biologiques comme dans la figure (1.7) qui montre une analogie entre la navigation en formation d'avions de chasse acrobatiques et un système biologique composé d'oiseaux volant en formation. Ce type d'approches extrait de la nature est venue enrichir le domaine de la robotique coopérative grâce à l'introduction des architectures de contrôles dédiées [Brooks 85]. Ces dernières sont décrites par la relation existante entre les trois éléments au contrôle des robots mobiles : perception, décision, et action. Le principe comportemental *« Behavior-based »* sert à examiner les caractéristiques et les règles de la vie sociale de certains animaux et insectes (oiseaux, poissons, fourmis, abeilles, etc.) afin de les reproduire sur des robots réels. Ainsi, dans [Mataric 95], des actions de fourragement, rassemblement en troupeaux, dispersion, etc. inspirées de sociétés vivantes ont été reproduites sur de véritables robots mobiles.

FIGURE 1.7 – Analogie avions militaires-oiseaux volant en formation

Dans ce qui suit, nous nous intéresserons plus particulièrement à deux des plus importantes tâches multi-robots qui concernent nos travaux de recherche, c'est la planification de trajectoire ainsi que la coordination de mouvements des systèmes multi-robots.

1.3.2 Planification de trajectoires pour les systèmes multi-robots

Compte tenu de notre domaine de recherche, à savoir les systèmes multi-robots, et vu le nombre conséquent de robots que peut contenir un SMR, il est utile de s'attarder sur la manière avec laquelle ces robots vont évoluer dans leur environnement, et comment ils vont planifier leurs trajectoires afin d'atteindre leurs destinations finales.

La planification de trajectoires pour les systèmes multi-robots [Svestka 98] a été largement étudiée au cours des dernières décennies. Le problème de planification de chemin en robotique consiste à générer un chemin continu pour un robot donnée entre une configuration initiale et finale. Sur ce chemin, le robot ne doit pas traverser des régions interdites (habituellement des obstacles) [Hopcroft 84]. Il y a deux approches de base pour la résolution du problème de la planification de chemin multi-robots : centralisée et distribuée. Dans le cas de l'approche centralisée, chaque robot est traité comme un système composite, et la planification est faite dans un espace de configuration composite, formée en combinant les espaces de configurations des robots individuels. Alors que dans le cas de l'approche distribuée, les chemins sont d'abord générées pour les robots de manière indépendante, puis leurs interactions sont considérées. L'avantage des approches centralisées, c'est qu'ils trouvent toujours une solution quand il en existe une. Cependant, la difficulté pratique est que si on exige l'exhaustivité dans le résultat, on obtient des méthodes dont la complexité temporelle est exponentielle dans l'espace de configuration composite. Mais les planificateurs distribués sont incomplets par nature (probabilités de configurations divers et variées) et peuvent donc conduire à des situations de blocage. Même le problème apparemment simple de planification de mouvement pour un nombre arbitraire de robots rectangulaires dans un espace de travail rectangulaire vide est encore PSPACE-complet [3] [Hopcroft 84].

Planification centralisée

Ardema et Skowronski [Ardema 91] ont décrit une méthode pour générer un contrôle de mouvement sans collision pour deux manipulateurs spécifiques, et ce en modélisant le problème comme un jeu non coopératif. [Bennewitz 02] a présenté une méthode pour trouver et optimiser des méthodes prioritaires pour ces

3. Dans la théorie de la complexité algorithmique, PSPACE est l'ensemble de tous les problèmes de décision qui peuvent être résolus par une machine de Turing polynomiale en utilisant une quantité d'espace. Un problème de décision est PSPACE-complet s'il est dans la classe de complexité PSPACE et que chaque problème peut être réduit à cette classe dans un temps polynomial.

techniques de planifications distribuées. Les approches existantes appliquent un régime de priorité unique ce qui les rend trop sujettes à l'échec dans les cas où des solutions existent (par exemple plusieurs robots rencontrent des obstacles au même moment). En cherchant dans l'espace des régimes de priorités, leur approche permet de surmonter cette limitation. Il effectue une recherche aléatoire pour trouver des solutions et de minimiser la longueur générale du chemin. Pour cibler cette recherche, l'algorithme est guidé par des contraintes générées à partir de la spécification des tâches. Les expérimentations menées ont conduit non seulement à une réduction significative du nombre d'échecs dans lesquels aucune solution ne peut être trouvée, mais ils ont également montré une réduction significative de la longueur générale du chemin. D'autres approches centralisées pour résoudre les problèmes de planification multi-robots comprennent diverses techniques de champs de potentiels. Ces techniques appliquent différentes approches pour traiter le problème de minima locaux. D'autres méthodes se basent sur la restriction des mouvements des robots pour réduire la taille de l'espace de recherche, comme dans [Tournassoud 86] où ils ont proposé une approche basée sur les champs de potentiels où la coordination de mouvements de robots est exprimée comme un problème d'optimisation locale, au lieu d'aborder le problème global de planification de trajectoires pour des systèmes multi-robots, ce qui conduit plutôt à des algorithmes impraticables. Ils se sont donc limités à l'utilisation d'une classe spécifique de trajectoires. La méthode est basée sur l'idée qui dit que le calcul des trajectoires de n'importe quel objet en mouvement, peut être traduit dans le choix de quelques obstacles mobiles virtuels, choisis comme hyperplans tangents aux objets. Les algorithmes de manipulation de la géométrie simple peuvent être mis en œuvre pour l'évitement en temps réel, et leur application peut être étendue aux manipulateurs à la géométrie plus complexe.

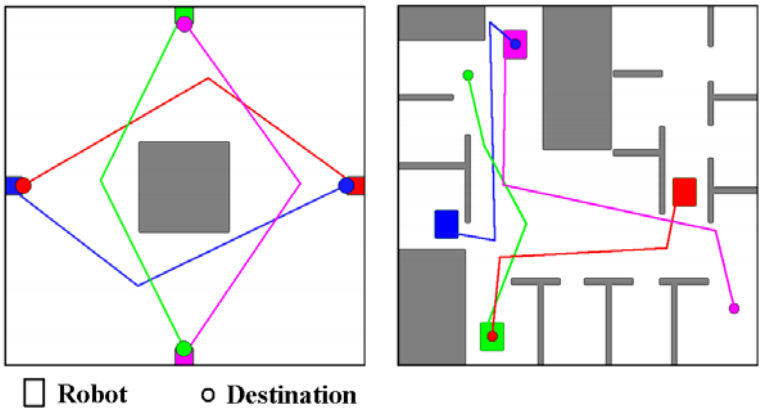


FIGURE 1.8 – Deux générations de trajectoires pour un système multi-robots mobiles

Une nouvelle méthode de navigation présentée dans [Morette 11], basée sur le modèle cinématique directe et l'approche de commande prédictive, où la recherche de la trajectoire optimale est décalée dans l'espace des paramètres continues, ce qui permet l'exploitation de toutes les fonctionnalités du robot. Dans [Barraquand 92] les auteurs ont présentés une technique de champs de potentiels pour plusieurs robots dans des environnements avec des couloirs étroits. Ils ont présenté quatre techniques numériques de champs de potentiels pour la construction du graphe local minimal pour la planification de trajectoire des robots : le meilleur d'abord (best-first mouvement), le mouvement aléatoire, le mouvement vallée guidée (valley-guided), et le mouvement limité. Toutes ces techniques appliquent la même approche générale, ils construisent un champ de potentiel dans l'espace de configuration du robot, puis un graphe reliant les minima locaux de ce potentiel. Ces techniques sont basées sur les différentes façons d'échapper aux minima locaux. Les principes de ces quatre techniques de planification sont quelque peu indépendantes de la représentation de l'espace de travail du robot et de l'espace de configuration. Toutefois, leur mise en œuvre efficace a été rendue possible (cf. figure 1.8).

Planification distribuée

Les planificateurs distribués aident à la génération de trajectoires de robots de manière indépendante avant d'employer différentes stratégies pour résoudre les conflits éventuels. Ils peuvent ne pas trouver une solution, même s'il y en a une. Une approche distribuée assez connue est la planification dans la configuration spatio-temporelle proposée par Erdmann et Lozano-Perez [Erdmann 87]. Elle peut être appliquée à chaque robot en prenant en considération les positions et les orientations de tous les autres robots à chaque point dans le temps. L'approche consiste à attribuer des priorités à chacun des robots qui se déplacent. Les trajectoires sont planifiées pour les robots dans l'ordre déterminé par l'ordre de priorité défini au préalable. Ainsi, le problème général est réduit à plusieurs versions du problème de la planification pour un robot mobile unique en présence d'autres robots qui se déplacent en présence d'obstacles fixes. Une trajectoire du robot est alors trouvée par la recherche d'un chemin du départ jusqu'à l'arrivée grâce à la configuration spatio-temporelle. Ferrari et al. [Ferrari 98] utilise un schéma de priorité fixe et choisit au hasard des détours pour les robots ayant une priorité inférieure. Des variantes de la solution initiale pour chemins sans collision de robots sont obtenues par rapport aux paramètres de qualité qui donnent des règles heuristiques pour évaluer la robustesse du plan. Tandis que les facteurs d'impact de collision sont considérés pour évaluer la qualité d'un seul chemin, les indices de performance, sont utilisées pour évaluer la qualité globale d'un plan. Erdmann et Lozano-Perez [Erdmann 87] ont proposés un schéma de planification

prioritaire. Chaque robot choisi une trajectoire, planifiée en évitant les collisions avec les obstacles statiques ainsi que les robots, qui sont considérés comme des obstacles mobiles. Une autre approche de la planification distribuée est la méthode de coordination de chemin. L'idée clé de cette technique est de garder les robots sur leurs parcours individuels et de laisser les robots s'arrêter, avancer, ou même reculer sur leurs trajectoires afin d'éviter les collisions. Dans [Tournassoud 86] les auteurs ont proposés une méthode pour suivre un chemin avec une contrainte de temps des robots avec des couples actionneurs/vitesses limitées. Un espace de coordination à deux dimensions est construit pour identifier une zone de collision le long des chemins, qui se transformera en une seule zone temporelle en fonction de l'évolution et de la longueur de la trajectoire, en prenant comme référence le profil de vitesse de l'un des deux robots.

Les approches de planification hybrides

De nos jours, plusieurs chercheurs se concentrent sur les techniques de planification hybrides, qui intègrent deux ou plusieurs techniques de planification en même temps afin d'éviter plusieurs problèmes indésirables rencontrés par chaque approche individuellement. Par exemple, une approche centralisée est utilisée pour la planification de parcours global de tous les robots dans l'espace de travail. Ces trajectoires sont en outre corrigées grâce à une deuxième technique de planification distribuée qui se fait localement dépendant de l'obstacle, sa position, sa forme, etc. [Ren 08].

1.3.3 Coordination de mouvements pour les systèmes multi-robots

Ayant vu comment peut-on planifier les trajectoires pour les SMR, la problématique qui émerge est celle de la coordination de leurs mouvements pour assurer la non collision des entités entre elles et avec leur environnement. Étant donné qu'au niveau des systèmes multi-robots on manipule des robots de grande valeur aussi bien technologique que financière, la coordination de mouvements est là pour ajouter un haut degrés de sécurité dans la flottille et minimiser les dégâts le plus possible. Elle peut aussi coexister avec les différentes approches de planifications.

L'étude de la coordination de mouvements en formation de robots non-holonomes étend l'étude de la planification de mouvement d'un robot unique. La coordination de mouvement est un phénomène remarquable dans les systèmes biologiques et un outil extrêmement utile pour les groupes de robots. Ces tâches de coordination doivent souvent être réalisées avec un minimum de communication entre

les robots, et aussi avec une information limitée sur l'état global du système. L'application de ces stratégies de coordination, généralement inspirées par la biologie devrait tenir compte des propriétés locales des robots, comme la saturation des contrôleurs etc. De nombreux travaux de recherche sur la coordination de mouvements des systèmes multi-robots distribués sont basés sur les modèles simplifiés des agents robots. En revanche, les stratégies de coordination qui prennent en compte les contraintes du robot non-holonome ont souvent besoin de l'information de l'état global du système. Nous présentons ci-après quelques stratégies générales et des outils d'analyse pour la coordination de mouvement multi-robots.

Les potentiels artificiels

Leonard et Fiorelli [Leonard 01], ont proposés une approche basée sur des potentiels artificiels qui définissent des forces de contrôle des interactions entre les robots voisins, et sont conçus pour appliquer une distance inter-robots souhaitée. Un leader virtuel est utilisé pour appliquer une influence à son voisinage par le biais d'un potentiel artificiel supplémentaire, il est également utilisé pour manipuler la géométrie du groupe et diriger le mouvement du groupe. Ce framework peut être appliqué à des groupes homogènes, sans avoir à commander les robots. Ozdemir et Temeltas dans [Ozdemir 07], ont définis une règle de rotation des forces potentielles afin d'éviter les cas des minima locaux. Le principal inconvénient de ces approches, c'est qu'elles sont basées sur des agents de masse ponctuelles, les contraintes physiques ou dynamiques ne sont pas incluses dans le modèle.

L'approche Leader-Follower

La stratégie de Leader-Follower (meneur-suiveur) a d'abord été introduite dans le système de contrôle par l'économiste allemand Heinrich Freiherr von Stackelberg, qui a publié le livre Marktform und Gleichgewicht en 1934, qui décrit ce modèle [Cruz 78]. Ces méthodes de contrôle, également connues sous le nom de stratégies de Stackelberg, sont appropriées pour les classes de problèmes de systèmes où il y a de multiples critères, multiples décideurs et une information distribuée. Dans ces stratégies, le contrôle des actions des suiveurs est basé sur l'état du leader. Le concept du Leader-Follower a été largement utilisé dans le contrôle des systèmes multi-robots. Feddema et al. [Feddema 02], ont étudié l'observabilité et l'accessibilité des contrôles de robots mobiles coopératifs basés sur l'approche Leader-Follower. Les autres applications du leader-follower dans les systèmes multi-robots ont été présentées pour les robots terrestres [Das 02], [Cruz 07],

[Chen 03], systèmes multi-robots aériens autonomes [Seiler 01], [Vanek 05] et sur des pelotons de robots sous-marins sans pilotes [Edwards 04], [Yang 07].

Dans les approches Leader-Follower appliquées à la navigation en formation d'un groupe de robots, chaque robot est positionné dans la formation par la géométrie relative à son voisin prédéfinie. Chaque robot suit son chef prédéfini avec une certaine relation géométrique. En utilisant cette relation leader-follower, un motif géométrique de n robots peut être obtenu. En règle générale, il n'y a qu'un leader unique de la formation. Ce chef unique ne suit aucun autre robot dans la flottille, mais il suit une trajectoire prédéfinie, ou doit aller vers une position prédéfinie dans l'environnement. La stabilité des systèmes basés sur l'approche Leader-Follower de robots unicycles a été étudiée notamment par Lechevin et al. [Lechevin 06].

1.4 Les architectures de contrôle/commande en robotique

Notre besoin en autonomie dans des environnements non structurés exige la capacité, à tout moment, d'évaluer l'état du robot et de son environnement, et ensuite prendre des décisions cohérentes.

Les logiciels de navigation mis en œuvre pour ce genre de robots deviennent rapidement complexe à la fois dans leur conception / implémentation et dans leur analyse. La littérature, est concentrée la plupart du temps sur la qualité de conception d'un logiciel. Marco, Healey et McGhee dans [Healey 96] proposent d'évaluer une architecture logicielle par rapport à huit questions principales :

- le contrôleur permet-il d'évaluer facilement la réponse du système et est-il possible d'ajuster les paramètres de commande facilement ?
- cela peut être fait en temps réel lors d'un test (dans des conditions réelles à l'extérieur) ?
- un nouveau capteur peut-il être ajouté sans nécessiter de changements majeurs dans le logiciel ?
- à quel niveau du code doit-on intervenir et combien de fonctions sont nécessaires lors de l'ajout d'un nouveau capteur ?
- quelle quantité de code doit être changé si l'on veut ajouter ou supprimer une étape dans la mission ?
- comment le code doit être modifié afin d'évaluer la performance d'un capteur ou actionneur ?
- est-il facile d'accepter des données provenant d'un autre ensemble de capteurs ?

— est-il facile de changer les conditions qui définissent les signaux de transition ?

Ces huit points sont basés sur un ensemble de critères généraux. Nous pouvons donc mettre en évidence plusieurs critères sur lesquels doit être basée la conception des architectures logicielles dédiées aux robots autonomes :

Modularité : la complexité des logiciels de navigation exige du concepteur de les diviser en plusieurs composants plus petits qui peuvent être développés, implémentés et testés séparément.

Ouverture : le logiciel doit être développé progressivement, de nouveaux composants logiciels doivent être ajoutés par différents acteurs, tout en réutilisant les composants existants.

Flexibilité : le logiciel doit avoir une structure flexible pour ajouter ou supprimer facilement des composantes logiciels pendant ou après la phase de développement.

Temps réel : un signal de commande doit être généré en toutes circonstances et dans un temps donné.

Gestion des contraintes : le robot doit être en mesure d'atteindre des objectifs longs termes de la mission tout en répondant aux contraintes immédiates.

Fiabilité et robustesse : le robot effectuera une série d'objectifs planifiés en présence d'incertitude et doit pouvoir gérer les situations qui sont temporellement imprévisible.

Surveillance du bon fonctionnement : le robot doit être en mesure de vérifier le bon fonctionnement de toute l'instrumentation, de détecter et de réagir de manière appropriée à toute défaillance.

1.4.1 Les architectures centralisées versus distribuées

Les architectures centralisées

Les architectures centralisées ont été essentiellement motivés par deux facteurs [Khoshnevis 98] :

— l'ambition d'alléger la structure physique des robots : débarrasser les robots de capteurs embarqués, et la puissante unité de calcul. Ainsi, réduire considérablement leur coût.
— le désir d'une meilleure connaissance de l'environnement global par la concentration des capteurs au niveau d'une unité centrale, ce qui peut assurer une meilleure prise de décision par rapport à des robots se basant que sur des informations locales fournies par des capteurs embarqués.

Dans ce type d'architecture, le contrôle est délocalisé par rapport à la structure physique des robots et se trouve au niveau d'une unité centrale (superviseur dans [Jones 01] ou planificateur central dans [Noreils 93]) qui gère et garantit l'exécution de la tâche. Cette unité contient les parties sensorielles (capteurs) afin de collecter les informations de l'environnement, et calcul les consignes de commande (positions, vitesses) qu'elle envoie aux entités robotiques. Elle est aussi responsable de prendre les décisions pour la réalisation de la tâche globale et les communiquer aux robots. Elle doit donc avoir une puissante capacité calculatoire pour satisfaire à toutes ces exigences.

Dans les architectures de contrôle centralisées, la manière de coordonner les actions entre le superviseur et les robots peut être comparée à celle de groupements sociaux, notamment animal ou encore humains dans certains aspects sommaires (la communication, la hiérarchie, etc.).

Les architectures distribuées

Devant la complexité grandissante des systèmes robotisés, l'idée de distribuer le contrôle sur plusieurs entités coopérantes afin de réaliser une tâche principale est très attrayante. La distribution du contrôle permet, si elle est concluante, de briser la complexité inhérente au système, et conduit ainsi à une mise en œuvre du contrôle beaucoup plus simple et surtout plus près du modèle du système [Adouane 05].

Par opposition aux architectures de contrôle centralisées, les ressources (capteurs, unités de calcul, etc.) des architectures de contrôle distribuées sont dans ce cas distribuées sur tous les éléments du SMR. Chaque robot n'utilise alors que ses propres capteurs et sa propre unité de calcul et de traitement. Il doit aussi pouvoir communiquer et partager des informations avec les autres robots afin de les informer et de s'informer sur la progression de la mission.

Ce type d'architectures de contrôle est donc venu dans l'ambition de pallier aux limitations des architectures de contrôle centralisées. Il est à remarquer néanmoins, qu'elles partagent la même source d'inspiration qui reste les sociétés vivantes. Ainsi, le projet CEBOT (CEllular roBOTics System) [Fukuda 89] est un système distribué inspiré de l'organisation des cellules d'un organisme biologique. Les entités robotiques sont alors assimilées à des cellules couplées entre elles. Ces robots cellules sont continuellement reconfigurable afin de répondre aux exigences de l'environnement. Beaucoup d'autres systèmes s'inscrivant dans le cadre des systèmes distribués peuvent être trouvés dans la littérature [Yamaguchi 01], [Zhiqiang 06].

Alors on centralise ou on distribue ?

Chaque type d'architectures de contrôle présente ses avantages et ses inconvénients. Il est alors difficile voire impossible de privilégier un type plutôt qu'un autre sans connaître la nature de la mission confiée au SMR ni le contexte de sa réalisation.

Modules	Architecture centralisée	Architecture distribuée
Perception et localisation	L'unité centrale utilise des capteurs centralisés et calcule la localisation absolue dans l'environnement de chaque robot. Les robots ne connaissent pas leur environnement.	Les robots utilisent leurs capteurs embarqués et se localisent de façon relative par rapport à leurs consignes.
Décision	L'unité centrale décide, calcule et génère directement des variables de commande (vitesses) qu'elle envoie aux robots.	Les robots génèrent leurs propres consignes de type chemin, trajectoire, points de passages, etc.
Action	Les entités robotiques appliquent directement les variables de commande calculées par l'unité centrale à leurs moteurs.	Les robots assurent le respect des consignes qu'ils ont générées à travers des lois de commande avec un asservissement en boucle fermée sur ces consignes.

TABLE 1.1 – Architecture centralisée versus distribuée [Adouane 05].

Dans les architectures centralisées, l'unité centrale dispose d'informations globales de l'environnement, ce qui aide à la coordination d'actions entre les éléments du SMR, et à la prise de décision. La limitation de ce type d'architectures réside dans cette même unité centrale du fait qu'elle doit gérer le contrôle et la communication de toutes les entités robotiques. De plus, le SMR dépend entièrement de cette unité, ce qui signifie qu'un défaut du superviseur central implique l'arrêt complet du système.

Les architectures de contrôle distribuées quant à elles, permettent de tirer profit de la distribution des ressources : chaque robot se contente de prendre une décision le concernant, en fonction des informations locales de l'environnement délivrées par ses propres capteurs et de sa communication avec ses voisins. Cette distribution allège la tâche des unités de calcul. De plus, les défauts de fonctionnement sont mieux gérés et la panne d'un robot ne signifie pas l'arrêt complet du

SMR. En revanche, l'absence d'un superviseur global et d'une information globale sur l'environnement rend difficile la coordination des robots et l'optimisation de l'exécution de la mission.

Le tableau 1.1, [Adouane 05] résume les principales différences observées entre ces deux types d'architectures centralisées et distribuées. En pratique, beaucoup de SMR n'ont pas un contrôle strictement centralisé ou strictement distribué et utilisent des architectures de contrôle hybrides centralisées/distribuées pour bénéficier des avantages des deux approches. Par exemple, dans [Beard 01], [Das 02], [Feddema 02], [Stilwell 05], des robots pourtant décisionnellement autonomes, sont soumis néanmoins ponctuellement à un planificateur central.

1.4.2 Les architectures réactives versus cognitives

Les architectures réactives

Dans une architecture réactive plusieurs modules relient les entrées capteurs aux actionneurs. Chaque module implémente un comportement, c'est à dire une fonctionnalité élémentaire du robot associant à chaque vecteur d'entrée (ensemble des valeurs capteurs) un vecteur de sortie appliqué aux actionneurs. Ces comportements sont dit **réactifs** car ils fournissent immédiatement une valeur de sortie dès qu'une valeur se présente en entrée. Par exemple un comportement dédié à l'évitement d'obstacle dans un robot mobile, appliquera une commande aux actionneurs des roues de façon à faire changer la direction à chaque fois que le capteur de proximité indique la présence d'un objet.

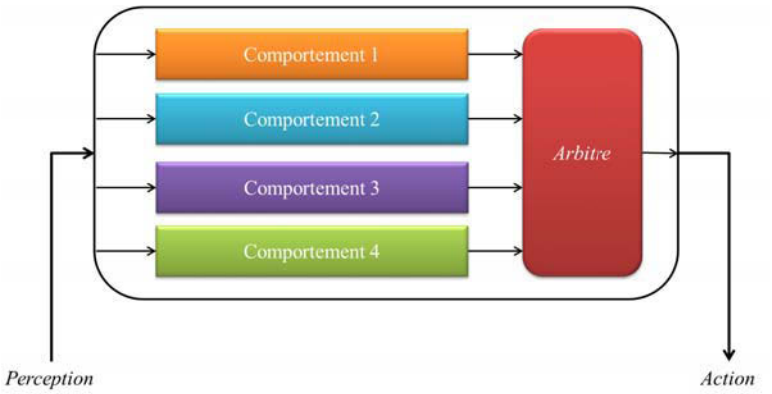


FIGURE 1.9 – Architectures réactives arbitrées [Brooks 86].

Inspirées de l'observation des comportements animaux, ces architectures sont construites selon l'idée que la composition d'un ensemble de comportements élémentaires simples peut émerger un comportement plus évolué. L'intégration de

plusieurs comportements amènent souvent des contradictions au niveau des commandes générées par ces derniers. On résout alors ce problème en ajoutant un module d'arbitrage qui va se charger de pondérer les sorties de chacun des comportements pour générer une sortie unique (figure 1.9). Le comportement final (global) du robot est alors étroitement lié à cette pondération.

Selon l'application que l'on veut faire réaliser au robot, il faut alors déterminer un ensemble "de poids" qui vont être utilisés par le module d'arbitrage. Toute la complexité réside dans la recherche de la bonne combinaison de valeurs numériques (poids) en intégrant des objectifs de haut-niveau, qui va donner le comportement global souhaité.

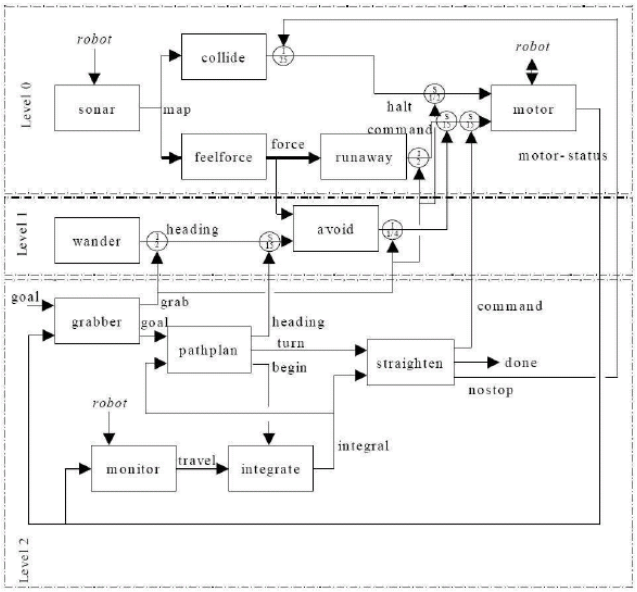


FIGURE 1.10 – Architecture subsumption [Brooks 90b].

Pour pallier à cette difficulté, vers la fin des années 80, Brooks a proposé l'architecture dite "subsumption" [Brooks 90b]. Selon cette approche, les différents comportements sont classés par niveaux de compétence. Chaque niveau renferme un ensemble de modules qui implémentent un comportement donné. Chaque module d'un niveau peut inhiber les entrées ou les sorties de modules du niveau immédiatement inférieur (figure 1.10).

Une autre variante des travaux de Brooks est l'architecture DAMN (Distributed Architecture for Mobile Navigation) proposée par Rosenblat [Rosenblatt 95]. Dans cette architecture (figure 1.11) plusieurs comportements élémentaires tels que le suivi de route ou l'évitement d'obstacle adressent des votes à un module spécial appelé "arbitre de commande". Ces entrées sont combinées et le résultat

est une commande qui sera transmise au contrôleur du robot. Un poids est assigné à chaque comportement reflétant sa priorité relative dans le contrôle du robot. A la différence de l'architecture subsumption, les poids peuvent varier au cours du temps. En effet, un module nommé "générateur de mode" va calculer ces derniers tout au long de la mission et favoriser ainsi à chaque instant certains comportements par rapport à d'autres, selon la situation rencontrée.

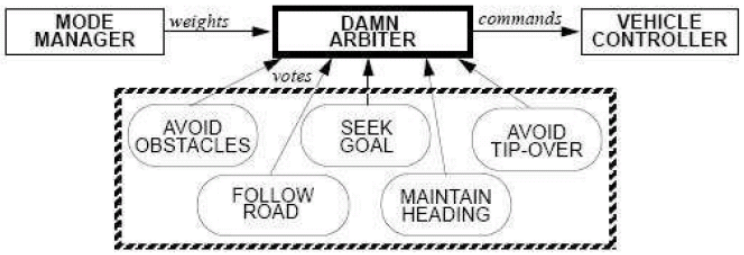


FIGURE 1.11 – Architecture DAMN [Rosenblatt 95]

Plusieurs expériences ont montré l'efficacité de ces architectures en robotique mobile. La réactivité qui les caractérise a mené à des résultats souvent impressionnants dans la navigation en milieu encombré. Cependant, la complexité de conception de ces architectures ne permet pas de les utiliser pour des applications demandant un comportement plus évolué que la navigation.

Les architectures cognitives

Le modèle d'organisation des architectures cognitives, également appelées architectures délibératives (figure 1.12), centre la conception sur le système décisionnel. Ces architectures sont organisées, dans la plupart des propositions, en plusieurs couches (également appelées niveaux) [Gat 98], [Albus 97] et [Albus 02]. Une couche communique uniquement avec la couche directement inférieure et la couche directement supérieure. Ces architectures comportent typiquement trois couches :

— le niveau fonctionnel contient des modules de perception qui sont en charge de la transformation des données numériques, provenant des capteurs, en données symboliques. Il contient également des modules de génération de trajectoires et des modules d'asservissement, qui sont responsables de la transformation des données provenant du niveau supérieur, en données numériques applicables aux actionneurs.

— le niveau exécutif est en charge de la supervision des tâches du robot (e.g. "se diriger vers la source de chaleur la plus proche"). Elle contient

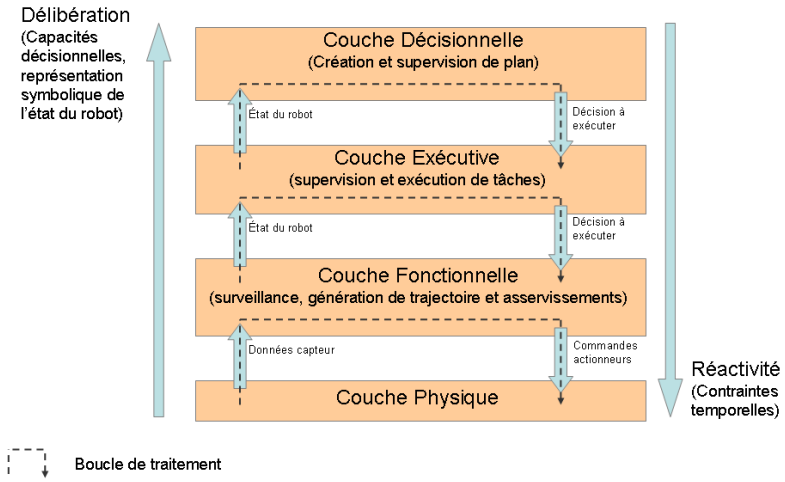


FIGURE 1.12 – Les architecture cognitives [Albus 02]

pour cela des modules qui dirigent l'exécution des modules de la couche fonctionnelle.

— le niveau décisionnel est en charge des mécanismes de planification (i.e. création et supervision de plans). Elle gère la façon dont le robot va enchaîner différentes tâches afin de réaliser chaque objectif de sa mission. Les mécanismes de décision de plus haut niveau, appelés traditionnellement mécanismes de délibération sont les mécanismes centraux dans ces architectures [Tate 94].

L'information provenant des capteurs est propagée verticalement, de la couche fonctionnelle vers la couche décisionnelle, en traversant potentiellement toutes les couches intermédiaires. L'information est transformée, au fur et à mesure des traitements réalisés dans chaque couche, en une information de plus en plus abstraite. Ceci nécessite des traitements adéquats afin de fusionner les données et d'en extraire les informations plus abstraites nécessaires au plus haut niveau. Inversement, la propagation de la décision (sous forme de données symboliques) se fait de la couche décisionnelle vers la couche fonctionnelle, en traversant successivement toutes les couches intermédiaires. Le niveau fonctionnel applique alors, en fonction de cette décision, les asservissements adéquats (i.e. activation des modules d'asservissements qu'elle contient). Le principal problème des contrôleurs conçus avec de telles architectures, vient de leur manque de réactivité, c'est-à-dire la capacité de réagir en temps voulu à des modifications rapides dans l'environnement.

Un exemple d'architectures cognitives est celui de l'architecture d'Alami [Alami 98a] du LAAS (Laboratoire d'Analyse et d'Architecture des Systèmes) à Toulouse, qui

propose un environnement logiciel constitué d'outils pour modéliser, programmer, exécuter et dans une certaine mesure valider, des architectures de contrôle. Cet environnement s'appuie sur un modèle (figure 1.13) qui définit un découpage d'une architecture de contrôle en trois niveaux :

— le niveau fonctionnel constitue l'interface entre les entités des couches supérieures et la partie physique du système.

— le niveau de contrôle d'exécution, est en charge de vérifier les requêtes envoyées aux modules du niveau fonctionnel et l'utilisation des ressources du robot.

— le niveau décisionnel contient les mécanismes de décision du robot, en particulier la production et la supervision de plans et la réaction à des situations particulières (e.g. pannes matérielles). Ce niveau comprend deux entités : un exécutif procédural et un planificateur/exécutif temporel. Le planificateur/exécutif temporel gère la planification de la mission globale du robot (production de plans pour réaliser les missions) sur un horizon à long terme. L'exécutif procédural est responsable de la supervision des missions envoyées par les opérateurs humains.

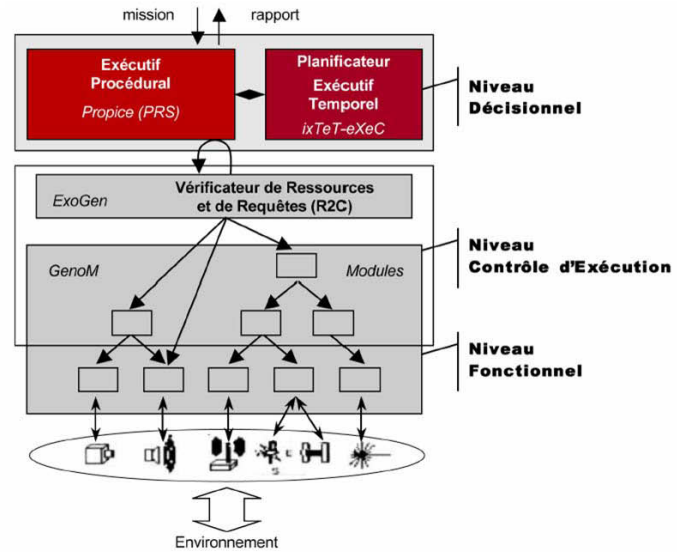


FIGURE 1.13 – Architecture du LAAS [Alami 98a]

Alors cognitive ou réactive ?

Comme nous avons pu le constater, chaque école présente à travers ses spécificités des caractéristiques intéressantes. L'avantage principal de l'approche cog-

nitive est qu'elle peut intégrer plusieurs principes de haut niveau : raisonnement, communication de haut niveau, etc. et elle s'appuie sur une modélisation de l'environnement lui permettant de prendre des décisions optimales avant d'entreprendre une quelconque action [Benzerrouk 11]. Cependant, ces avantages sont rattrapés par certains inconvénients :

— les principes de haut niveau nécessitent des temps de calculs importants et des moyens à coûts élevés,
— l'environnement d'évolution doit être modélisable. En effet, un environnement fortement dynamique nécessite des mises à jour fréquentes, ce qui peut devenir rapidement ingérable.

Il reste que les développements réalisés sur le plan technologique ont engendré des avancées intéressantes pour les approches cognitives, ce qui aide à surmonter certains inconvénients liés au temps de calcul grâce à la puissance actuelle des machines. Cependant, les environnements dynamiques sont synonymes de mise à jour régulière de leurs cartes et de re-planification de la nouvelle tâche du robot dans le nouvel environnement. Dans les environnements fortement dynamiques, et même si certains travaux réclament avoir nettement allégé le temps de calcul [VanDenBerg 05], ce dernier reste directement lié à la fréquence de re-planification.

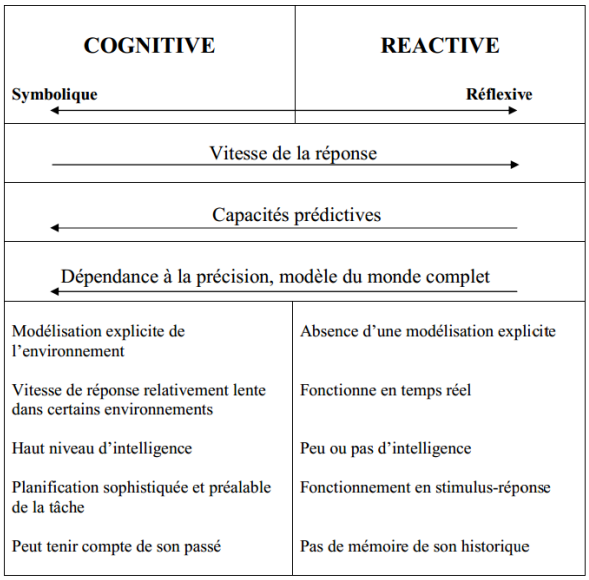


FIGURE 1.14 – Contrôle cognitif versus réactif [Adouane 05].

Les architectures de contrôle réactives permettent d'avoir une réponse plus rapide grâce à un lien direct capteurs-actionneurs. Disposer des comportements

élémentaires dans le cas des approches comportementales permet de les tester individuellement jusqu'à ce qu'ils soient adaptés à leurs tâches (trouver les meilleurs gains de commande, vérifier leur stabilité, etc.). La mise à jour se fait alors simplement en les ajoutant aux comportements déjà existants. En contrepartie, l'absence d'une représentation explicite de l'environnement et de raisonnement de haut niveau (décomposition de la mission et répartition des tâches) limitent leurs applications dans des cas complexes. La figure 1.14 [Arkin 98], [Adouane 05] compare les caractéristiques de deux types d'architectures de contrôle.

1.4.3 Les architectures hybrides

Les architectures hybrides représentent une combinaison des différents types d'architectures de contrôle existant afin d'augmenter le rendu final du système, à savoir rapidité, flexibilité et efficacité. Les avantages des deux approches précédentes (i.e., réactives et cognitives) ont poussé certains roboticiens à combiner les deux approches : on trouve alors des architectures de contrôle combinant navigation réactive et planification de trajectoire (cognitive) [Ranganathan 03], dans le but de pallier aux inconvénients des deux autres approches.

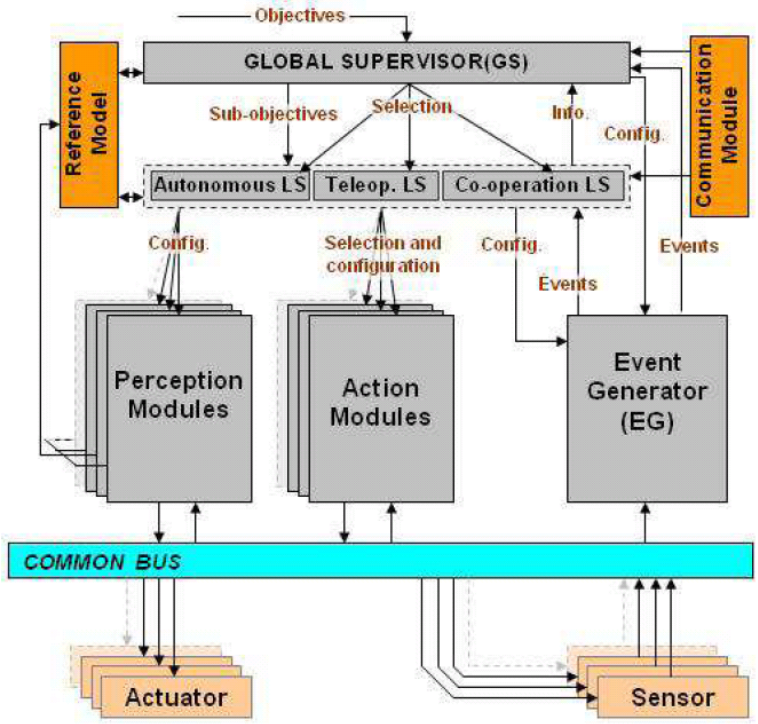


FIGURE 1.15 – Architecture COTAMA du LIRMM [Jalaoui 05]

Les architectures hybrides permettent de prendre en compte à la fois l'aspect cognitif et réactif, ou encore l'aspect centralisé et distribué. Ces architectures se développent pour répondre à la nécessité qu'ont les robots de fournir un comportement autonome dans des environnements inconnus. Ces architectures intègrent une hiérarchisation des couches permettant la symbolisation et donc la prise de décision. A cette hiérarchisation s'ajoutent des boucles réactives imbriquées permettant à chaque couche de fournir des réactions adaptées à sa dynamique.

Un exemple d'architecture hybride a été développé par El Jalaoui [Jalaoui 05] au LIRMM (Laboratoire d'Informatique de Robotique et de Microélectronique de Montpellier), il propose une architecture de contrôle pour AUV (Autonomous Under water Vehicles), c'est un exemple assez intéressant d'architectures hybride. Cette architecture se compose de trois niveaux (figure 1.15) [Jalaoui 05] : superviseur global de contrôle, superviseur de contrôle local et le contrôle de bas niveau. Chaque niveau gère les objets correspondant à son niveau d'abstraction, trois types d'objets sont utilisés : le superviseur global reçoit un ensemble d'objectifs (définition de la mission) de l'opérateur et transmet un ensemble de sous-objectifs au superviseur local dédié. Le superviseur local donne à la commande de bas niveau et aux modules d'instrumentation un ensemble de commandes.

Dans [Gilles 11] un autre exemple d'architecture hybride se basant sur un formalisme multi-niveaux, permettant ainsi un contrôle hiérarchique se basant sur une représentation graphique qui permet de distinguer précisément toutes les caractéristiques de la tâche robotique souhaitée.

1.5 Conclusion

Ce chapitre nous a permis en premier lieu d'introduire le domaine de la robotique mobile et ses applications. Étant donné que l'objectif de la thèse est de contrôler un groupe de robots mobiles naviguant en collaboration dans un environnement peu connu, nous nous sommes d'abord intéressé aux éléments nécessaires au contrôle d'un seul robot mobile, à savoir la perception, la décision et l'action. Avec la maîtrise du contrôle d'un robot mobile, l'idée d'en faire coopérer plusieurs devient attrayante. L'objectif est évidemment de tirer profit de la coopération du groupe. Nous nous sommes alors intéressés aux systèmes multi-robots, leur classification, leurs planification de tâches et leurs coordination afin d'avoir une idée globale sur le contrôle et la gestion des systèmes multi-robots. L'objectif étant de pouvoir s'orienter vers un choix à travers lequel on va procéder à la coordination de notre SMR.

Nous avons pu voir dans ce premier chapitre les différentes architectures de contrôle existantes dans la littérature. Elles ont été classifiées selon leur méthode

d'implémentation : centralisées au niveau d'un superviseur global, ou distribuées, ou encore hybrides.

En regroupant toutes les briques de l'état de l'art étudiés, nous nous sommes engagé dans la réalisation d'une architecture de contrôle hybride, pouvant reproduire, selon les besoins, un grand nombre de types de comportements, une architecture centralisée ou distribuée, ou encore centralisé/distribué, et pouvant aussi contenir du contrôle cognitif et réactif avec la possibilité de mixer les deux.

Vu les dernières avancées informatiques dans le domaine des systèmes multi-agents et leur haute performance dans les domaines de la coopération et l'autonomie, nous allons nous intéresser dans ce qui suit à ces systèmes dans le contexte de la robotique coopérative, nous verrons leurs définitions, leurs relations avec le domaine de la robotique ainsi que leurs mécanismes de coopération.

Chapitre 2

Les systèmes multi-agents situés

Les systèmes multi-agents ont aussi des apports importants dans le domaine de la robotique coopérative. Cela a commencé avec l'apparition de la notion d'Intelligence Artificielle (IA), puis de l'Intelligence Artificielle Distribuée produite par un ensemble d'agents coopératifs. Ainsi, plusieurs domaines de la robotique ont pu bénéficier d'avancées technologiques, dont le domaine de la robotique mobile (terrestre, aérienne, sous-marine, etc.).

Dans ce chapitre nous présenterons les différentes approches multi-agents utilisées dans la robotique, que ce soit pour la coordination ou encore dans le cadre des architectures de contrôle.

2.1 Introduction aux systèmes multi-agents

Un agent physique est une entité qui opère dans le monde réel : un robot, un avion ou une voiture par exemple, à l'inverse d'un agent logiciel qui lui est virtuel, mais capable de coordonner voir d'ordonner et assister la réalisation d'actions concrètes. C'est ce concept d'agents qui nous intéresse dans cette thèse.

Les agents ainsi définis sont, non seulement capables de raisonner comme dans les systèmes d'intelligence artificielle classiques, mais ils sont aussi capables d'agir. L'action, qui est un concept fondamental pour les systèmes multi-agents, est basée sur le fait que les agents effectuent des actions qui modifient leur environnement et donc leur prise de décisions futures. Ils peuvent également communiquer avec d'autres, c'est aussi l'un des principaux modes d'interaction entre les agents. Comme les agents sont dotés d'autonomie, cela signifie qu'ils ne sont pas dirigés par des commandes de l'utilisateur (ou un autre agent), mais par un ensemble de tendances qui peuvent être sous forme d'objectifs individuels, ou des fonctions de satisfaction ou de survie que l'agent essaie d'optimiser.

La notion d'interaction constitue l'essence d'un système multi-agents puisque c'est grâce à elle que les agents vont pouvoir produire des comportements collectifs complexes. Jacques Ferber définit l'interaction comme une mise en relation dynamique de deux ou plusieurs agents par le biais d'un ensemble d'actions réciproques [Ferber 95]. Les interactions s'expriment ainsi à partir d'une série d'actions dont les conséquences exercent en retour une influence sur le comportement futur des agents.

Un système multi-agents se distingue, donc, d'une collection d'agents indépendants par le fait que les agents interagissent en vue de réaliser conjointement une tâche ou d'atteindre conjointement un but particulier. Ces agents peuvent interagir en communiquant directement entre eux ou indirectement par l'intermédiaire d'un autre agent ou même en agissant sur leur environnement [Jarras 02]. La communication dans ce cas est vue comme une forme d'interaction.

Au premier abord, un système multi-agents peut d'une manière simpliste être considéré comme un ensemble d'agents partageant un environnement commun. Certes, la notion d'environnement est primordiale dans un système multi-agents mais elle reste, tout de même insuffisante pour qualifier un système multi-agents. En outre, il est difficile de caractériser un système multi-agents, il est donc question d'identifier les aspects fondamentaux qui caractérisent un système multi-agents d'un système mono-agent.

D'après la définition de Ferber [Ferber 95], On appelle système multi-agents un système composé des éléments suivants :

— un environnement E, disposant en général d'une métrique.

— un ensemble d'objectifs O, auxquels on peut associer une position dans E à un moment donné. Ces objets (hormis les agents) sont passifs : les agents peuvent les percevoir, les créer, les détruire et les modifier.
— un ensemble d'agents A, lesquels représentent les entités actives du système.
— un ensemble de relations R, qui unissent les objets (et agents) entre eux.
— un ensemble d'opérateurs Op permettant aux agents de A de percevoir. produire, consommer, transformer et manipuler des objets de O.
— Des opérateurs chargés de représenter l'application de ces opérations et la réaction du monde à cette tentative de modification.

Le monde des systèmes multi-agents peut être qualifié de discipline qui s'intéresse aux comportements collectifs produits par les interactions de plusieurs entités autonomes et flexibles appelées agents et que ces interactions consistent en la coopération et la concurrence entre ces agents.

2.2 Les systèmes multi-agents en robotique

Le domaine dans lequel la recherche sur les agents a eu le plus d'impact sur la recherche en robotique est le domaine des systèmes multi-robots. Cela est dû à la fréquence des problèmes liés à la complexité combinatoire dans les tâches multi-robots. Si plusieurs robots doivent être coordonnées d'une certaine façon, cette tâche est plus difficile que celle de la prise de décisions d'un seul robot, car en plus de décisions individuelles, il faut se préoccuper des combinaisons d'actions sélectionnées.

Une grande partie de la recherche sur les systèmes multi-robots ces dernières années, se concentre sur une seule tâche à la fois. Quelques exemples de ces missions canoniques comprennent le maintien de formation [Kaminka 08], la couverture multi-robot [Agmon 08], la recherche de nourriture [Rosenfeld 08], [Kaminka 10], les patrouilles de surveillance [Agmon 08], [Jensen 11], [Basilico 09], [Smith 11], [Agmon 11] et [Fave 09], la recherche urbaine et sauvetage est présentée dans [Murphy 08]. Beaucoup d'entre eux sont abordés sous l'angle d'un problème de contrôle distribué. Les contrôleurs sont mis en place pour chaque tâche, mais comme les applications robotiques gagnent en complexité, les contrôleurs devront tenir compte de l'attribution, la planification et l'exécution de plusieurs tâches, qui ont lieu simultanément ou en séquence.

2.2.1 Les agents et les architectures de contrôle

Aujourd'hui la recherche en robotique doit adresser des missions de plus en plus complexes pour les robots. Auparavant, le problème de commande d'un robot pourrait être résolu par la mise en place de contrôleurs conçus avec soin. Maintenant, les tâches relativement simples (et très utiles) comme trouver un chemin, éviter les obstacles, et naviguer vers un lieu de destination, sont des domaines de recherche matures chez les roboticiens. Les robots sont désormais censés aller au-delà d'une navigation vers un but, il faut qu'ils puissent effectuer des missions dans lesquelles il est envisagé de remplir plusieurs buts changeants : des tâches multiples, qui doivent être réalisées en simultané ou dans l'ordre. Ces missions complexes nécessitent de la planification et de la gestion des ressources, avec en particulier des demandes de prise de décisions en permanence.

Les chercheurs dans le domaine des agents ont commencé à prôner l'idée que les agents ne sont pas seulement les planificateurs. Au contraire, les agents doivent raisonner à travers des plans [1] : les générer, les adaptant et les opposer, pour prendre des décisions qui profiteront à différents objectifs. En particulier, l'accent sera mis sur l'intégration de la planification et de l'exécution dans le développement des robots de manière à refléter ce raisonnement basé sur les plans. Cette approche, repose sur l'utilisation d'une représentation en plan comme un objet central, qui est générée, adaptée, modifiée, et gérée par la durée de vie du robot. Les planificateurs sont utilisés non seulement pour générer des plans, mais aussi pour fournir des prévisions. Indépendamment, les roboticiens ont commencé à envisager des notions similaires, en s'appuyant sur la hiérarchisation des modules de planification et d'exécution, d'une manière qui permet d'apprendre, envisager et décider quel plan exécuter. Nous retenons les deux principales approches pour les architectures multi-robots utilisant les systèmes multi-agents que nous présenterons dans ce qui suit.

Le language BDI (Beliefs, Desires, Intentions) : L'agent dans un environnement dynamique doit gérer le processus de planification, mais aussi décider du moment de la planification afin d'éviter au maximum une re-planification (coûteuse en temps et énergie). Pour ce faire, la construction des agents doit permettre une représentation explicite des croyances, des objectifs et des plans (que ce soit pré-planifiés ou générés dynamiquement). Ceux-ci seront révisés, manipulés et traités par la sélection de l'action des agents et le processus de perception.

Dans une large mesure, l'énorme littérature sur les attitudes mentales des agents, nous permet de considérer des théories et des architectures de type BDI (Belief, Desire, Intention) comme réponse à ce défi [Sardiña 11]. Ces dernières

1. Un plan est définit comme étant la liste de toutes les actions à exécuter par un agent dans le but d'accomplir un but ou un objectif.

années, on voit apparaître l'évolution de la théorie et de la pratique des représentations des régimes qui se prêtent à la planification et à l'exécution. Une variété d'implémentations BDI universitaires et commerciales existe, comme CogniTAO [CogniTeam 09].

Représentation de plans de première classe : En plus de ces langages BDI qui ont été utilisés dans les domaines de la robotique, il y a eu aussi de nombreuses représentations de plans qui ont été testées sur des robots, mais qui donnent le statut de première classe seulement au plan, mais pas aux croyances ni aux objectifs. Néanmoins, ils sont utiles dans la construction de robots qui appliquent un contrôle basé sur les plans. Il s'agit notamment des représentations à états finis [Tousignant 11], des représentations Petri-net [Ziparo 10], Et les langages de planification et d'ordonnancement temporel (par exemple, T-REX [Py 10], ce qui permet la programmation multi-résolution des tâches).

Le problème de la coordination prend aussi une grande place dans les objectifs de la thèse, en effet, nous voulons élaborer une architecture de contrôle qui nous permet d'être capables d'apporter de la coordination et un certain degré de coopération entre les robots du SMR. C'est pour cela qu'on va s'intéresser dans la section suivante à la problématique de la coordination d'agents dans un contexte multi-robots.

2.2.2 Les agents et la coordination d'un groupe de robots

Une idée fondamentale acquise dans la communauté AAMAS (Autonomous Agents and Multi-Agent Systems) au cours des 15 dernières années est que des tâches multi-agents peuvent être décomposés conceptuellement et techniquement en deux composantes. La première, appelée *taskwork* (tâche), incluant des capacités individuelles dépendantes du domaine. La seconde, appelée le *teamwork* (travail de groupe) qui comprend les capacités de collaboration (en équipe), ou le maintien d'autres relations sociales, et *socialwork* (travail social) qui comprend des mécanismes de choix social, comme les protocoles d'attribution des tâches aux différents membres des équipes, ou des protocoles pour la prise de décisions conjointes. La combinaison de *taskwork* et *socialwork* donne un système multi-agents spécialisé.

Cette prise de conscience s'est manifestée de différentes façons dans la recherche en robotique. On va aborder dans ce qui suit quelques-uns de ces travaux de recherche associant le monde des agents à celui des robots.

Allocation de tâche inspirée par le marchandage : En termes d'impact sur la robotique, l'utilisation de méthodes basées sur le marchandage ou les enchères pour l'attribution des tâches aux robots jouit d'une grande popularité. Elle

est maintenant adoptée et étudiée par les roboticiens en dehors de la communauté AAMAS. À partir de ce travail sur l'utilisation des mécanismes de marché pour la coordination de robots pour accomplir les tâches d'exploration [Xin 06] et de cartographie [Ortiz 11], il y a eu beaucoup de travaux dans ce domaine, qui ont permis de relever les défis qui se posent lorsque l'on travaille avec des robots, un exemple présenté dans [Michael 08] aborde ce défi en utilisant des protocoles de coordination basés sur les enchères pour accomplir une mission de contrôle de formation, où chaque agent est capable d'enchérir pour la répartition des tâches. Cette même méthode d'allocation de tâche est présentée dans [Tang 07] est dédiée à la génération de tâches multi-robots.

Décision commune d'un système multi-robots : Dans ce cadre, les chercheurs ont démontré que le travail d'équipe d'un système multi-robots implique plus que de l'allocation des tâches. Il induit en plus un accord sur un objectif commun, et un accord sur un plan pour atteindre cet objectif commun pour assister les robots coéquipiers lorsque c'est nécessaire, etc. Un exemple illustrant cela est présenté dans [Traub 11], c'est une analyse des avantages de l'utilisation d'une architecture utilisant une équipe de systèmes multi-robots, en utilisant un modèle de logiciel standard (COCOMO [Fei 92]).

Le travail d'équipe d'un système multi-robots a été étudié au sein de la communauté des systèmes multi-agents depuis de nombreuses années, il y a eu une série d'articles sur le travail d'équipe, utilisant la logique pour modéliser et prévoir le travail d'équipe. Ces modèles décrivent les conditions dans lesquelles un agent doit informer ses coéquipiers de ses croyances privées, ce qui permet de maintenir efficacement la synchronisation de l'équipe. Le principe de cette approche est décrit par un ensemble de règles de comportement qui, si elles sont suivies, pourraient emmener l'agent à agir de manière appropriée dans le cadre d'une équipe, indépendamment de la tâche qui lui a été assignée, ou le domaine d'application.

Un exemple illustrant le travail d'équipe multi-robots est celui du maintien de formation d'un groupe de robots mobiles, où les robots doivent se déplacer à l'unisson le long d'un chemin donné, tout en conservant une forme géométrique donnée. Diverses méthodes de maintien de formation ont été étudiées [Kaminka 08], [Elmaliach 08]. Ces méthodes sont distribuées ; tous exigent que chaque robot exécute un processus de contrôle local, qui exécute la commande qui correspond au rôle du robot. Par exemple, un robot de gauche dans une formation en triangle équilatéral garderait le leader dans une distance fixe correspondant à la distance maintenue par le robot à droite, de sorte qu'entre lui, le robot leader et le deuxième follower il on maintient un angle de 60 degrés.

La notion de réputation est de plus en plus investiguée dans le domaine des systèmes multi-agents pour améliorer la coordination entre agents dans les envi-

ronnements décentralisés, ce qui pourrait donner un progrès significatif dans le domaine des systèmes multi-robots, dans [Guemkam 13] on trouve une approche intéressante appliquée aux réseaux Ad hoc mobiles.

2.3 Les modèles de description d'organisation et d'institution multi-agents

La coordination est un aspect central des systèmes multi-agents. Dans le sens le plus large, la "coordination" représente un agent qui prend en considération les autres agents dans son environnement [Durfee. 01]. Elle peut-être utilisée dans le contexte de l'allocation de ressources, la répartition des tâches, la communication, ou le mouvement. La plupart des approches de coordination se situent dans l'une des trois grandes catégories, ou peuvent être construites à partir d'une combinaison d'approches émanant de ces catégories [Boutilier 96] :

Les rôles et les conventions sont les moyens les plus simples de coordonner des agents, et les moins flexibles. Une convention est communément connue pour être une règle à laquelle adhèrent les agents. Par exemple, le contrôle de circulation est souvent basé sur des conventions telles que l'arrêt aux feux rouges. Cette coordination a l'avantage d'être simple et ne nécessite aucun temps de configuration [Fitoussi 00]. Cependant, elle n'est pas flexible, et s'appuie sur tous les participants qui connaissent les conventions et coopèrent entre eux.

Une extension de la notion de convention de coordination est l'utilisation des rôles au sein des structures organisationnelles. Le rôle actuel d'un agent détermine les conventions qui sont appropriées, et quels protocoles sont mis à sa disposition.

La communication est une autre technique de coordination commune à l'homme. La coordination par communication a un petit temps de configuration et nécessite peu de coûts en bande passante. Elle nécessite un langage commun, la flexibilité de ce langage détermine la flexibilité de la coordination qui en résulte. Le développement d'un potentiel pour les modèles probabilistes de la lange [Fischer 05], permet (par exemple) l'adaptation à des environnements changeants.

L'apprentissage est à la fois le moyen le plus complexe et le plus souple de la coordination. Il peut être combiné avec l'approche des conventions ou celle de la communication, les conventions peuvent être apprises, par exemple, grâce à la communication [Kazakov 04]. Il existe de nombreuses façons différentes d'appliquer l'apprentissage à la coordination, par exemple à partir de l'apprentissage du choix entre des protocoles de coordination [Toledo 05] ; ou par le biais d'apprendre à utiliser les techniques de communication simples pour la coordination comme dans [Kazakov 04]. Les techniques d'apprentissage permettent l'évolution

des techniques de décision dans les espaces vastes et complexes, aussi elles permettent l'adaptation à des systèmes dynamiques. Cependant, ils ont un coût de configuration élevé, comme dans l'apprentissage de la prise de décision, qui peut prendre beaucoup de temps et qui nécessite un calcul intensif.

Dans ce qui suit nous présenterons une étude des modèles d'Organisation et des modèles d'institution. Pour chaque modèle de contrainte, en plus de définir leurs différentes dimensions de modélisation, nous verrons le niveau de description des contraintes (pouvant aller d'une description abstraite à l'implémentation) et leur portée c'est-à-dire le nombre d'agents pouvant être influencés.

2.3.1 Modèles d'organisation

Les modèles organisationnels spécifient la structure et les fonctionnalités d'une société d'agents, c'est-à-dire un ensemble d'agents évoluant au sein du même environnement. Ces modèles sont plus ou moins orientés autour de la spécification de la fonctionnalité du système et/ou de la structure de l'ensemble d'agents.

Modèles d'analyse au sein de GAIA

GAIA [Wooldridge 00] [Zambonelli 03] est une méthodologie pour l'analyse et la conception de systèmes orientés agent. GAIA modélise un SMA en une organisation informatisable composée de rôles interagissant les uns avec les autres. Pour cela, un ensemble de modèles basés sur des expressions d'exigences permet d'obtenir une analyse d'un niveau d'abstraction assez bas. Ainsi une technique de conception plus traditionnelle (orientée objet par exemple) peut être appliquée pour implémenter les agents. Ces modèles sont représentés sur la figure 2.1. Ceux qui nous intéressent ici sont celui du rôle, composé d'un ensemble de rôles et celui d'interaction, composé d'un ensemble de protocoles, tous deux composant le modèle d'organisation.

Un rôle est défini par quatre attributs :

— les responsabilités déterminent la fonctionnalité des agents.
— pour remplir ses responsabilités, un rôle possède un ensemble de permissions. Les permissions limitent les ressources auxquelles le rôle peut avoir accès. Généralement, ces ressources sont des informations que le rôle peut lire, écrire ou créer. Les permissions spécifient donc ce que peut et ce que ne peut pas utiliser le rôle.
— les activités sont des tâches ou des actions qu'un rôle peut exécuter sans qu'il interagisse avec d'autres rôles.

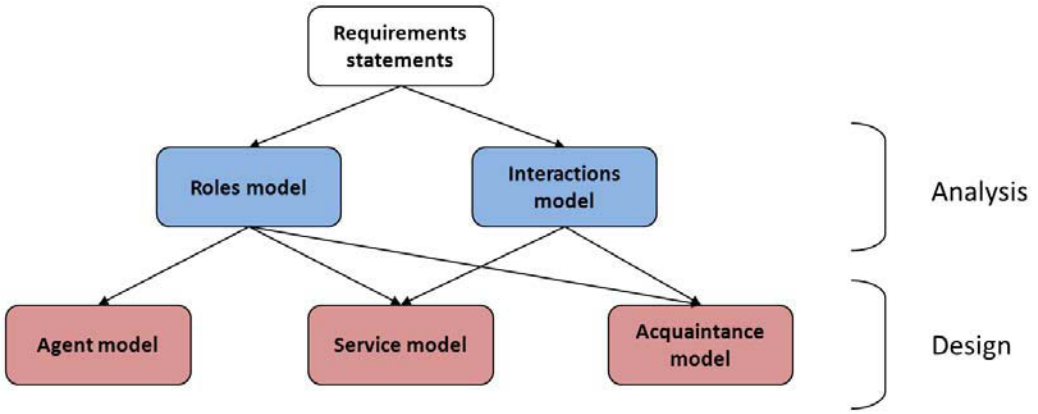


FIGURE 2.1 – Liens entre les modèles de la méthodologie GAIGA

— les protocoles sont des tâches ou des actions qu'un rôle peut exécuter et qui impliquent d'autres rôles. Les protocoles définissent la façon dont un rôle peut interagir avec les autres rôles.

Le modèle d'interactions complète le modèle de rôles. En effet, il contient une définition pour chaque protocole de chaque rôle dans le système. Les protocoles définissent toutes les interactions inter-rôles. Plus précisément, ils décrivent brièvement la nature de l'interaction (son but) en ignorant les détails d'implémentation ainsi que les séquences de messages échangés. La définition des protocoles représente également le rôle initiateur (celui qui démarre l'interaction), le rôle répondeur (celui avec lequel l'initiateur interagit), des informations d'entrée (utilisées par l'initiateur durant l'exécution du protocole) et de sortie (fournies par le répondeur ou pour le répondeur durant le déroulement de l'interaction) et enfin une brève description du traitement que l'initiateur effectue pendant l'exécution de ce protocole.

Comme illustré sur la figure 2.1, ces deux modèles d'analyse (modèle d'interactions et modèle de rôles) sont ensuite raffinés en modèles moins abstraits tels que le modèle agent (les types d'agents qui seront utilisés dans le système), le modèle de services (les fonctionnalités exhibées par les agents) et le modèle d'accointances (les liens de communication entre les agents). Dans [Zambonelli 01], les auteurs de GAIA proposent, en plus des modèles d'origine, un modèle de coordination se situant au même titre que les modèles de rôles et d'interactions au niveau analyse de la méthodologie. Ce modèle de coordination prend en compte la façon d'exprimer la coordination entre rôles au travers d'un medium spécifique. Il définit ainsi des lois sociales afin de diriger toutes les communications à travers

le medium modélisé.

Modèles Agent-Groupe-Rôle (AGR)

Le modèle organisationnel AGR pour Agent, Groupe et Rôle [Ferber 03] est une évolution du modèle Aalaadin [Ferber 98] : la modélisation d'un ensemble d'Agents jouant des Rôles dans des Groupes. Un modèle AGR est séparé en une représentation de niveau Agent instanciant et une représentation de niveau organisationnel. Ces deux niveaux sont représentés sur la figure 2.2 définissant le méta-modèle de AGR. Nous constatons que le niveau organisationnel permet de spécifier des structures de groupe (Group Structure), des Rôles (Role), des contraintes entre Rôles (Constraint) et des interactions entre rôles (Interaction). Le niveau Agent permet d'instancier les Groupes et d'y associer des ensembles d'Agents en y jouant des Rôles.

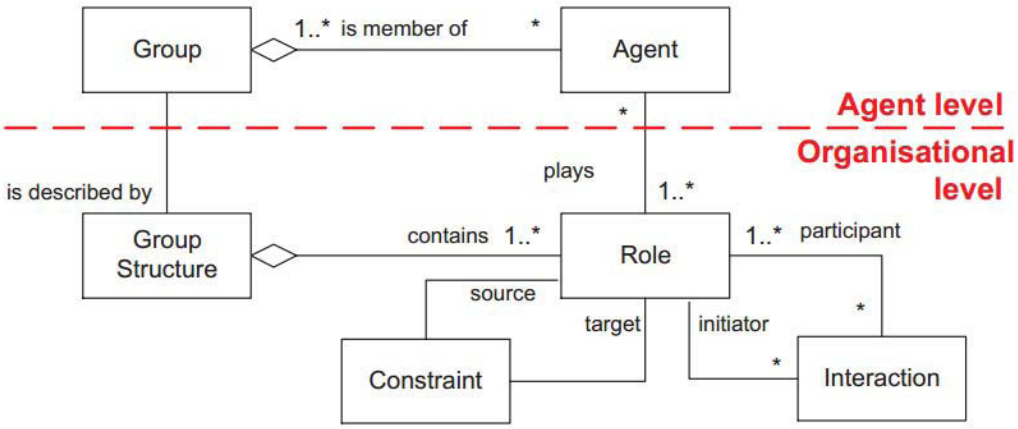
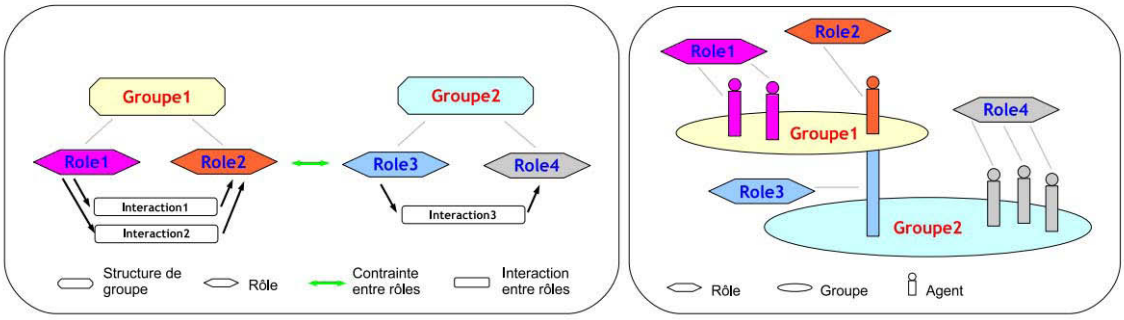


FIGURE 2.2 – Méta-modèle de AGR [Ferber 03]

La Structure d'un Groupe est une description abstraite d'un Groupe composée de Rôles. Les Rôles sont des descriptions a priori du comportement que les Agents auront. Les Rôles décrivent également les contraintes que les Agents jouant ces Rôles doivent satisfaire. Ainsi, une Structure de Groupe permet de diviser l'organisation en contextes d'activités communes (activités de défense pour une équipe de football par exemple). Les Rôles sont locaux aux Structures de Groupe. Ils sont associés entre eux au travers de description d'interactions et de relation de dépendance (le même Agent devra jouer ces deux Rôles). La figure 2.3(a) donne un exemple de représentation graphique du niveau organisationnel d'un modèle

AGR. Nous y retrouvons deux Groupes Groupe1 et Groupe2 constitués respectivement des Rôles : Role1, Role2, Role3 et Role4. Deux relations d'interactions permettent aux Rôles Role1 et Role2 d'interagir selon un protocole bien particulier de même pour les Rôles Role3 et Role4. Enfin, les Rôles Role2 et Role3 sont liés par une contrainte de dépendance obligeant un Agent à jouer ces deux Rôles.

Le modèle agent est la concrétisation du modèle organisationnel. Les Agents sont des entités actives et communicantes pouvant jouer plusieurs Rôles et appartenir à plusieurs Groupes. AGR ne contraint pas les Agents au niveau de leur architecture ou de leurs capacités de traitement de l'information. Les Agents ne peuvent communiquer entre eux que s'ils font partie du même Groupe. Un Rôle ne peut appartenir qu'à un seul Groupe mais peut être joué par plusieurs Agents. La figure 2.3(b) donne un exemple de représentation graphique du niveau agent d'une modélisation AGR. Nous y retrouvons les mêmes Rôles que dans l'exemple précédent, joués par un ensemble d'agents regroupés dans des Groupes. Un Agent respecte la contrainte entre les Rôles Role2 et Role3 en jouant ces deux rôles dans les deux instances de Groupe Groupe1 et Groupe2.



(a) Structure organisationnelle (b) Modèle agent

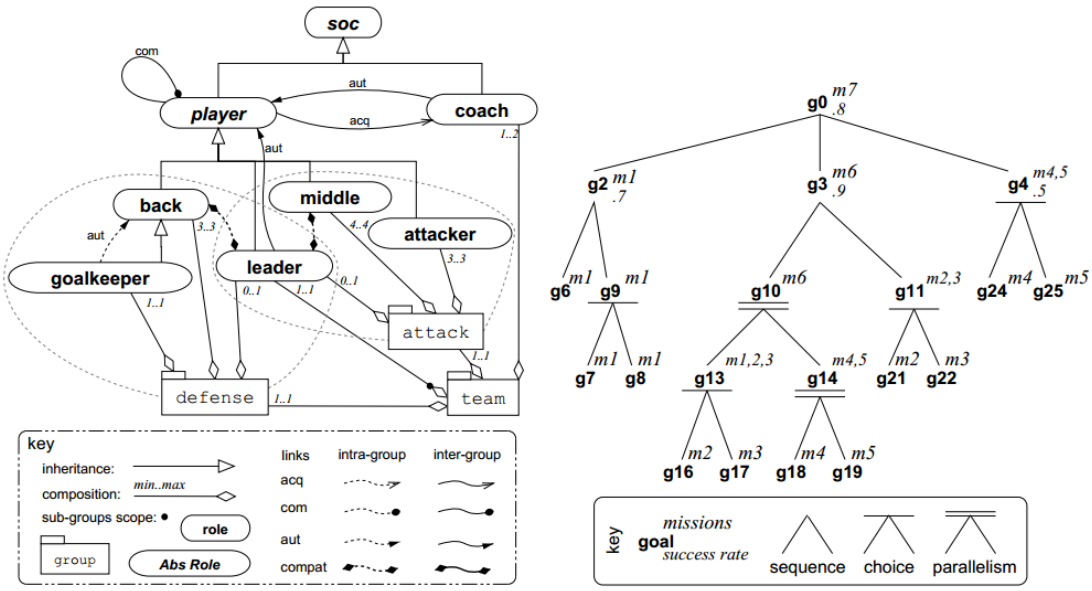
FIGURE 2.3 – Exemple de modélisation avec AGR [Ferber 03].

Modèles $\mathcal{M}$OISE$^+$

$\mathcal{M}$OISE$^+$ (Model of Organization for multI-agent SystEms) [Hübner 03] [Hübner 02] est un modèle organisationnel spécifiant un ensemble de contraintes pour les agents selon trois dimensions : une spécification structurelle, une spécification fonctionnelle et une spécification déontique. La spécification structurelle est construite sur les concepts de rôle, de liens entre rôle et de groupes. Comme illustré sur la figure 2.4(a), une spécification structurelle se représente graphiquement et nous permet de constater que les groupes sont composés de rôles, que ces rôles sont hiérarchisés et qu'ils peuvent être liés entre eux avec quatre types de liens.

Les rôles sont des labels utilisés pour assigner un ensemble de contraintes sur le comportement des agents les jouant. Les liens sont des relations entre deux rôles qui contraignent directement les agents dans leurs interactions avec les autres agents jouant les rôles correspondants. Les groupes sont des ensembles de liens, de rôles et de relations de compatibilité entre rôles.

En ce qui concerne la spécification fonctionnelle, les concepts de bases sont le schéma, le plan, le but et la mission. Comme illustré sur la figure 2.4(b), un schéma social est composé d'un ensemble de buts, de plans et de missions et représente une fonction d'un ou de plusieurs agents au sein de l'organisation. Le schéma a un but racine qui sera le but à atteindre pour exécuter le schéma social. Les buts sont découpés en plans, c'est-à-dire en ensembles de sous-buts qu'il faut atteindre séquentiellement, en parallèle, ou au choix (seulement l'un d'entre eux). Enfin, les missions regroupent les buts en ensembles cohérents au sein d'un même schéma pour être assignées aux rôles.



(a) Spécification Structurelle (b) Spécification Fonctionnelle

FIGURE 2.4 – Exemple d'organisation respectant le modèle $\mathcal{M}$OISE$^+$ [Hübner 03].

La spécification déontique est composée d'un ensemble d'expressions mettant en relation les deux autres spécifications (un rôle et une mission) paramétrées par un opérateur déontique (Obligation, Permission ou Interdiction) et une contrainte temporelle. Ces relations rendent explicite la dimension normative des rôles que joueront les agents en définissant par exemple l'Obligation pour le rôle $R1$ d'accomplir la mission $m1$ avant la date t. Ces expressions contraignent les agents sur

la possibilité de s'engager sur une mission ou d'atteindre un but suivant le rôle qu'ils jouent.

La spécification de ces trois dimensions forme une spécification d'organisation (SO) tandis que l'instanciation des trois dimensions à l'aide d'agents jouant les rôles et donc contraints par l'organisation représente une entité organisationnelle (EO). A un instant t de l'évolution de l'EO, chaque agent se voit contraint d'accomplir ou de ne pas accomplir une mission.

Le modèle organisationnel $\mathcal{M}$OISE^{Inst}

Le modèle organisationnel $\mathcal{M}$OISE^{Inst} [Gateau 07] est basé sur le modèle $\mathcal{M}$OISE^{+} et l'étend. En effet, les expressions déontiques utilisées dans $\mathcal{M}$OISE^{+} sont trop restrictives vis à vis de la définition des rôles qu'on cherche à accomplir. Pour pallier à ce problème, plus d'informations sont ajouté à une nouvelle spécification, transformée en un ensemble de normes, elle s'appelle la spécification normative. En plus de cette définition de l'organisation via des spécifications, dans ce modèle il y a plusieurs niveaux d'organisation (individuel, social et collectif). $\mathcal{M}$OISE^{Inst} est alors composé de quatre spécifications : la Structurelle, Contextuelle, Fonctionnelle, et Normative (cf. figure 2.5) :

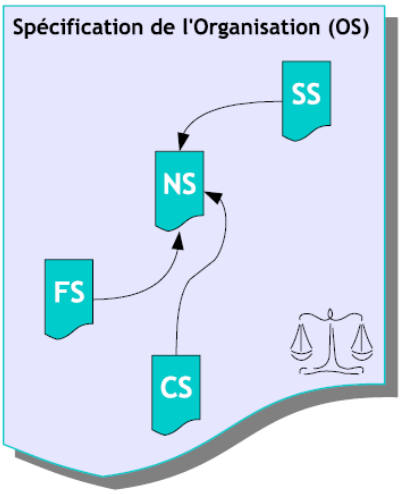


FIGURE 2.5 – Spécification de l'organisation de modèle organisationnel $\mathcal{M}$OISE^{Inst}.

La spécification structurelle (SS) : définition des rôles (niveau individuel) des agents, ainsi que les relations entre ces rôles (niveau social) et les groupes

d'agents (niveau collectif). Elle peut être décrite en trois composantes : les entités structurelles, les relations entre ces entités structurelles, les contraintes de cardinalité.

La spécification fonctionnelle (SF) : établissement d'objectifs **"goals"** à atteindre (niveau individuel) par l'organisation. Ils sont regroupées dans des tâches comprises dans un modèle de fonctionnement social à suivre. Elle exprime le processus social que les agents doivent suivre pour atteindre l'objectif collectif représenté par l'objet racine du système.

La spécification contextuelle (SC) : définit les différents contextes dans lesquels l'organisation va évoluer, et les transitions d'un contexte à l'autre. C'est donc l'ensemble des contextes dans lesquels elle peut évoluer ainsi que tous les événements déclencheurs du changement de contexte.

La spécification normative (SN) : définit les relations déontiques reliant les trois spécifications indépendantes citées précédemment (SS, SF, SC). Cette spécification énonce clairement les droits et les devoirs de chacun des rôles, de la spécification structurelle, par rapport aux missions (ensemble de goals) de la spécification fonctionnelle dans un cas spécifique de fonctionnement appelé scène (ensemble de contextes) de la spécification contextuelle.

Ce modèle a fait ses preuves dans le centre de recherche public Henri Tudor au Luxembourg car il représente une approche très performante dans l'organisation et le contrôle des systèmes multi-agents, cependant il n'intègre pas l'aspect dynamique de l'organisation des agents. Plusieurs travaux dessus sont en cours pour améliorer ses performances notamment sa dynamicité.

2.3.2 Modèles d'institution

Modèle OMNI

La plate-forme OMNI (Organizational Model for Normative Institutions) [Virginia 04] [Virginia 05] couvre d'après Virginia Dignum et Javier Vàzquez-Salceda tous les aspects qu'un SMA doit pouvoir fournir. Ces aspects sont :

— pouvoir modéliser des systèmes complexes en définissant l'organisation dans laquelle l'agent évolue.
— pouvoir définir et détecter les bons et mauvais comportements des agents dans le but d'instaurer un environnement de confiance entre les agents.
— l'organisation doit pouvoir représenter tout ce qui permettrait de modéliser le contexte dans lequel un agent peut interagir.

Le modèle OMNI est basé sur les modèles OperA de Virginia Dignum [Gateau 04] apportant la dimension organisationnelle de OMNI et HarmonIA de Javier Vàzquez-Salceda apportant la dimension normative. Le modèle OMNI peut

se représenter comme une matrice avec d'un côté un découpage en dimensions et de l'autre un découpage en niveaux. Nous avons ainsi les dimensions Normative, Organisationnelle et Ontologique ainsi que les niveaux Abstraits, Concrets et d'Implémentation.

Le but de OMNI est finalement de définir de manière précise les contextes au sein desquels les agents devront interagir. Ainsi, les rôles, les scènes et les interactions sont vus comme des normes, c'est à dire que pour chaque concept, des normes sont incluses dans leur définition. Le concept de rôle permet de structurer une organisation de 1 à N agents.

Modèle ISLANDER

ISLANDER [Marc 02] se définit comme un IDL (Institution Definition Language). Ce langage basé sur XML est une syntaxe pour la définition d'Institutions électroniques représentées comme un système dialogique permettant l'échange de messages. Ces interactions sont structurées au travers de regroupement d'agents appelés scènes et qui suivent des protocoles explicitement définis. Une Institution Électronique est vue par Marc Esteva comme étant composée d'un cadre d'exécution dialogique (Dialogic Frameworw, DF), d'une Structure Performative (SP) et de normes. Les normes gouvernent les interactions entre les agents et sont exprimées par une représentation en langage informatisable.

La division de toutes les interactions possibles entre les agents en scènes permet à ISLANDER d'avoir une conception modulaire concrète du système, apportant ainsi de la flexibilité et de la lisibilité aux modèles. Cependant, en comparaison avec $\mathcal{M}\text{OISE}^{Inst}$, la spécification structurelle en hiérarchie de rôles est minimale dans le sens où on peut seulement définir des rôles et des sous-rôles ainsi que des liens de compatibilité entre rôles.

2.4 Conclusion et positionnement

Conclusion

Dans ce chapitre nous nous sommes intéressé aux systèmes multi-agents, nous les avons d'abord définis. Ensuite, nous avons vu la relation entre ces systèmes multi-agents et la robotique mobile via les différentes approches utilisées.

Comme on a pu le remarquer dans notre investigation de l'état de l'art, les approches issues du domaine de l'informatique influencent de plus en plus celles des roboticiens, venant du fait que les roboticiens ont besoin d'expérimenter d'autres

approches qui permettent d'avoir de meilleurs résultats, notamment grâce à la force et l'efficacité des SMA notamment au niveau de la coordination des robots mobiles. La combinaison de ces deux domaines commence à contribuer à l'élaboration de différents exemples intéressants d'architectures pour des systèmes multi-robots, où les agents décident : quand et quoi communiquer, comment arriver à un accord commun avec les autres, et enfin comment répartir les tâches aux différents robots. Un des intérêts de cette thèse est d'étudier les différents types de coordination existants basées sur les agents et le mettre à profit de systèmes multi-robots mobiles.

Une partie de ce chapitre a été consacrée à la présentation des différentes manières d'organiser un système multi-agents. Remarquant l'émergence des modèles organisationnels dans ce domaine et de leurs efficacité, et voulant tester et valider une nouvelle approche d'organisation d'un système multi-robots grâce à ces modèles, on a donc parcouru les différents modèles organisationnels existants dans la littérature, en analysant leurs points forts et points faibles ainsi que leurs différents domaines d'application.

Vu la contraintes imposées sur les SMR dans cette thèse, nous avons besoin d'une organisation qui nous permet de changer de mode de fonctionnement en fonction du type de mission voulu, ce qui représente un besoin d'organisation, de coordination et de coopération assez élevé. À l'issue de cette analyse, nous avons abouti au choix du modèle $\mathcal{M}\text{OISE}^{Inst}$ comme modèle à exploiter dans notre architecture de contrôle qui sera mis au cœur de l'organisation d'un système multi-robots mobiles. En effet, grâce à la spécification normative nous pouvons aisément adapter le mode de fonctionnement très facilement, ce qui permet de contrôler avec flexibilité et efficacité un groupe d'agents/robots. En plus de ça, $\mathcal{M}\text{OISE}^{Inst}$ est considéré comme un des meilleurs modèles organisationnel qui existe à ce jour (du point de vue organisation normative).

Positionnement de l'architecture de contrôle proposée

L'objectif de cette thèse est de réaliser une architecture de contrôle permettant une meilleure gestion d'un système multi-robots mobiles se déplaçant dans un environnement semi-inconnu, et que cette architecture soit la plus générique et flexible possible. En d'autres termes, nous souhaitons qu'elle soit simple d'utilisation, insensible au nombre de robots présents dans le système et valable y compris dans des environnements encombrés et dynamiques. Elle doit être aussi ouverte et évolutive que possible dans le but d'accomplir des tâches plus complexes sans pour autant être complètement reformalisées.

Considérant les caractéristiques désirées, citées ci-dessus, on s'aperçoit que confier le contrôle et la gestion du SMR à un superviseur central devient ra-

pidement une tâche inextricable pour ce dernier, notamment dans le cas d'une augmentation du nombre de robots mobiles à manipuler. Il est alors judicieux voire indispensable de doter chaque entité (agent/robot) de ses propres outils décisionnels.

Par ailleurs, un contrôle cognitif se base sur la planification de l'action à accomplir, ce qui nécessite de modéliser l'environnement et de minimiser un certain coût (e.g. puissance ou énergie). Modéliser un environnement hautement dynamique peut s'avérer difficile voire impossible. Les coûts minimisés prennent en considération plusieurs facteurs : distance parcourue, temps nécessaire, risque de collision, etc. La tâche de quantifier ces critères est tout aussi difficile dans ce type d'environnement du fait de ses changements fréquents. Pour y remédier, la solution consiste à re-planifier l'action en prenant en considération le nouvel environnement, voire en anticipant les actions des éléments dynamiques. C'est pourquoi nous avons opté pour le rajout d'un contrôle réactif pour établir une partie de la stratégie du contrôle.

Considérant tout ce qu'on vient de citer plus haut, nous avons fait le choix ambitieux de développer une architecture de contrôle ouverte et englobant un grand nombre de types de traitements et de contrôle multi-robots. En effet, l'architecture de contrôle proposée est hybride, et doit pouvoir représenter le comportement des deux types d'architectures existants dans la littérature, ainsi que les deux types de contrôle multi-robots selon le besoin, à savoir :

— *Comportement Centralisé* : englobant seulement le contrôle cognitif.
— *Comportement Distribué* : englobant seulement le contrôle réactif.
— *Comportement Centralisé/Distribué* : englobant le contrôle cognitif et réactif, c'est un mix des deux.

Pour ce faire, l'architecture de contrôle proposée se base principalement sur deux stratégies que sont :

— la hiérarchisation des couches en plusieurs niveaux de traitement de l'information.
— l'utilisation et l'adaptation de modèles organisationnels multi-agents dans un contexte multi-robots.

Ces modèles ont prouvé leur efficacité dans le contrôle et la coordination des systèmes multi-agents, ce qui engendre un deuxième défi qui est de réaliser cette tâche pour des robots mobiles, et donc trouver la meilleure façon de combiner ces deux stratégies pour tirer profit de leurs avantages, et contrôler notre système avec plus de fiabilité. Notre architecture est composée alors de trois niveaux : le niveau physique, le niveau contrôle/commande et le niveau organisation. Elle est présentée ainsi que ses différentes composantes dans la partie suivante.

Chapitre 3

Le contrôle/commande dans l'architecture hybride proposée

Dans ce chapitre nous allons introduire l'architecture de contrôle/commande proposée. Après une description générale des approches de coordination entre robots et de contrôle/commande adoptés, les différents éléments de l'architecture sont détaillés individuellement. S'inspirant d'une approche hybride (cf. Chapitre 1, Section 1.4), notre architecture est composée principalement de trois niveaux : *le niveau organisation*, *le niveau contrôle/commande* et *le niveau physique*.

Ainsi, nous présenterons une architecture ouverte, pouvant s'exécuter de plusieurs façons différentes, allant du mode complètement centralisé, centralisé/distribué ou totalement distribué. Avec divers types de comportements : cognitif, cognitif/réactif, ou réactif seul. La mise en place de ces modifications de comportement se fait d'une manière très flexible, permettant aux robots d'adapter leurs comportements à la criticité de la situation ou au type d'environnement.

3.1 Principe général

Considérons un SMR à N entités qui peuvent avoir des configurations initiales différentes dans l'environnement et dont le but est d'atteindre M destinations différentes (avec $M \leq N$), dans un environnement encombré. Les robots doivent aussi pouvoir éviter les obstacles (statiques et dynamiques) existants dans l'environnement, ainsi que les collisions entre eux-mêmes. On peut prendre comme exemple de tâche multi-robots : un espace de stockage où chaque robot du SMR doit déplacer une charge d'un point A à un autre point B.

La tâche générique investiguée dans cette thèse n'est autre qu'une tâche de coordination et planification multi-robots qui peut être accomplie suivant divers scénarios, nous en citerons trois scénarios avec des niveaux ascendants de complexité qui nous intéressent plus particulièrement :

- *Scénario 1* : un SMR avec N robots et N destinations, la tâche consiste à coordonner les plans des N robots pour atteindre leurs cibles respectives sans conflits entre eux, tout en évitant les obstacles statiques et dynamiques présents dans l'environnement.
- *Scénario 2* : un SMR avec N robots et N destination, mais tout en maintenant une formation désirée, on doit donc coordonner les robots pour évoluer en formation dans l'environnement tout en évitant les conflits entre eux, ainsi que les obstacles statiques et dynamiques présents.
- *Scénario 3* : un SMR avec N robots et M destinations (avec $M \leq N$), dans ce scénario notre groupe de robots est divisé en deux catégories : un groupe qui a le même objectif et la même destination, qui est d'aller vers une cible en maintenant une formation donnée ; tandis que le deuxième groupe est composé de *1* ou plusieurs robots, qui ont chacun une destination différente. L'objectif est donc de coordonner en même temps les deux groupes de robots pour que les missions de chaque élément du SMR soient accomplies. En d'autres termes, faire en sorte de coordonner les mouvements de tous les robots, pour éviter les conflits entre eux, tout en maintenant la formation pour les robots concernés. Ainsi, garantir la sécurité des robots dans l'environnement en évitant les obstacles statiques et dynamiques présents. L'architecture aura donc un comportement de systèmes hétérogènes qui intègrent différents types de traitements cognitifs et réactifs.

Nous aborderons la résolution de la tâche de coordination par l'utilisation des systèmes multi-agents (cf. section 2), et plus particulièrement l'organisation du haut niveau de l'architecture par le modèle organisationnel multi-agents $\mathcal{M}\text{OISE}^{Inst}$ (cf. section 2.3.1).

Avant de parler de coordination, il est impératif de savoir comment contrôler/commander chaque élément du SMR, nous nous intéresserons donc dans les

prochaines sections à une seule entité robotique pour comprendre comment réaliser son contrôle.

3.2 Modèle robotique utilisé

Le problème du contrôle d'un robot mobile est grandement lié à ses caractéristiques cinématiques et dynamiques. En effet, la notion de robot mobile est vaste et inclut différents types : les robots à roues, à pattes (robots marcheurs), à hélice, robots sous-marins, etc. Cependant, ces robots diffèrent clairement par leurs caractéristiques physiques et structurelles ce qui génère une formulation différente du problème de contrôle pour chaque type.

Dans cette thèse, nous nous focaliserons sur les robots mobiles à roues qui nous serviront pour expérimenter l'architecture de contrôle proposée.

3.2.1 Robots mobiles à roues

FIGURE 3.1 – Exemple de deux robots mobiles holonomes : (a) utilisant des roues à galets [Asama 95] et (b) ayant des roues sphériques [Tomsguide 12].

Les robots à roues sont les premiers robots mobiles utilisés et sont les plus étudiés, sans doute car la roue est l'un des premiers mécanismes de locomotion créée par l'homme et reste largement suffisant pour les déplacements des robots. Les caractéristiques des roues d'un robot mobile (forme, angle de braquage, rayon, etc.) définissent ses caractéristiques et les degrés de liberté de sa mobilité. Ainsi, on trouve les robots dits omnidirectionnels (ou plus connus chez les roboticiens sous le terme de robots mobiles holonomes). Ce type de robots peut se déplacer

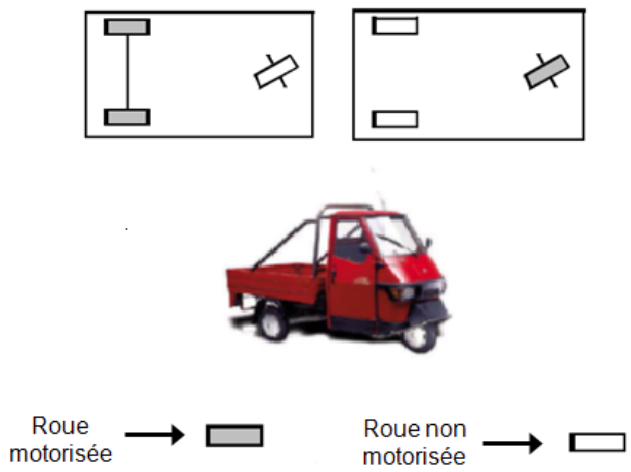


FIGURE 3.2 – Représentation de deux types de tricycles : avec une roue de direction centrée motorisée, et avec une roue de direction centrée et un train arrière motorisé.

instantanément dans n'importe quelle direction indépendamment de son orientation. Dans cette catégorie s'inscrivent les robots à roues de formes sphériques ou à galets pouvant tourner librement [Higashimori 11] (cf. Figure 3.1).

Contrairement à ce type de robots, on trouve des robots non-holonomes plus communs dans la communauté robotique. Ils perdent un degré de liberté correspondant à la translation instantanée dans une direction latérale. Il y a principalement trois types de robots mobiles non-holonomes :

— le robot unicycle : il est actionné par deux roues dont les moteurs sont indépendants. Il peut contenir d'autres roues folles dont le but est uniquement d'assurer sa stabilité horizontale. Les robots Khepera III (cf. Figure 3.4) que nous utilisons dans nos travaux sont des robots unicycle, c'est la raison pour laquelle nous allons nous intéresser au modèle cinématique de ces robots.
— le robot tricycle : il est constitué de deux roues fixes de même axe lui conférant la vitesse linéaire et d'une roue centrée orientable placée sur son axe longitudinal. L'angle de braquage de cette roue définit le centre de rotation du robot (cf. figure 3.2).
— le robot de type voiture : similaire au tricycle, il diffère au niveau qui contient deux roues plutôt qu'une seule assurant ainsi une meilleure sta-

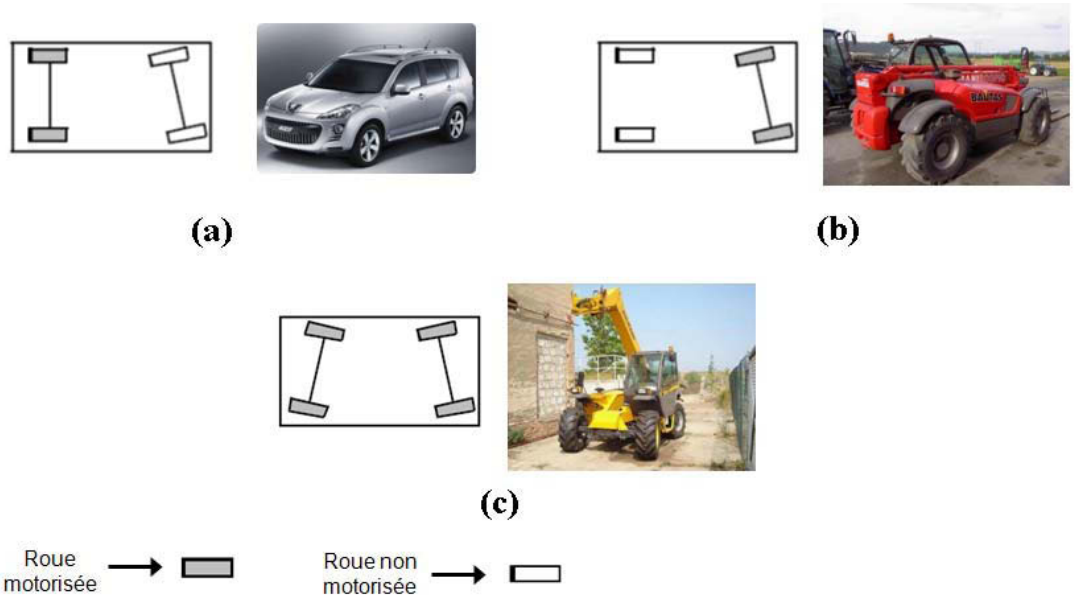


FIGURE 3.3 – "a", "b" et "c" représentent les catégories des robots de type voiture, qui diffèrent par la localisation des roues motorisées dans chaque véhicule, et la mobilité ou non du train arrière.

bilité. Les robots autonomes utilisés pour la navigation notamment en convoi, ou les robots de chantier illustrent ce type de robots (cf. Figure 3.3).

3.2.2 Modèle cinématique d'un robot unicycle

Pour modéliser le déplacement d'un robot mobile en tenant compte des contraintes de roulement sans glissement de ses roues, un repère convenable doit être judicieusement choisi. Ainsi, afin de spécifier sa position, il est possible de considérer le robot comme un objet rigide se déplaçant sur un plan horizontal avec deux degrés de liberté : le premier correspond à un mouvement longitudinal (déplacement à l'avant ou à l'arrière) tandis que le deuxième est celui de la rotation autour d'un axe vertical. Un repère relatif (O_r, X_r, Y_r) est associé à cet objet (cf. Figure 3.5). L'origine O_r correspond au centre du robot, la position du robot et son orientation sont définies en exprimant la relation entre le repère relatif et un repère absolu (O_a, X_a, Y_a). Ainsi, la position d'un robot mobile i du SMR est spécifiée par (x_i, y_i) correspondant aux coordonnées du point O_{r_i} dans le repère(O_a, X_a, Y_a). Son orientation θ_i est l'angle compris entre l'axe X_{r_i} et X_a.

FIGURE 3.4 – Le robot Khepera III [K-Team 03], et le robot ALICE [Caprari 00] développés à l'EPFL.

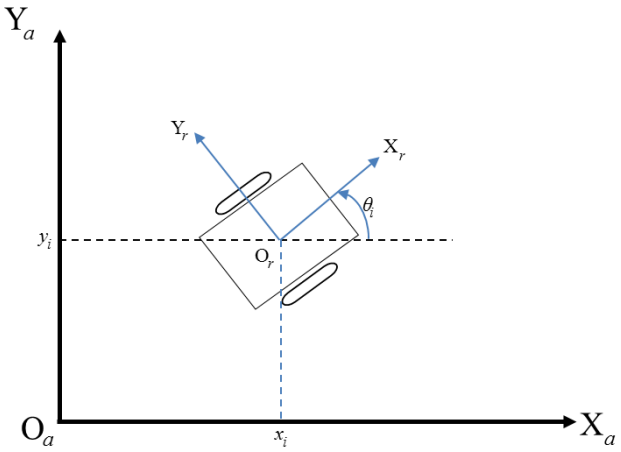


FIGURE 3.5 – Représentation de la pose d'un robot unicycle dans un repère global et relatif lié au robot.

$$O_{r_i} = \begin{pmatrix} x_i \\ y_i \\ \theta_i \end{pmatrix} \tag{3.1}$$

Le déplacement du robot peut alors être exprimé par le système d'équation suivant :

$$\dot{x}_i = v_i.\cos(\theta_i) - u_i.\sin(\theta_i) \tag{3.2a}$$
$$\dot{y}_i = v_i.\sin(\theta_i) + u_i.\cos(\theta_i) \tag{3.2b}$$
$$\dot{\theta}_i = \omega_i \tag{3.2c}$$

Avec : v_i, u_i, sont les composantes de la vitesse linéaire du robot i exprimées dans le repère (O_r, X_r, Y_r) et ω_i est sa vitesse angulaire. On peut alors exprimer le système précédent sous une forme matricielle :

$$\dot{O}_{r_i} = R(\theta) \begin{pmatrix} v_i \\ u_i \\ \omega_i \end{pmatrix} \tag{3.3}$$

La matrice $R(\theta)$ est la matrice de rotation permettant d'exprimer la variation des positions du robot dans le repère (O_a, X_a, Y_a) et est définie comme suit :

$$R(\theta) = \begin{pmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 \\ \sin(\theta_i) & \cos(\theta_i) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.4}$$

Cependant, la contrainte non-holonome impose que le déplacement linéaire du robot se fait uniquement selon l'axe X_r relatif. Ainsi, la composante u de la vitesse linéaire correspondant au déplacement latéral sur l'axe Y_r est nulle. Les équations 3.2 se réduisent alors à

$$\dot{x}_i = v_i.\cos(\theta_i) \tag{3.5a}$$
$$\dot{y}_i = v_i.\sin(\theta_i) \tag{3.5b}$$
$$\dot{\theta}_i = \omega_i \tag{3.5c}$$

Dans le cas du robot unicycle, où les deux roues sont contrôlées par deux moteurs séparément, et en l'absence de glissement, les vitesses linéaire et angulaire se calculent par

$$\begin{pmatrix} v_i \\ \omega_i \end{pmatrix} = \begin{pmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{l} & \frac{-r}{l} \end{pmatrix} \begin{pmatrix} \omega_{d_i} \\ \omega_{g_i} \end{pmatrix} \tag{3.6}$$

où ω_{d_i} et ω_{g_i} sont les vitesses angulaires des roues droite et gauche respectivement et r est le rayon de ses roues. l représente la distance entre les roues (cf. 3.5).

Généralement, on définit le contrôle nécessaire du robot en établissant les vitesses linéaire et angulaire v et ω. Le contrôle des roues tiendra donc compte du modèle décrit par le système d'équations 3.6.

3.3 Structure de l'architecture de contrôle hybride proposée

L'élaboration d'une architecture innovante pour contrôler ou commander un groupe de robots autonomes dans des environnements non structurés, nécessite la prise en compte de différents aspects, c'est pour cela que l'architecture de contrôle proposée contient trois niveaux :

- *le niveau physique :* il représente tout ce qui est parties physique dans les robots, commençant par les actionneurs associés aux roues ainsi que les différents capteurs présents sur les robots, et finissant par toutes les composantes physiques qui constituent les robots. Ce niveau communique avec le niveau contrôle, par envoi de données à travers ses capteurs, et par réception de commande appliquées aux actionneurs. Le bon fonctionnement de ce niveau est primordial au bon déroulement des divers missions que les robots auront à exécuter.
- *le niveau contrôle :* est constitué par les différents contrôleurs (bas niveau) destinés à contrôler la navigation du robot, soit les contrôleurs de : *suivie de trajectoire, évitement réactif d'obstacles, attraction vers une cible (statique ou dynamique)*, et enfin *le maintien de formation.*
- *le niveau organisation :* est dédié à des objectifs plus complexes, qui sont la coordination de la navigation du *SMR*, la gestion des différents modes de fonctionnement (planification/re-planification, navigation en formation, etc.).

L'architecture de contrôle proposée est représentée sur la figure 3.6. Le niveau organisation sera détaillé dans le chapitre 4, nous nous focaliserons dans la suite de ce chapitre sur le niveau contrôle/commande et ses composantes.

Le niveau de contrôle/commande repris sur la figure 3.7 comme nous pouvons le constater, en plus des modules de planification/re-planification de trajectoire, de maintien de formation, sélection d'action hiérarchique et de communication avec le niveau organisation, le niveau de contrôle est composé par différents contrôleurs :

- *Suivi de Trajectoire.*
- *Attraction vers une cible.*
- *Évitement réactif d'obstacle.*

Nous commencerons par définir les deux blocs *module de planification/re-planification*, et *module de communication* entre les niveaux organisation et contrôle/commande :

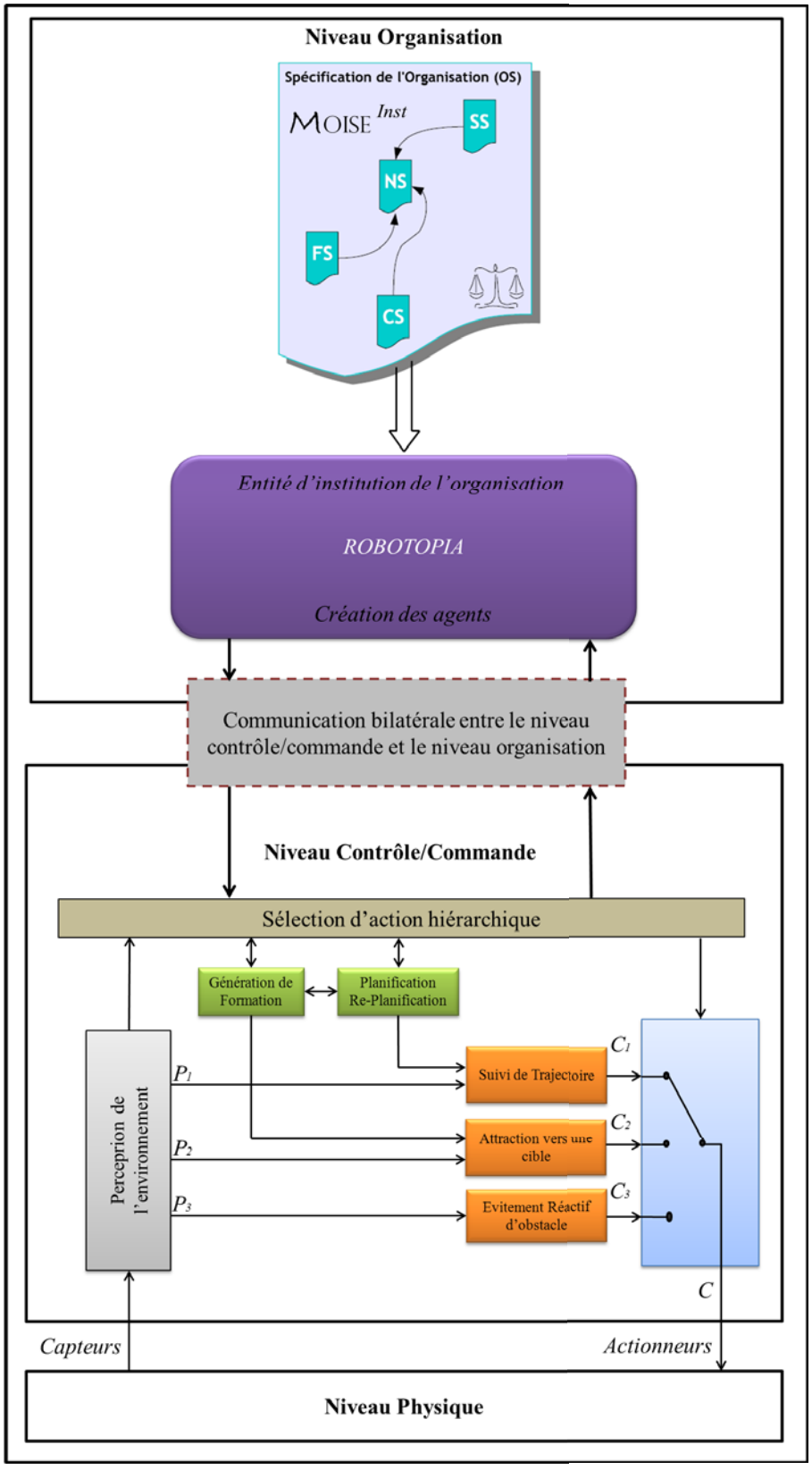


FIGURE 3.6 – Représentation l'architecture de contrôle proposée "Architecture de Contrôle/COmmande dédiée aux systèmes Distribués Autonomes"ACO^2DA.

a) le module de planification : est appelé à chaque fois qu'il est nécessaire de faire une planification de trajectoire, à n'importe quel moment de la navigation, et par n'importe quelle entité du *SMR*. Recevant comme entrée une posture de départ et une d'arrivée (les configurations sont définis par (x_i, y_i, θ_i), où x_i et y_i sont les coordonnées du robot et θ_i son orientation), et fournissant une trajectoire composée de configurations permettant au robot d'atteindre la posture finale souhaitée tout en s'éloignant des obstacles présents dans l'environnement.

b) le module de communication : dans le contexte des systèmes multi-robots contrôlés à travers un système multi-agents, la structure la plus appropriée est celle où il y a une communication explicite avec d'autres agents/robots, par diffusion de messages intentionnels. Ainsi, dans notre modèle nous l'utilisons avec une interface de communication à l'écoute sur tous les robots. Grâce à cela, un agent au niveau *organisation* peut se connecter à son niveau de contrôle associé, à travers ces messages intentionnels.

Le développement des contrôleurs élémentaires ne représente pas les contributions principales de cette thèse, néanmoins il faut les choisir au mieux en fonction du cahier des charges ainsi que des tâches à réaliser. Nous allons dans ce qui suit décrire les détails de chacun des contrôleurs existant dans le niveau de contrôle/-commande, ainsi que son rôle exact au sein de l'architecture de contrôle proposée.

3.3.1 Évitement d'obstacles

La caractéristique de mobilité des robots impose aux robots de pouvoir naviguer dans des environnements encombrés voire dynamiques. L'évitement d'obstacles est donc un comportement de base devant être présent dans tous les robots mobiles. Ainsi, quasiment tous les robots mobiles sont équipés de capteurs (télémètre laser, ultrasons, infrarouges, etc.) permettant de détecter les éventuels obstacles à proximité. La littérature regorge de travaux sur l'évitement d'obstacles dont l'objectif est de garantir le maximum de sécurité des robots mobiles en cours de navigation.

Parmi les méthodes d'évitement d'obstacles réactives (basées sur la perception locale de l'environnement), la méthode des champs de potentiel est sans doute la plus répandue dans la littérature. Initialement proposée par Khatib [Khatib 86] elle considère le robot mobile comme une particule soumise à divers champs de forces régissant son mouvement. Deux principaux champs de potentiel sont considérés (cf. figure 3.8) :

— un champ de potentiel attractif provenant de la position finale à atteindre.

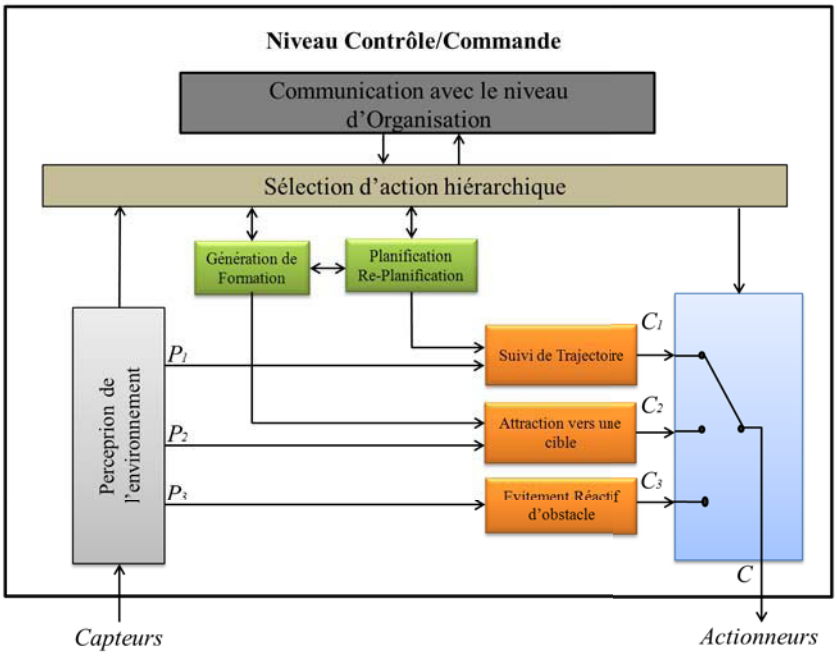


FIGURE 3.7 – Niveau de contrôle/commande.

— un champ de potentiel répulsif provenant des obstacles statiques et mobiles de l'environnement.

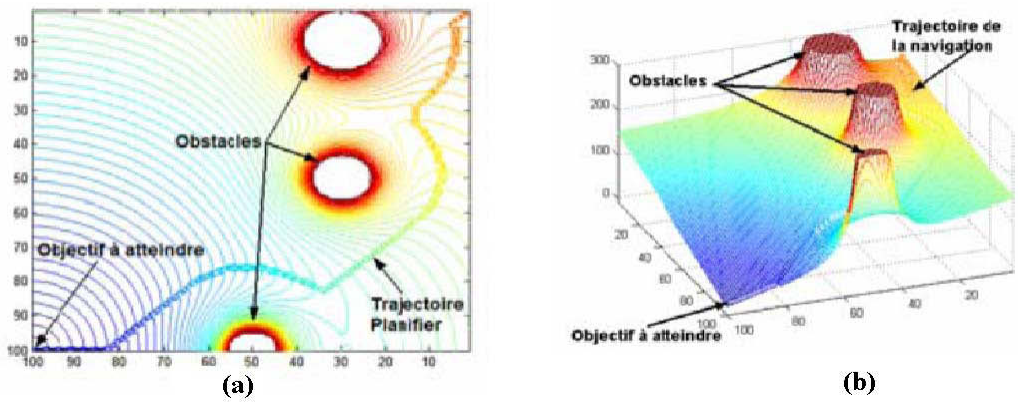


FIGURE 3.8 – Calcul d'un chemin entre un point de départ et un point d'arrivée par la méthode des champs de potentiel. (a) Représentation de la trajectoire générée. (b) Espace de configuration du robot en 3D et représentation des champs répulsifs des obstacles, et Champs attractifs générés par la position finale.

Initialement conçus pour le calcul des mouvements de bras manipulateurs, cette méthode dispose de l'avantage de calculer ces champs de potentiel dans l'espace de travail [1], définissant ainsi une direction privilégiée à suivre par l'organe terminal du bras robotisé. Une modification de la configuration du robot dans son espace articulaire est alors déduite de ce champ dans un second temps.

Bien que simple et élégante, cette méthode possède des inconvénients mis en évidence par Koren et Borenstein [Koren 91]. D'une part, cette approche est sujette à des minima locaux, par conséquence, la convergence vers le but n'est pas assurée. D'autre part, ces potentiels peuvent donner lieu à de fortes oscillations du mouvement du robot en présence d'obstacles et principalement lorsque celui-ci navigue dans des passages étroits (couloirs, portes), et d'une autre part comme tout obstacle dans l'environnement génère des forces virtuelles, ces forces vont affecter tous les robots se trouvant dans son voisinage. Ainsi, certains obstacles non gênants en réalité, affecteront la direction et la vitesse de ces robots. Malgré ces limitations, de nombreuses techniques de navigation ont découlé de ces champs de potentiel. Ils ont par exemple été adaptés à la navigation en présence d'obstacles mobiles dans [Ge 02], [Huang 08] en prenant en compte non seulement la distance aux obstacles mais également la vitesse de ces derniers pour calculer les champs répulsifs.

Dans le même sens des approches par champs de potentiel, sont apparus les histogrammes par champs de vecteur (Vector Field Histogram - VFH). Ceux-ci, introduits par Koren et Borenstein [Koren 91] sont nés de la combinaison des champs de potentiels et des grilles d'occupation : un histogramme basé sur une grille cartésienne de l'environnement est construit et mis à jour au fur et à mesure de la navigation afin de reporter la présence d'obstacles à proximité du robot, et choisit une direction à suivre en discrétisant les différentes directions possibles autour du robot.

On trouve aussi l'approche de navigation par diagrammes de proximité (Nearness Diagram Navigation - ND) proposée par Minguez et Montano [Minguez 00] fortement inspirée des VFH et se basant sur la topologie de l'environnement pour choisir un contrôle à appliquer à chaque instant. Pour ce faire, deux diagrammes polaires sont construits : l'un représente la distance du centre du robot aux obstacles dans toutes les directions autour de celui-ci ; le second calcul cette même distance à partir des contours du robot afin d'estimer la distance de sécurité à conserver. L'étude des discontinuités de ces diagrammes de proximité permet de caractériser les "vallées" libres d'obstacles autour du robot.

À l'opposé des méthodes dîtes directionnelles présentées précédemment, on trouve celles appelées méthodes de l'espace des vitesses. Nous en citons celle nom-

1. L'espace de travail représente ici l'espace opérationnel du robot.

mée Curvature Velocity Method [Simmons 96], Lane Curvature Method [Ko 98], dynamic window [Fox 97], [Ogren 05], etc. Le principe général de ces méthodes est de sélectionner un couple de vitesses linéaire et angulaire (v, ω) qui répond à différentes contraintes dont celle de l'évitement d'obstacles. Un tel couple de vitesses produit une trajectoire pour laquelle la satisfaction des différentes contraintes est évaluée. A l'issue de l'évaluation de tous les couples de vitesses possibles, le plus pertinent par rapport à différents critères est choisi. Cependant, comme les couples de vitesse prennent en considération plusieurs contraintes (éviter l'obstacle, atteindre le point désiré, etc.), il devient difficile de prédire le comportement global du robot et s'il accomplit tous les objectifs correctement.

On trouve aussi des méthodes d'évitement d'obstacles basées sur des concepts de trajectoires décrites par des équations différentielles [Stuart 96]. Parmi elles, on trouve celle du cycle-limite [Kim 03] [Adouane 09]. En plus de l'évitement d'obstacles, elle permet de définir une distance que le robot garde avec l'obstacle durant cette phase, ce qui offre l'assurance de ne pas trop s'en approcher. Aussi, connaissant la position de sa cible, le robot peut choisir le coté d'évitement optimal lui permettant de l'atteindre plus rapidement (cf. annexe A).

Dans l'architecture de contrôle proposée, notre système multi-robots doit aussi être capable de gérer les risques de collision avec des obstacles en mouvement (dynamiques). Ces obstacles dynamiques peuvent être aussi bien d'autres robots du *SMR* que des passants ou des véhicules étrangers au *SMR*. On a pu voir la diversité des méthodes d'évitement d'obstacles existantes dans la littérature ce qui nous a permis de choisir celle qui va convenir le mieux à nos besoins. Malgré ses limitations, la méthode des champs de potentiels reste la mieux adaptée à notre cahier des charges, grâce à son universalité, elle peut être appliquée dans la planification de trajectoires, dans l'évitement d'obstacles statiques et dynamiques et aussi dans l'attraction vers une cible.

Le contrôleur d'évitement d'obstacles utilisé dans nos travaux consiste en l'exécution en temps réel de l'algorithme de planification par champs de potentiels pour calculer à chaque instant la prochaine position à atteindre par le robot en tenant compte des obstacles présents dans l'environnement.

3.3.2 Attraction vers une cible

Dans l'architecture proposée, on aborde le problème de l'attraction vers une cible dans deux cas différents : l'attraction vers une cible statique, et dynamique. Pour ce faire, on a choisi d'appliquer le même contrôleur pour accomplir ces deux tâches.

La problématique de l'attraction ou le suivi d'une cible dynamique a été étudiée dans la littérature, et utilisée dans divers applications comme le suivi de ballon par des robots footballeurs [2] (cf. figure 3.9), ou encore dans l'usage militaire comme le suivi d'un objet dynamique dans une zone de surveillance dans le but de l'identifier ou encore l'arrêter [Schulz 01]. Cette tâche devient très vite compliquée, spécialement quand la dynamique de la cible est inconnue ou difficile à prédire.

FIGURE 3.9 – Robots footballeurs attirés par leur cible "la balle".

Plusieurs travaux ont été faits pour résoudre cette problématique, nous commençons par une approche consistant à établir des contrôles basés sur la logique floue [Zadeh 75] et s'intéresse à l'attraction et le suivi de cibles dynamiques [Stonier 98], [Jeong 99]. Cependant, la logique floue a l'inconvénient qu'il faut déterminer les règles de contrôle de façon appropriée, sans oublier qu'elles doivent être régulièrement mises à jour, surtout dans le cas des changements fréquents de la dynamique de l'environnement (ce qui est généralement le cas).

D'autres approches basées sur des modèles analytiques consistent en une fusion des méthodes des champs de potentiels et des fonctions de Lyapunov : dans [Zhu 09], ces deux approches sont combinées pour établir le contrôle d'un robot mobile aérien suivant une cible dynamique dont la vitesse est supposée constante. Le facteur vent, important dans la commande de ce type de véhicules, est aussi supposé constant. Les travaux présentés dans [Adams 99] s'inscrivent aussi dans cette même approche de fusion. Dans [Lee 00], une loi basée sur une

2. www.robocup.org

fonction de Lyapunov est établie. Elle prouve que le robot converge vers une distance désirée de la cible.

Notre contrôleur d'attraction vers une cible se base sur l'approche des champs de potentiels, son exécution consiste à exécuter en temps réel l'algorithme de champs de potentiel décrit précédemment, il nous fournit la prochaine position du robot en tenant compte des obstacles présents dans son voisinage.

3.3.3 Suivi de trajectoire

Ces dernières années, le problème de suivi de trajectoire (de référence) pour un robot mobile non-holonome est apparu comme un problème de premier ordre pour la communauté robotique. En effet, la forte utilisation des robots mobiles dans les domaines à haut risque notamment dans les sites nucléaires ou dans l'exploration spatiale, où l'être humain ne peut intervenir, nécessite la mise en œuvre d'algorithmes autonomes et performants pour assurer la tâche de suivi de trajectoire assignées aux robots une fois celle-ci déterminée, que ce soit dans un contexte d'évitement d'obstacle ou non. Dans la littérature nous trouvons plusieurs travaux concernant la poursuite de trajectoire ont été développés, nous en citerons quelques-uns après une brève formulation du problème.

Considérons le modèle non linéaire (représentatif d'un robot mobile) suivant :

$$\dot{q} = f(q, u) \tag{3.7}$$

où $q \in \mathbb{R}^n$ est l'état du système, $u \in \mathbb{R}^m$ la commande du système et $f : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$ une fonction différentiable partout. Le suivi d'une trajectoire de référence q_r consiste à trouver la commande u en fonction de q, q_r et du temps t, tel que l'erreur de suivi $q_e = q_r - q$ converge asymptotiquement vers zéro. Une illustration du principe de suivi de trajectoire pour un robot mobile non holonome est décrite dans la figure 3.10.

On trouve des approches de stabilisation par retour d'état non stationnaire continu et discontinu. L'approche discontinue est appropriée pour les systèmes qui ont des comportements fortement non linéaires, ainsi que des perturbations bornées [Young 88], [Slotine 83] et [de Wit 91]. Plusieurs travaux ont été réalisés pour la stabilisation du modèle du robot mobile non holonome [Khennouf 95], [Bloch 96] mais la plupart d'entre eux ne considèrent pas les perturbations. L'utilisation du principe de la commande par modes glissants avec action intégrale [Utkin 96] a été introduite par [Defoort 07] et [Defoort 05] pour le suivi de trajectoire d'un robot mobile non holonome en présence de perturbations.

L'approche de stabilisation par retour d'état non stationnaire continue à quant à elle été abordée suivant plusieurs méthodes différentes, parmi elles on trouve

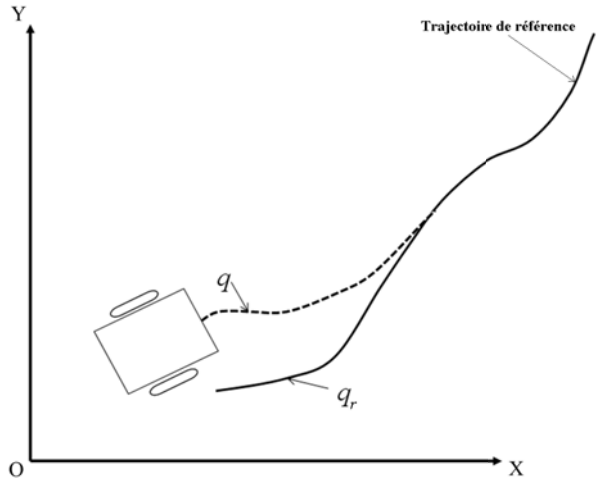


FIGURE 3.10 – Suivi de trajectoire pour un robot mobile non holonome de type unicycle.

la commande par fonction transverse, introduite par [Morin 01] et [Morin 03], elle assure une poursuite de trajectoires quelconques, y compris des trajectoires non admissibles. C'est une technique qui garantit la stabilisation pratique des erreurs de suivi. Il y a aussi la commande sous forme chaînée développée par [Floquet 03], la commande en boucle ouverte ou fermée basée sur l'application directe de commandes de la platitude [Fliess 95], la dernière que nous citerons est la commande non linéaire non stationnaire basée sur le modèle de l'erreur de suivi de trajectoire, développée par [Kanayama 91] et étendue par [Maalouf 06].

C'est cette dernière méthode qu'on a choisi d'implémenter dans notre contrôleur de suivi de formation au vu de sa flexibilité. Sachant que le modèle cinématique du robot utilisé est décrit par les équations 3.5, l'objectif est de suivre un robot virtuel de référence qui a respectivement v_r et ω_r comme vitesse linéaire et angulaire :

$$\begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\theta}_r \end{bmatrix} = \begin{bmatrix} \cos\theta_r & 0 \\ \sin\theta_r & 0 \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} v_r \\ \omega_r \end{bmatrix} \tag{3.8}$$

Les variables d'erreur e_x, e_y et e_θ correspondant aux variables d'erreur de posture instantanées sont définis comme suit :

$$\begin{bmatrix} e_x \\ e_y \\ e_\theta \end{bmatrix} = A \begin{bmatrix} x_r & x \\ y_r & y \\ \theta_r & \theta \end{bmatrix}$$
$$avec: \quad A = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.9}$$

l'équation 3.9 correspond aux erreurs de posture selon une trame de référence locale du robot (cf. figure 3.11). La matrice de rotation A, convertit donc les coordonnées globales en coordonnées locales. Lorsqu'on dérive les erreurs en utilisant les contraintes $\dot{x}_r \sin\theta_r = \dot{y}_r \cos\theta_r$ et $e_\theta = \theta_r \quad \theta$ on obtient :

$$\begin{bmatrix} \dot{e}_x \\ \dot{e}_y \\ \dot{e}_\theta \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} v + \begin{bmatrix} e_y \\ e_x \\ 1 \end{bmatrix} \omega + \begin{bmatrix} v_r \cos e_\theta \\ v_r \sin e_\theta \\ \omega_r \end{bmatrix} \tag{3.10}$$

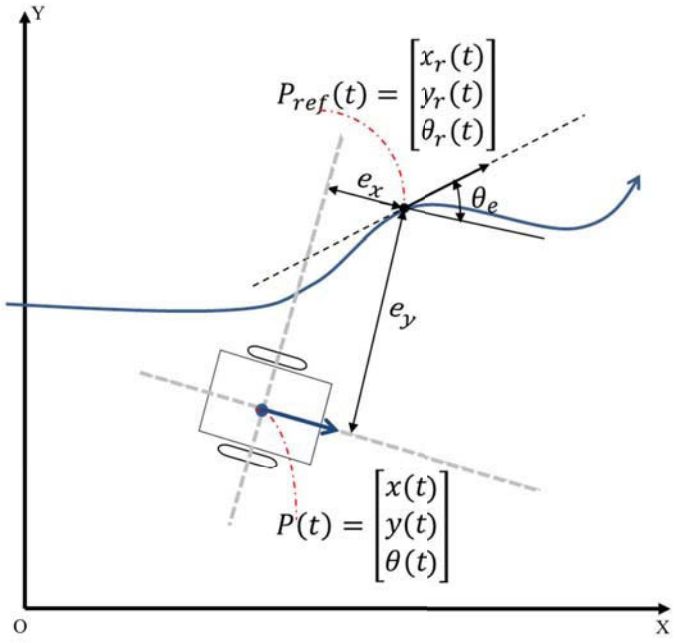


FIGURE 3.11 – Les paramètres de l'erreur de posture.

À partir de l'équation 3.10, l'objectif de la loi de commande est de faire en sorte de converger les erreurs vers zéro. On obtient les vitesses linéaires et angulaires d'entrée de la loi de commande v_f et ω_f suivantes :

$$\begin{array}{c} v_f = v_r \cos e_\theta + K_x e_x \\ \omega_f = \omega_r + v_r K_y e_y + K_\theta \sin e_\theta \end{array} \tag{3.11}$$

En substituant v_f et ω_f dans l'erreur dérivée de l'équation 3.9 on obtient :

$$\begin{bmatrix} \dot{e}_x \\ \dot{e}_y \\ \dot{e}_\theta \end{bmatrix} = \begin{bmatrix} e_y(\omega_r + v_r(K_y e_y + K_\theta \sin e_\theta)) - K_x e_x \\ -e_x(\omega_r + v_r(K_y e_y + K_\theta \sin e_\theta)) + v_r \sin e_\theta \\ -v_r(K_y e_y + K_\theta \sin e_\theta) \end{bmatrix} \tag{3.12}$$

La preuve de la stabilité du système est basée sur le choix de la fonction de Lyapunov [Maalouf 06] :

$$V = \frac{1}{2}(e_x^2 + e_y^2) + \frac{1 - \cos e_\theta}{K_y} \tag{3.13}$$

En dérivant V par rapport au temps on obtient :

$$\dot{V} = -K_x e_x^2 - \frac{K_\theta \sin^2 e_\theta}{K_y} \leq 0 \tag{3.14}$$

En imposant K_x, K_y et K_θ comme des constantes positives, l'inégalité donnée par 3.14 sera satisfaite, et donc le système avec la loi de commande sera stable. Le résultat nul correspondra au point d'équilibre où $e_x = e_y = e_\theta = 0$.

Par l'implémentation de de cet algorithme à notre contrôleur de suivi de trajectoire, nous obtenons des résultats très satisfaisants.

3.3.4 Génération de formation

Dans le module navigation en formation, un groupe de robots mobiles naviguent en maintenant des formes géométriques prédéterminées (ligne, colonne, triangle, etc.) en contrôlant l'emplacement de chaque robot par rapport aux autres, et ce tout en permettant de déplacer le groupe dans son ensemble. La formation géométrique peut être établie à partir de positions initiales prédéterminées, voire de positions aléatoires, et elle est maintenue au cours du mouvement du groupe. Cette navigation en formation doit garantir l'évitement des obstacles présents dans l'environnement. Ce genre de navigation est utile dans de nombreuses tâches de coopération telles que la tonte d'un terrain de sport, le transport ou manipulation d'objets impliquant conjointement plusieurs robots mobiles.

Pour contrôler la formation, chaque robot doit connaître sa position géométrique à maintenir dans la formation. Il existe trois façons pour accomplir cette détermination de la position : Unité-Centrale-Référencée (Unit-Center-Referenced), Chef-Référencé (Leader-Referenced) et Voisin-Référencé (Neighbour-Referenced) [Balch 98]. Dans l'approche Unité-Centrale-Référencée, chaque robot détermine son propre emplacement par rapport au centre de l'ensemble du groupe. Pour les deux autres approches Chef-Référencé et Voisin-Référencé, le principe est que chaque robot détermine sa position dans la formation soit par rapport au robot leader dans la méthode Chef-Référencé, ou par rapport à un autre robot prédéterminé dans la méthode Voisin-Référencé. Le principe ici est d'utiliser un robot comme chef de file central pour localiser et contrôler un groupe de robots. Le problème de la localisation dans cette équipe de robots est résolu par la détermination de la position et l'orientation de chaque robot suiveur par rapport à un seul chef.

Une variété d'approches a été proposées pour mettre en œuvre le contrôle de la formation d'un groupe de robots mobiles. [Balch 98] a proposé une approche basée sur le comportement pour le maintien de la formation d'une équipe de véhicules militaires terrestres sans pilotes. Chaque véhicule est équipé de GPS, de caméra et de capteurs laser. Une méthode de contrôle basée sur les schémas moteurs est utilisée pour mettre en œuvre les comportements de formation. Bien que cette méthode peut effectuer des comportements d'évitement d'obstacles à la formation, elle nécessite une communication intensive et des capteurs coûteux dans chaque robot. [Lewis 97] a appliqué le concept de structures rigides virtuels pour le maintien de formation. Chaque robot est commandé pour maintenir une relation géométrique rigide par rapport à un autre robot et à un cadre de référence. Mais leur intervalle d'application est contraint dans des milieux expérimentaux, en raison de la localisation où le calcul est basé sur un système de caméras fixes et indépendantes.

[Das 02] a présenté un paradigme pour la commutation entre les contrôleurs décentralisés qui permet des changements dans la formation. Une seule caméra omnidirectionnelle est utilisée dans tous les robots et un ordinateur hôte est utilisé comme une unité de traitement centralisée. L'ordinateur hôte reçoit une vidéo de tous les robots et calcule les vitesses relatives, ainsi que la position et l'orientation, entre chaque suiveur et ses leaders. [Vidal 03] traduit le problème de contrôle de la formation en une tâche d'asservissement visuel distinct pour chaque suiveur à l'aide d'une vision omnidirectionnelle. Le robot suiveur utilise la segmentation de mouvement pour estimer la position et la vitesse de son chef. L'utilisation de caméras distribuées dans ces approches nécessite une capacité importante de détection de chaque robot, ce qui augmente le coût du système (en terme de puissance calcul) de chaque robot. On peut citer d'autres approches

se basant sur la vision pour évaluer la position relative et l'orientation du robot leader comme [Fredslund 02], [Michaud 02], et [Chiem 04]. Il y a beaucoup d'autres travaux qui se concentre sur des questions telles que la planification de mouvement [Barfoot 04] et de contrôle [Fierro 01] des SMR en formation.

Le module de maintien de formation développé pour notre architecture contient un algorithme de génération de formation basé sur l'approche Leader-Follower. Cette approche consiste à générer les points de formation suivant un modèle géométrique, c'est une allocation dynamique de cibles pour les robots suiveurs. On a alors un calcul des positions des robots suiveurs (quel que soit leurs nombre) selon la forme géométrique choisie, pour chaque position du robot leader en tenant compte de sa posture (position et orientation).

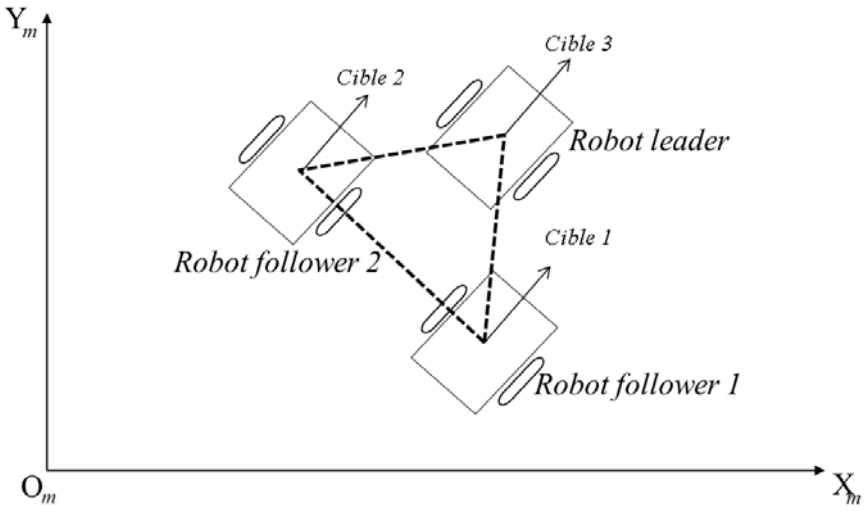


FIGURE 3.12 – Exemple d'une formation en triangle équilatéral.

L'algorithme utilisé pour l'allocation dynamique des cibles est basé sur la méthode de la distribution hiérarchique des cibles décrite dans [Benzerrouk 10], où une hiérarchie entre les robots est considérée. Ainsi, si plusieurs robots réclament la même cible, celui dont le rang hiérarchique est le plus élevé l'obtient. L'attribution des cibles est alors optimisée en se basant sur les distances mais aussi sur la résolution des problèmes de conflits grâce à une hiérarchie entre les robots. Pour que les robots puissent se partager les cibles de cette façon, notons qu'il est nécessaire d'établir une communication entre eux. Comme le critère de choix de la cible est la distance et l'ordre dans la hiérarchie de chaque robot en cas de conflit, chaque entité doit alors être capable d'envoyer aux autres éléments un vecteur composé de la distance qui la sépare de chaque cible, ainsi que son

ordre dans la hiérarchie. Ainsi, et pour N_c cibles composant la structure virtuelle, chaque élément d'ordre i du SMR doit envoyer un vecteur D_{S_i} de taille $N_c + 1$. Ce vecteur est composé des distances qui le séparent de chaque cible (dans un ordre connu de tous) et de l'ordre dans la hiérarchie.

Entrées: Distances $d_{S_{ij}}$ séparant le robot i des cibles j $(j = 1..N_c)$
Sorties: numéro cible choisie
1: Recevoir les vecteurs D_{S_k} provenant des entités robotiques pour $k = 1, ..., i-1$;
2: Envoyer le vecteur D_{S_i} après un temps δt ;
3: Recevoir les vecteurs D_{S_k} provenant des entités robotiques pour $k = i+1, ..., N$;
4: définir le booléen : CibleChoisie = faux ;
5: **Tantque** CibleChoisie == faux
6: classer par ordre croissant les distances $d_{S_{ij}}$ séparant le robot de rang i de toutes les cibles j dans un vecteur D_{S_i} ;
7: choisir la cible la plus proche j correspondant à $D_{S_i}(1)$;
8: **Si** la cible choisie j est réclamée par un robot de rang k tel que $k > i$ **alors**
9: Enlever du vecteur D_{S_i} la distance $d_{S_{ij}}$ séparant le robot i de la cible j ;
10: **Sinon**
11: CibleChoisie = vrai ;
12: numéro cible choisie=j ;
13: **finSi**
14: **fin Tanque**

Algorithm 1: Attribution des positions (cibles) aux robots suiveurs dans la formation [Benzerrouk 10].

Pour éviter des conflits de communication, l'ordre d'hiérarchie définit l'ordre d'envoi. Ainsi, si nous considérons l'ordre 1 comme étant le plus élevé, le robot correspondant envoie alors son vecteur D_{S_1} à tout le SMR. Une fois que le robot d'ordre 2 a reçu D_{S_1}, il envoie D_{S_2} et ainsi de suite jusqu'au robot d'ordre N. A noter qu'il est possible d'imposer que chaque entité i attend un temps δt avant d'envoyer son vecteur D_{S_i} pour prendre en considération un retard éventuel dans la réception des trames.

De cette manière, lorsqu'un robot i choisit une cible (la plus proche), il est en mesure de vérifier (en analysant les vecteurs reçus des autres entités) s'il peut se l'approprier ou si elle sera prise par un autre robot de rang supérieur (permettant de réduire le pour le SMR) et qu'il doit donc vérifier la cible suivante (cf. algorithme 1) [Benzerrouk 11].

Le contrôleur de suivi de cible dynamique est exécuté dès la réception d'une cible par un robot ce qui lui permet de garder sa position dans la formation.

Ainsi, on parvient à construire et maintenir la formation décrite dans la figure 3.12 jusqu'à arriver à la destination finale du SMR en mouvement.

3.3.5 Sélection d'action hiérarchique

Ce bloc est responsable de la commutation entre les différents contrôleurs du niveau organisation, soit : le contrôleur d'*Attraction vers une Cible*, celui de l'*Évitement Réactif d'Obstacle*, et le contrôleur de *Suivi de Trajectoire*, ainsi que l'activation des différents modules responsables soit de la *Génération de Formation* ou de la *Planification ou Re-Planification* de trajectoire. La logique ainsi que les différents modes de fonctionnement de ce bloc est donné dans la figure 3.13. C'est le bloc le plus important du niveau contrôle/commande, sa présence est primordiale pour aboutir à une architecture hybride (centralisée, distribuée, etc.).

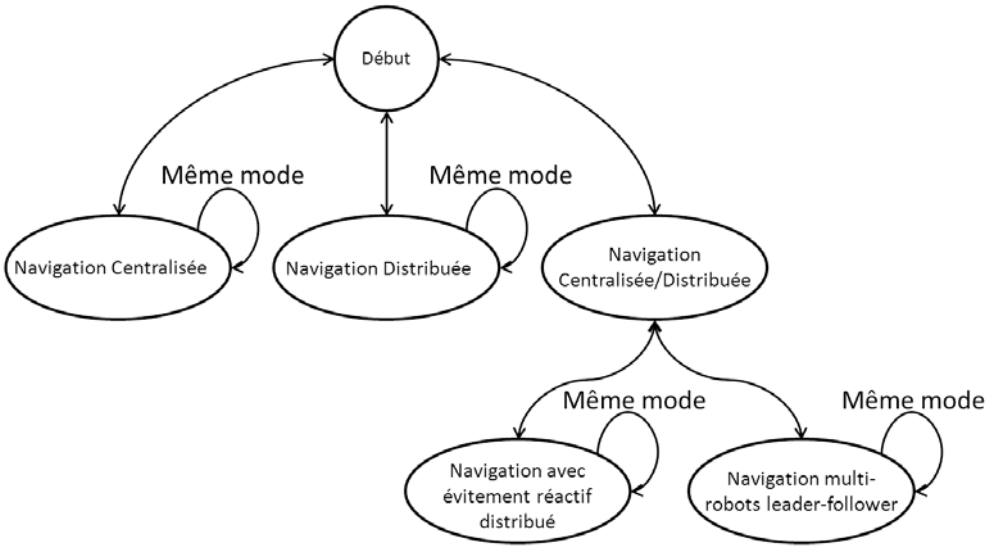


FIGURE 3.13 – Représentation des différents modes de fonctionnement du module de sélection d'action hiérarchique.

Comme on peut le remarquer dans la figure 3.13, on a spécifié quatre modes de navigations. Le module de sélection d'action hiérarchique active les contrôleurs et les modules qui régissent le fonctionnement de chaque mode de navigation. Dans

ce qui suit nous allons détailler le rôle du module de sélection d'action hiérarchique à travers le fonctionnement de chaque mode de navigation.

Navigation Centralisée :

Pour reproduire une navigation centralisée, le module de sélection d'action hiérarchique active le mode 'Centralisé'. On a alors une planification de trajectoire qui se fait pour tous les robots de la flottille qui est supervisée par l'agent superviseur, et dès qu'un robot rencontre un événement inattendu (nouvel obstacle, robot, etc.) on fait une re-planification supervisée pour obtenir une nouvelle trajectoire plus sûre. La figure 3.14 représente le fonctionnement de la navigation centralisée au sein du bloc de sélection d'action hiérarchique.

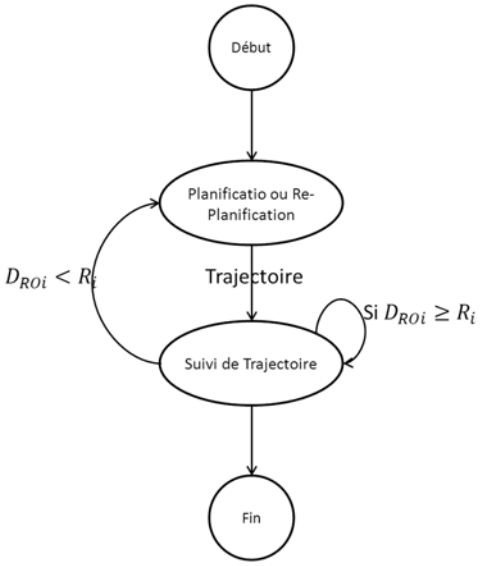


FIGURE 3.14 – Représentation du fonctionnement du module de sélection d'action hiérarchique en mode navigation centralisée.

Navigation Distribuée :

Quand la navigation distribuée est activée par le bloc de sélection d'action hiérarchique, tous les robots de la flottille sont totalement distribués avec une information localisée sur l'environnement dans lequel ils évoluent. Ils sont attirés par leurs cibles finales, et évitent les obstacles d'une manière réactive comme décrit dans la figure 3.15.

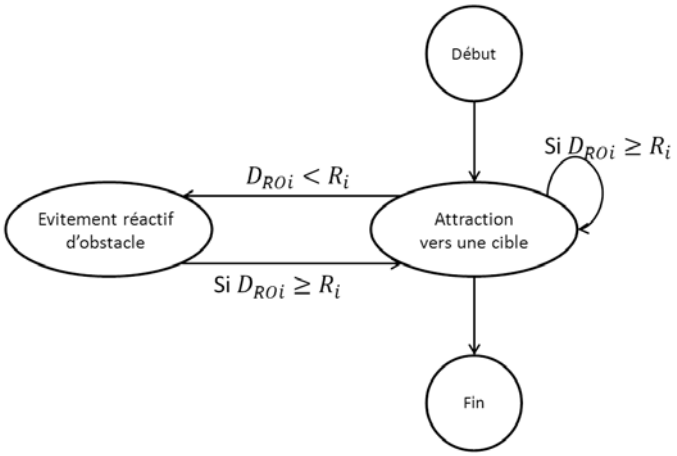


FIGURE 3.15 – Représentation du fonctionnement du module de sélection d'action hiérarchique en mode navigation distribuée.

Navigation Centralisée/Distribuée :

Dans l'architecture de contrôle proposée, nous avons différencié entre deux types de navigation centralisée/distribuée. Le premier est celui d'une navigation multi-robots où la partie centralisée est représentée par une planification de trajectoire supervisée par l'agent superviseur, et la partie distribuée intervient lors d'événements inattendus, les robots font alors un évitement réactif de l'obstacle en question. La figure 3.16 représente le fonctionnement de cette navigation centralisée/distribuée au sein du bloc de sélection d'action hiérarchique.

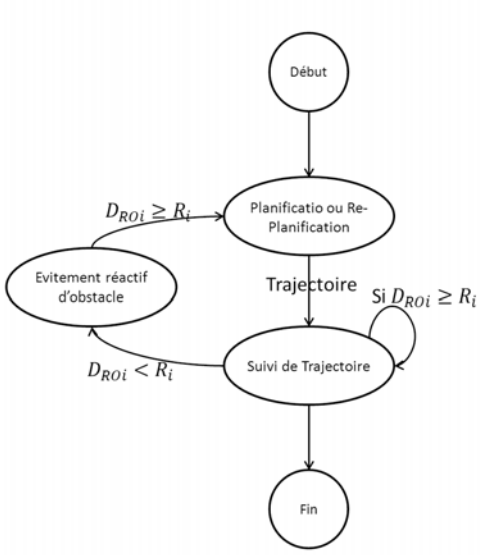


FIGURE 3.16 – Représentation du fonctionnement du module de sélection d'action hiérarchique en mode navigation centralisée/distribuée.

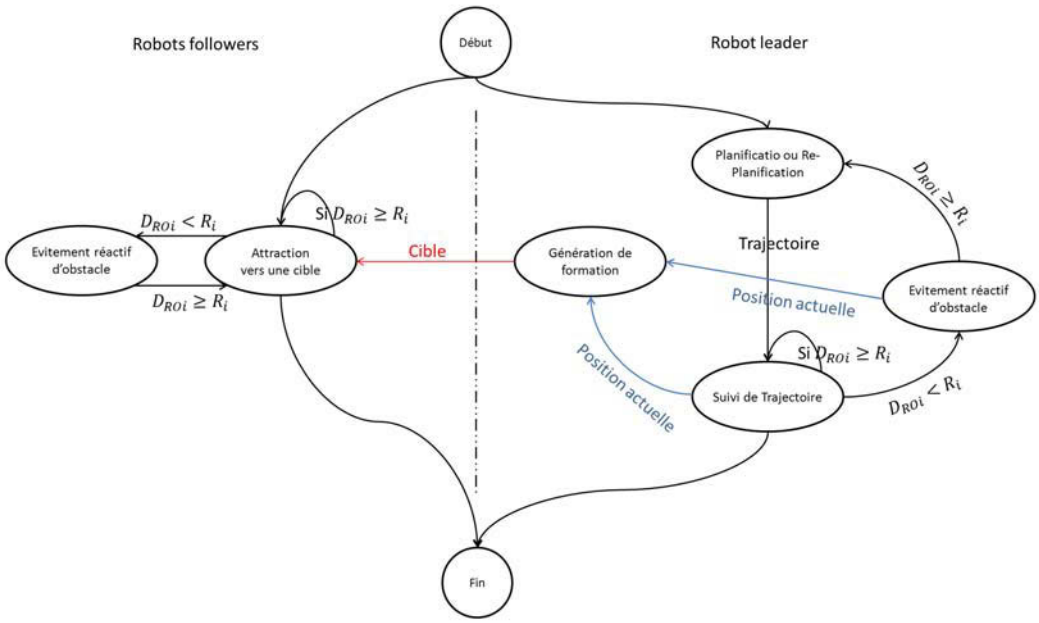


FIGURE 3.17 – Représentation du fonctionnement du module de sélection d'action hiérarchique en mode navigation centralisée/distribuée en formation.

Le deuxième type de navigation centralisée/distribuée est celui d'une navigation multi-robots en formation. Cette formation est guidée par un robot 'leader' qui est en navigation centralisée avec une gestion distribuée des événements inattendus, alors que les suiveurs sont en navigation totalement distribuée tout en maintenant leurs positions dans la formation. La figure 3.17 représente le fonctionnement de cette navigation centralisée/distribuée au sein du bloc de sélection d'action hiérarchique.

L'activation de ces différents modules et contrôleurs est étroitement liée au scénario et au type de navigation choisit. On peut quand même souligner que l'activation du contrôleur d'évitement survient si le robot entre dans le cercle d'influence de l'obstacle à éviter ou en d'autres termes, si $D_{RO_i} \leq R_i$.

3.4 Conclusion

L'architecture de contrôle proposée est inspirée des différentes stratégies proposées dans la littérature. En effet, on a adopté une approche hybride intégrant une partie cognitive et une autre réactive. La conception de l'architecture de contrôle proposée respecte les différents critères cités dans la section 1.4.

La particularité de notre architecture de contrôle réside dans sa hiérarchisation. En effet, on l'a organisée en deux principaux niveaux, le haut niveau appelé *Niveau d'organisation* et le bas niveau appelé *Niveau de Contrôle*. Dans ce chapitre nous avons présenté notre architecture de contrôle hybride globale, ainsi que les différentes composantes bas niveau de celle-ci, soit le niveau de contrôle/commande qui, grâce aux différents modules le composant, nous permet de répondre à pratiquement tous les besoins bas niveau dont un système multi-robots aurait besoin, pour accomplir la majorité des missions de navigation multi-robots, dans des environnements encombrés ou pas.

Dans ce qui suit, nous présenterons le niveau organisation de l'architecture de contrôle proposée, qui permet la coordination et la supervision de mission d'un groupe de robots mobiles se basant sur une approche organisationnelle multi-agents, ce qui représente une contribution importante de nos travaux de thèse. Nous verrons ainsi ses différentes composantes et leurs relations, ainsi que comment l'organisation des différents agents/robots se met en place.

Chapitre 4

Modèle de l'organisation dédié à l'architecture hybride proposée

Dans ce chapitre, nous nous focaliserons sur le niveau organisation de notre architecture de contrôle/commande dédié à la coordination d'un système multi-robots. Notre proposition consiste à évaluer le potentiel des modèles organisationnels basés sur les systèmes multi-agents, afin d'assurer la coordination des missions effectuées par un système multi-robots.

Le modèle choisi s'inspire de ceux appliqués aux institutions électroniques, notamment le modèle $\mathcal{M}\text{OISE}^{Inst}$ [Gateau 07] (cf. figure 4.1) qui dispose de modèles appropriés de la gestion opérationnelle des systèmes multi-agents. En effet, grâce à ses différentes spécifications il permet une organisation souple et efficace du système multi-agents ainsi qu'une bonne gestion des différents types de fonctionnement souhaités de l'organisation.

4.1 Spécification de l'organisation

Dans la structure globale de l'architecture de contrôle/commande proposée, précédemment décrite dans la figure 3.6, nous constatons que le niveau organisation de celle-ci est composé principalement du modèle organisationnel multi-agents $\mathcal{M}$OISE^{Inst}. L'adaptation de ce modèle d'organisation à un cas d'école en robotique mobile, notamment celui du transport et déplacement de marchandises dans un entrepôt par un système multi-robots, nous permet de modéliser un système multi-agents/robots à travers quatre dimensions (cf. figure 4.1) :

— tous les agents/robots existants dans la flottille sont référencés dans la Spécification Structurelle **(SS)**, ainsi que leurs contraintes physiques, leurs rôles et les liens qui existent entre eux.
— la Spécification Fonctionnelle **(SF)** regroupe tous les buts à accomplir par les agents/robots (i.e., but planification de trajectoire, but attente autorisation, but suivi de trajectoire, etc.), qui sont organisés séquentiellement dans des missions spécifiques dans la **SF** (i.e., mission planification, mission évitement d'obstacles, etc.).
— la Spécification Contextuelle **(SC)** contient les différents contextes où peut se trouver l'organisation, on y trouve ainsi plusieurs scènes regroupant plusieurs missions ce qui représente un déroulement d'un scénario d'exécution (i.e., une scène planification et re-planification sera composée de : mission planification, mission évitement d'obstacles).
— la Spécification Normative **(SN)** à la tâche de forcer, autoriser ou interdire une action. Dans notre cas de figure, elle active les éléments dont l'organisation va se servir et désactive les autres.

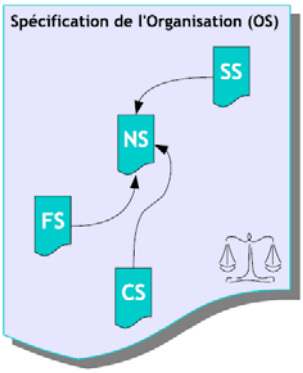


FIGURE 4.1 – Spécification de l'organisation de modèle organisationnel $\mathcal{M}$OISE^{Inst}.

Pour permettre une meilleure compréhension de ces quatre dimensions, nous décrirons l'approche orientée-institution pour modéliser notre système SMAR multi-agents/robots. En effet, pour atteindre un objectif par le système multi-robots dans un environnement partiellement connu, nous avons donné à nos agents/robots la possibilité d'atteindre un objectif selon plusieurs modes de fonctionnement :

— *Mode de navigation par planification.*
— *Mode de navigation réactive.*
— *Mode de navigation en formation.*

Nous leur avons donné aussi la possibilité de transitions entre ces différents modes de fonctionnement, pour obtenir une navigation sûre pour le système multi-robots, prenant en compte les événements inattendus. Selon la manière d'implémenter de cette approche, l'exécution de la navigation mono et multi-robots peut correspondre à une large gamme d'architectures de contrôle :

— *centralisées.*
— *distribuées.*
— *centralisées/distribuées.*

Dans ce qui suit, nous décrirons le fonctionnement des différents modes de navigations à travers les quatre spécifications du modèle organisationnel utilisé, pour aboutir à reproduire le comportement des trois types d'architectures mentionnées, et pour pouvoir passer facilement de l'une à l'autre.

4.1.1 Spécification structurelle *(SS)*

Dans la spécification structurelle on définit des rôles (niveau individuel) des agents/robots, ainsi que les relations entre ces rôles (niveau social) et les groupes d'agents/robots (niveau collectif).

La *SS* peut être décrite en trois composantes :

— les entités structurelles.
— les relations entre ces entités structurelles.
— les contraintes de cardinalité.

Sur cette base on a défini deux entités structurelles (cf. figure 4.2) : la première c'est le rôle "r" pour chaque agent/robot : $rRobot\{i\}$ avec $i = \{1..n\}$, n étant le nombre total de robots. La deuxième c'est la définition d'un autre rôle pour un agent superviseur : $rSuperviseur$, c'est un agent externe au système multi-robots qui a comme tâche globale, celle de gérer et coordonner les autres agents/robots dans le cas de l'exécution d'une navigation **centralisée**.

Tous ces rôles ont une cardinalité de 1, de cette façon un et seulement un agent est associé à chaque robot.

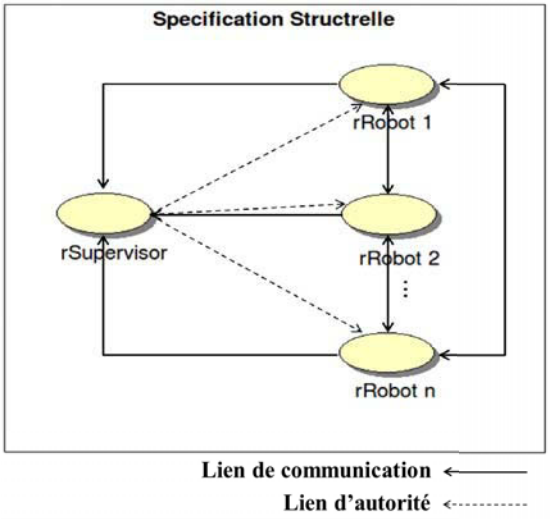


FIGURE 4.2 – Représentation de la Spécification Structurelle.

On a aussi défini deux types de relations entre les entités structurelles (cf. figure 4.2) :

— *Communication* : une relation de communication bilatérale est définie entre tous les agents existants.
— *Autorité* : une relation d'autorité est définie seulement entre le superviseur et les agents/robots, de façon à ce que le superviseur puisse leurs envoyer des ordres.

4.1.2 Spécification fonctionnelle *SF*

Dans la spécification fonctionnelle on procède à l'établissement d'objectifs **"goals"** à atteindre (niveau individuel) par l'organisation. Ils sont regroupées dans des tâches comprises dans un modèle de fonctionnement social à suivre. Celui-ci exprime le processus social que les agents/robots doivent suivre pour atteindre l'objectif collectif représenté par l'objet racine du système qui est la mission de navigation ou d'évitement d'obstacle, etc. elle peut donc être représentée par un arbre dont les feuilles représentent ces objectifs **"goals"** qui peuvent être réalisés par un seul agent (niveau individuel). Les missions regroupent alors les **"goals"** en des ensembles cohérents, prêtes à être effectuées par les agents.

Les goals sont exécutés dans un ordre bien déterminé, selon deux modalités :

— Séquence ($\rightarrow$) : $g_1 \rightarrow g_2 \rightarrow ... \rightarrow g_p$: cette séquence stipule que le goal g_1 doit être exécuté avant le goal g_2 et ainsi de suite jusqu'à arriver au goal

g_p.

— Parallélisme ($\|$) : $g_1 \| \{g_2, g_3, ..., g_p\}$: cette expression signifie que les goals $g_2, g_3, ...et g_p$ doivent être exécutés en parallèle après l'exécution du goal g_1.

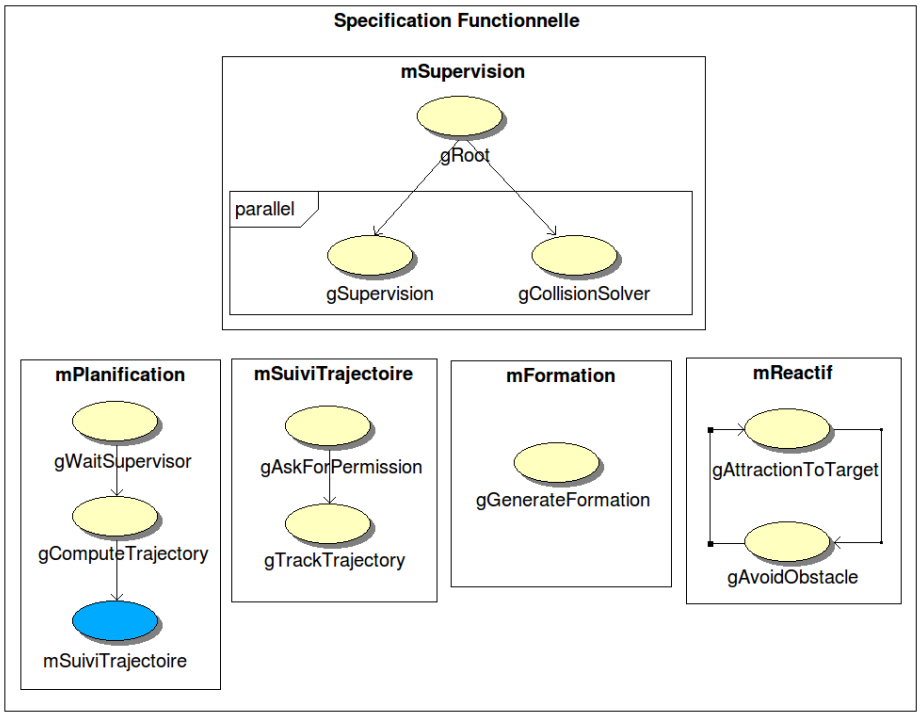


FIGURE 4.3 – Représentation de la Spécification Fonctionnelle.

Dans le cas de la navigation de robots mobiles, nous avons besoin de cinq missions principales pour permettre la navigation sans conflits dans un environnement peu connu voir inconnue. En effet, il nous faut une mission de supervision pour éviter les conflits entre les robots, une mission de planification de trajectoire pour offrir un chemin sécurisé aux robots, une mission de suivi de cette trajectoire, une mission de navigation réactive pour les obstacles inattendus, et une mission de navigation en formation. Ces cinq missions sont détaillées dans ce qui suit (cf. figure 4.3) :

1. La mission ***mSupervision***, est représentée par :

$$mSupervision = gRoot \| \{gSupervision, gCollisionSolver\} \quad (4.1)$$

C'est une mission dédiée à l'agent superviseur, elle contient trois goals :

— *gRoot* : sert à la création de l'agent superviseur.

— *gSupervision* : c'est un goal qui est réalisé par l'agent superviseur, et dans lequel il gère les messages et les requêtes venant des agents robots.
— *gCollisionSolver* : c'est un goal qui est aussi réalisé par le superviseur, dans lequel il résout les problèmes et les conflits qui peuvent apparaître pendant la navigation des agents/robots. Ces conflits peuvent être liés principalement à la navigation des robots qui suivent leurs trajectoires respectives. Puisque nous disposons d'un algorithme de génération de trajectoires fiables, ce goal nous permet d'ajuster les vitesses des robots pour éviter les collisions, il s'exécute suivant l'algorithme 2. Dans un cas où le robot 2 intègre la flottille.

Entrées: La trajectoire et la vitesse du robot qui intègre la flottille : $Robot_2(Traj_2, Vitesse_2)$
Sorties: La nouvelle vitesse du $Robot_2$

Pour tous les robots existants dans la flottille **faire**
  Pour chaque t temps d'exécution **faire**
    Pour chaque position du robot choisit $Robot_i$ **faire**
      Calculer $Dist_{Robot_i,Robot_2}$ la distance entre le $Robot_i$ et $Robot_2$
      Si $Dist_{Robot_i,Robot_2} <= Dist_{Critique}$
        $Vitesse_2 = Vitesse_2 - 0.05$;
        break
      finSi
    finPour
  finPour
finPour

Algorithm 2: Algorithme de résolution de conflits.

L'expression 4.1 signifie que le goal *gRoot* s'exécute en premier, suivi par les deux autres goals *gSupervision* et *gCollisionSolver* exécutés en parallèle. Grâce à cette mission on obtient un agent **central** qui gère la navigation du système multi-agents/robots, en étant constamment connecté aux robots et résout tous les problèmes liés à cette navigation. On a là un comportement d'architecture centralisée.

2. La mission ***mPlanification***, est représentée comme suit :

$$mPlanification = gWaitSupervisor \rightarrow gComputeTrajectory \rightarrow mSuiviTrajectoire \quad (4.2)$$

C'est une mission qui peut concerner n'importe quel agent/robot, elle contient deux goals et elle fait appel à une autre mission :
— *gWaitSupervisor* : c'est un goal qui, dès que l'agent/robot est prêt, attend que l'agent superviseur soit créé et initialisé, il reçoit alors un message de ce dernier pour qu'il s'arrête et on passe au goal suivant.

— *gComputeTrajectory* : c'est un goal qui exécute une planification de trajectoire, c'est un goal qui fait appel à notre module de planification de trajectoire du niveau de contrôle de l'architecture de contrôle proposée, décrit dans la section 3.3.1. Il génère donc une trajectoire en prenant en compte la cinématique du robot utilisé ainsi que les obstacles connus de l'environnement, d'une position de départ à une autre d'arrivée. Il transmet cette trajectoire à la mission suivante.
— *mSuiviTrajectoire* : c'est une mission de suivi de trajectoire, qui va être décrite ci-après.

L'expression 4.2 signifie que cette mission s'exécute séquentiellement, commençant par le goal *gWaitSupervisor* ensuite *gComputeTrajectory*, une fois la trajectoire générée elle est transmise à la mission *mSuiviTrajectoire* pour qu'elle soit suivie.

3. La mission ***mSuiviTrajectoire***, est représentée comme suit :

$$mSuiviTrajectoire = gAskForPermission \rightarrow gReachTarget \qquad (4.3)$$

C'est une mission étroitement liée à la mission précédente (***mPlanification***), elle comporte deux goals :
— *gAskForPermission* : est un goal qui a la tâche de récupérer la trajectoire envoyée par la mission *mPlanification*, concorde cette trajectoire avec la vitesse max du robot concerné et la transmet à l'agent superviseur comme requête afin d'exécuter le suivi de la trajectoire avec comme vitesse max celle du robot concerné. Une fois cette requête transmise, ce goal attend soit une permission de suivre la trajectoire comme indiqué initialement, soit en cas de conflits éventuel l'agent superviseur procède à un changement de la vitesse avec laquelle le suivi de trajectoire va être effectué, puis l'envoi au goal.
— *gReachTarget* : quelle que soit la permission envoyée par l'agent superviseur, elle est transmise à ce goal accompagné par la trajectoire à suivre, *gReachTarget* a alors la tâche d'exécuter le suivi de trajectoire en faisant appel au contrôleur de suivi de trajectoire décrit dans la section 3.3.3.

Cette mission s'exécute d'une manière séquentielle (cf. expression 4.3), commençant par *gAskForPermission* puis *gReachTarget*.

4. La mission ***mRéactif***, est représentée comme suit :

$$\begin{aligned} mReactif = {} & gAttractionToTarget \rightarrow gAvoidObstacle \\ & \rightarrow gAttractionToTarget... \end{aligned} \qquad (4.4)$$

Elle est composée de deux goals qui s'exécutent séquentiellement et d'une manière bouclée infiniment tant qu'il y a des obstacles nouvellement détectés, avec des conditions de passage de l'un à l'autre bien définies :

— *gAttractionToTarget* : c'est un goal de navigation réactive, ayant comme approche celle de l'attraction vers une cible dynamique décrite précédemment (cf. section 3.3.2), ce goal une fois actif fait appel au contrôleur d'attraction vers une cible présent dans le niveau de contrôle de l'architecture de contrôle proposée. Il s'arrête de fonctionner dès qu'il y a détection d'obstacle.
— *gAvoidObstacle* : ce goal fait appel au contrôleur d'évitement d'obstacles présent dans le niveau contrôle de l'architecture de contrôle proposée, il permet donc l'évitement réactif d'obstacles. Une fois qu'on considère que le robot a évité l'obstacle on repasse au goal *gAttractionToTarget*.

Cette mission nous permet d'avoir une navigation réactive, en faisant une attraction vers la cible qu'on veut atteindre, durant cette phase on peut rencontrer des obstacles, c'est pour cela que notre mission fait appel au contrôleur d'évitement d'obstacles pour effectuer cette tâche. Une fois l'évitement accompli on repasse au contrôleur d'attraction vers notre cible jusqu'à l'arrivée à destination. L'appel du contrôleur d'évitement d'obstacles se fait autant de fois qu'il est nécessaire jusqu'à l'arrivée du robot à sa destination.

5. La mission ***mFormation***, contient un seul goal :

$$mFormation = gGenerateFormation \tag{4.5}$$

Le goal *gGenerateFormation* permet à un robot considéré comme leader d'un système multi-robot de générer des positions des robots suiveurs selon la formation choisie (on a pu coder deux types de formations : triangle et ligne horizontale), à chaque fois qu'il est nécessaire, en faisant appel à ce goal qui à son tour exécute le module de maintien de formation contenu dans le niveau de contrôle de l'architecture de contrôle proposée (cf. section 3.3.4), une illustration du fonctionnement de cette mission est présentées dans la figure 7.12.

On a ainsi mis en place cinq fonctionnalités distinctes dans le modèle organisationnel utilisé, celles-ci peuvent être combinées entre elles de façon à obtenir le fonctionnement désiré.

4.1.3 Spécification contextuelle *SC*

Dans la spécification contextuelle on définit les différents contextes dans lesquels l'organisation va évoluer, et les transitions d'un contexte à l'autre. C'est

donc l'ensemble des contextes dans lesquels l'organisation peut évoluer ainsi que tous les événements déclencheurs du changement de contexte.

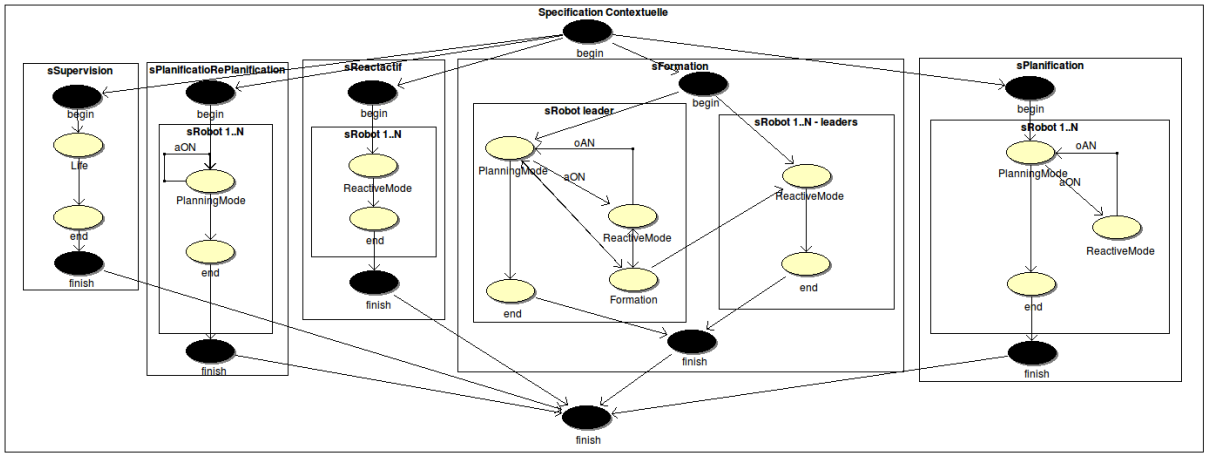


FIGURE 4.4 – Représentation de la Spécification Contextuelle.

La spécification contextuelle peut être vue, dans notre contexte, comme un graphe d'états-transition récursif, où les états sont appelés *contextes*, et un groupe de contextes est appelé *scène s*.

On a donc défini dans notre spécification contextuelle cinq scènes, chacune correspond à un scénario de fonctionnement bien déterminé (cf. figure 4.4), chaque scène combine une ou plusieurs missions décrites dans la spécification fonctionnelle d'une manière à avoir un scénario de fonctionnement donné. Elles sont décrites comme suit :

Les scènes *sSupervision* et *sPlanificationRePlanification*

On a choisi de grouper la description de ces deux scènes à cause du lien étroit qui les lie. En effet, l'exécution de ces deux scènes décrit une navigation multi-robot centralisée en utilisant l'algorithme de planification et re-planification décrit dans le niveau de contrôle de l'architecture de contrôle proposée. La figure 4.5 décrit le fonctionnement de ces deux scènes.

— La scène *sSupervision* active l'agent superviseur en exécutant la mission *mSupervision* décrite dans la spécification fonctionnelle (cf. section 4.1.2). Ce superviseur a la charge de la gestion et du contrôle du système multi-agents/robots. Il est lié étroitement à la scène *sPlanificationRePlanification*, récupérant les trajectoires envoyés par la scène tout en résolvant les

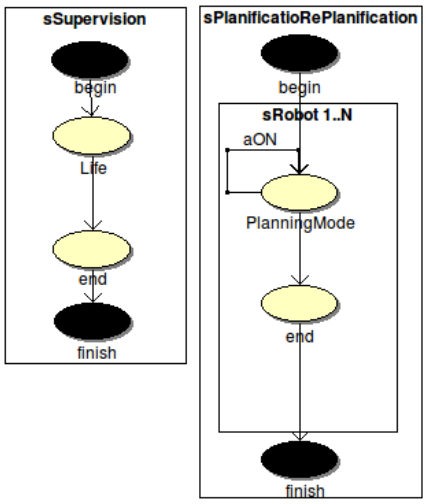


FIGURE 4.5 – Les deux scènes *sSupervision* et *sPlanificationRePlanification*.

conflits entre robots, afin d'autoriser le suivi de ces trajectoires.

— La scène *sPlanificationRePlanification* s'active une fois que les agents/robots ont été créés et initialisés, dans cette scène l'agent/robot active la mission *mPlanification* de la spécification fonctionnelle qui fait une planification de trajectoire comme décrit précédemment puis, l'agent/robot la suit après réception l'autorisation du superviseur. Dans le cas où l'agent/-robot rencontre un obstacle inattendu, cette scène garantit la résolution de ce problème en faisant un autre appel à la même mission de planification en prenant en compte l'obstacle inconnu grâce à la transition *aON* (Avoid Obstacle *N*) (cf. figure 4.5).

On a donc une navigation dite centralisée, gérée par un agent superviseur constamment connecté aux autres agents/robots, chacun d'eux exécute d'une façon distincte la scène *sPlanificationRePlanification* et communiquent avec l'agent superviseur et exécutent ses consignes pour une navigation sûre et sans conflits. Le fonctionnement de cette scène peut être représenté par l'expression suivante (la notation / renvoie aux composantes de la scène qui la précède) :

$$\begin{array}{l} sRobot\{i\}/PlanningMode \quad \forall\, i \in \{1..n\} \\ sSupervision/life \end{array} \tag{4.6}$$

La scène *sReactif*

Cette scène représente un exemple de navigation complètement distribuée (cf. figure 4.6), une fois les agents/robots créés et initialisés, ils exécutent chacun la scène *sReactif*, qui fait à son tour un appel à la mission *mReactif* de la spécification fonctionnelle.

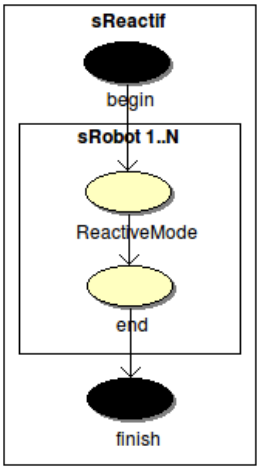


FIGURE 4.6 – La scène *sReactif*.

Le système multi-robots en mode réactif navigue alors vers la cible à atteindre tout en ayant la possibilité de résoudre les problèmes liés à l'apparition dans l'environnement d'un obstacle inattendu grâce au contrôleur d'évitement d'obstacles contrôlé par la mission *mRéactif* à travers le goal *gAvoidObstacle* de la spécification fonctionnelle.

La scène *sPlanificationReactifReplanification*

L'exécution de cette scène présente un exemple d'architectures centralisées/-distribuées. Elle représente une combinaison entre les deux scènes précédemment décrites *sReactif* et *sPlanificationRePlanification* (cf. figure 4.7), où on a un superviseur qui gère les agents/robots quand ils sont en planification. Cette connexion est interrompue lorsque les agents/robots passent en mode réactif.

Une fois les agents (superviseur et robots) sont créés et initialisés. Chacun des agents/robots commencent avec le mode planification supervisée par l'agent superviseur, comme précédemment décrit. Ils font appel au module de planification de trajectoire (correspondant à la mission $mPlanification$), générant une

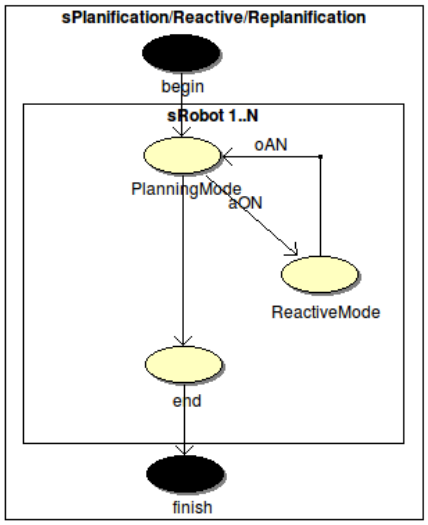


FIGURE 4.7 – La scène *sPlanificationReactifReplanification*.

trajectoire qui doit être validée par le superviseur puis ils la suivent (exécution de la mission $mSuiviTrajectoire$). Cette exécution peut être exprimée par les expressions suivantes pour un nombre total n de robots :

$$\begin{aligned} &sRobot\{i\}/PlanningMode \quad \forall\, i \in \{1..n\} \\ &sSupervision/life \end{aligned} \tag{4.7}$$

Dans le cas où un obstacle inattendu survient par exemple pour l'agent/robot numéro 5, on exécute la transition aON (Avoid Obstacle N) et il se met en mode réactif, et exécute la mission $mReactif$ qui grâce au goal $gAvoidObstacle$ permet l'évitement réactif de cet obstacle initialement non pris en considération (en activant le contrôleur d'évitement d'obstacles), dans ce cas l'expression 4.7 devient :

$$\begin{aligned} &sRobot\{5\}/ReactiveMode \\ &sRobot\{i\}/PlanningMode \quad \forall\, i \in \{1..n\} \;\; et\; i \neq 5 \\ &sSupervision/life \end{aligned} \tag{4.8}$$

Une fois l'évitement d'obstacles effectué on exécute la transition oAN (Obstacle Avoided N) pour que l'agent/robot en question (numéro 5) repasse en mode planification supervisée par l'agent superviseur.

La scène *sFormation*

Cette scène représentée dans la figure 4.8, décrit le scénario d'une navigation d'un système multi-robots en formation, son exécution est typique des architectures de contrôles centralisées/distribuées. En effet, cette scène contient deux autres scènes correspondantes aux deux types d'agents/robots du groupe, soit l'agent/robot leader et les agents/robots suiveurs (followers) :

— la scène *sRobot Leader* : est exécutée par l'agent/robot leader, elle est très similaire à la scène *sPlanificationReactifReplanification*, sauf qu'on a rajouté à celle là le contexte *Formation*, le leader exécute alors la mission $mPlanification$ supervisée par l'agent superviseur, comme on l'a décrit précédemment. Pendant le déroulement de cette mission on exécute la mission $mFormation$ qui grâce au goal $gGenerateFormation$ génère les positions des robots suiveurs dans la formation, et modifie les positions de destination des robots suiveurs par l'allocation dynamique de cibles, en les envoyant à la scène *sRobotFollowers*.
Dans le cas où le leader rencontre un obstacle inattendu, on applique la transition aON (Avoid Obstacle N) pour passer en mode réactif exécutant la mission $mReactif$ tout en exécutant la mission $mFormation$. De cette façon le leader continue à transmettre les positions de la formation aux robots suiveurs. Une fois l'évitement d'obstacle effectué, on applique la transition oAN (Obstacle Avoided N) pour repasser en mode planification supervisée avec les mêmes consignes que la première fois jusqu'à ce que le leader atteint sa cible.

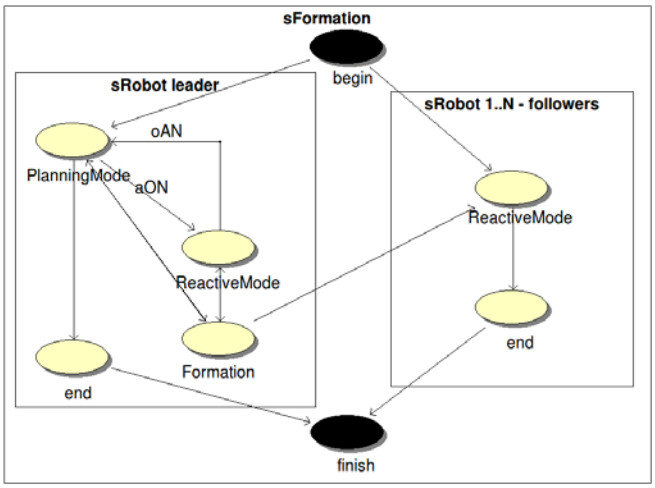


FIGURE 4.8 – La scène *sFormation*.

— la scène *sRobot Followers* : est dédiée aux agents/robots suiveurs, ils sont en mode réactif, exécutant la mission *mReactif* qui grâce au goal *gAttractionToTarget*, naviguent vers une cible dynamique, cette cible change constamment car la scène *sRobotFollowers* est connectée par un lien de communication à la mission *mFormation*, qui lui envoie pendant toute la navigation les nouvelles positions dans la formation de l'agent concerné. Cette allocation de cible s'arrête une fois que l'agent/robot leader arrive à destination.

La scène $sFormation$ contrôle l'exécution des deux sous-scènes qui y sont incluses. Suivant le type d'agent/robot concerné, on peut représenter le fonctionnement de cette scène avec les expressions suivantes :

— Au début de l'exécution de cette scène on a :

$$\begin{aligned} &sRobot\{Leader\}/PlanningMode \\ &sRobot\{Follower(i)\}/ReactiveMode \quad \forall\, i \in \{1..(n-1)\} \\ &sSupervision/life \end{aligned} \tag{4.9}$$

— Dans le cas où l'agent/robot leader rencontre un obstacle inattendu, l'expression 4.9 devient :

$$\begin{aligned} &sRobot\{Leader\}/ReactiveMode \\ &sRobot\{Follower(i)\}/ReactiveMode \quad \forall\, i \in \{1..(n-1)\} \\ &sSupervision/end \end{aligned} \tag{4.10}$$

Une fois cet évitement réactif terminé on retrouve le fonctionnement initial (cf. expression 4.9).

Dans cette scène on retrouve en même temps les deux types de comportements, centralisés et distribués. Le robot leader est supervisé par l'agent superviseur central, et les suiveurs ne sont pas supervisés car ils sont en mode réactif et suivent le leader. L'architecture de contrôle proposée dans ce cas, se comporte typiquement comme une architecture centralisée/distribuée.

4.1.4 Spécification normative *SN*

La spécification normative définit les relations déontiques reliant les trois spécifications indépendantes citées précédemment (*SS, SF, SC*). Cette spécification normative énonce clairement les droits et les devoirs de chacun des rôles, de la spécification structurelle, par rapport aux missions (ensemble de goals) de la spécification fonctionnelle dans un cas spécifique de fonctionnement appelé scène (ensemble de contextes) de la spécification contextuelle.

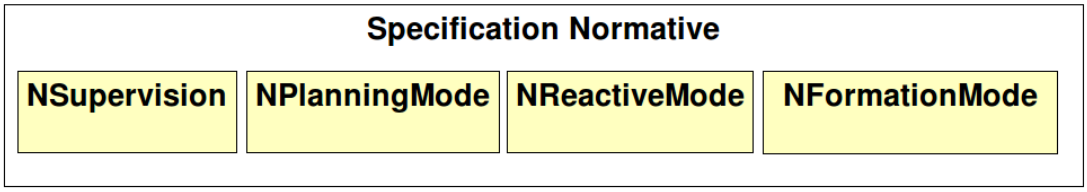


FIGURE 4.9 – Représentation de la Spécification Normative.

Cette spécification a donc la charge de faire le lien entre toutes les spécifications existantes dans le modèle organisationnel utilisé. Pour ce faire on a défini quatre normes qui nous permettront de contrôler efficacement la navigation de notre SMAR (système multi-agents/robots) comme décrit dans la figure 4.9 :

— la norme *NSupervision* est une norme dédiée exclusivement à l'agent superviseur. Elle force l'agent jouant le rôle *rSuperviseur* à exécuter la mission *mSupervision* quand l'institution se trouve dans le contexte *sSupervision/-life*.

Pour les n robots du système multi-agents/robots on a défini les trois normes suivantes :

— la norme *NPlanningMode* représentée aussi par l'expression :

$$N\{i\}PlanningMode \quad avec\, i \in \{1..n\}$$

Quand l'institution est dans le contexte $sRobot\{i\}/PlanningMode$, cette norme force l'agent jouant le rôle $rRobot\{i\}$ à exécuter la mission *mPlanification*.

— la norme *NReactiveMode* exprimée aussi par :

$$N\{i\}ReactiveMode \quad avec\, i \in \{1..n\}$$

Cette norme permet de forcer l'agent jouant le rôle $rRobot\{i\}$ à exécuter la mission $mReactif$ quand l'institution se trouve dans le contexte $sRobot\{i\}/ReactiveMode$.

— la norme *NFormationMode* aussi exprimée par :

$$N\{i\}FormationMode \quad avec\, i \in \{1..n\}$$

Comme les autres normes, cette norme permet de forcer l'agent jouant le rôle $rRobot\{i\}$ à exécuter la mission $mFormation$, quand l'institution se trouve dans le contexte $sRobot\{i\}/Formation$.

Cette description des différentes normes de la spécification normative, nous a permis de voir les liens qu'elle crée entre les différentes spécifications du modèle organisationnel proposé. En effet, cette spécification représente le noyau du niveau organisation de l'architecture de contrôle proposée, car à travers elle on peut avoir les types d'exécutions souhaités : la navigation centralisée, la navigation centralisée/distribuée, ou encore la navigation distribuée. Nous distinguons ainsi les cas suivants :

- pour avoir un comportement **centralisé**, on active les normes $NSupervision$ et $NPlanningMode$ dans la spécification normative, afin d'avoir ainsi une navigation avec planification et re-planification supervisées.
- pour avoir un comportement **centralisé/distribué**, on a deux cas de figures, soit :
  1. on active les trois normes : $NSupervision$ et $NPlanningMode$ s'exécutant en parallèle et enfin $NReactiveMode$ pour obtenir une navigation avec planification centralisée puis un évitement réactif d'obstacles (distribué), et enfin une re-planification centralisée.
  2. on active les quatre normes : $NSupervision$ avec $NPlanningMode$, $NReactiveMode$ et $NFormationMode$ pour obtenir une navigation en formation avec un robot leader en planification centralisée, et des robots suiveurs en mode navigation réactive (distribuée).
- pour avoir un comportement **distribué**, seule la norme $NReactiveMode$ est activée, où tous les robots du système multi-robots sont en mode réactif pendant toute la navigation jusqu'au point d'arrivé. C'est donc un comportement totalement distribué.

Dans ce qui suit, nous présenterons les différents algorithmes de coordination multi-robots, de la navigation en formation ainsi que celui des deux types de planification et re-planification présentées.

4.2 Logiciel de mise en œuvre de l'institution des agents pour la coordination des robots

Nous nous intéresserons dans cette section à l'institution du système multi-agents destiné à la coordination dans l'architecture de contrôle proposée, ce qui nous permet la création et la gestion d'un système multi-agents/robots.

En effet, pour effectuer la mise en œuvre de l'institution des agents dans l'architecture proposée, nous avons développé simulateur *ROBOTOPIA* (décrit

en détail dans le chapitre chapitre 6) se basant sur le framework de programmation orienté institution Utopia [Schmitt 10], et prenant en considération les différentes spécifications du modèle organisationnel multi-agents $\mathcal{M}\text{OISE}^{Inst}$. Il se base sur la théorie des graphes récursifs [Harel 98], pour faciliter le partage des informations entre les agents, ce genre de graphes a la possibilité d'en obtenir des sous-ensembles (des sous-graphes récursifs), et permet donc le stockage de l'ensemble des spécifications de l'Institution Électronique ($\mathcal{M}\text{OISE}^{Inst}$), et le bon partage de l'information entre agents pour que chaque agent puisse demander et recevoir les informations lui étant nécessaire (cf. Annexe B).

Ces graphes récursifs sont considérés comme une sorte de structure de données qui répond particulièrement aux besoins sous-jacents de $\mathcal{M}\text{OISE}^{Inst}$. Ils sont la plupart du temps récursifs : les groupes peuvent inclure d'autres groupes, les missions peuvent inclure d'autres missions, etc.

En outre, l'extraction de sous-graphes récursifs rend le partage des données plus facile en utilisant Utopia.

Utopia se compose de cinq entités principales (cf. figure 4.10) : le *Superviseur*, le *RoleManager*, le $ContextManager$, le $GoalManager$ et l'$AgentUtopia$, nous décrirons chacun d'eux dans ce qui suit.

4.2.1 Le *ContextManager*

Il vise à gérer tous les contextes (états) dans lesquels le système peut être trouvé conformément à la spécification contextuelle (cf. figure 4.11). Il peut être considéré comme une machine parallèle à états finis exécutant un thread pour chaque scène. Une scène est un ensemble de contextes, des transitions et éventuellement d'autres sous-scènes.

Le thread de la scène commence avec le sommet initial et passe à l'état suivant si le message reçu est une transition appropriée. Les commandes exécutées par le $ContextManager$ sont principalement des transitions et des requêtes pour tous les états courants. Enfin, lorsque chaque thread a terminé son exécution (lorsque l'état final de la SC est atteint), le $ContextManager$ informe tous les agents que l'instanciation de la spécification est terminée.

4.2.2 Le *RoleManager*

Il permet la gestion des rôles de la spécification structurelle (cf. figure 4.12) qui consiste à :

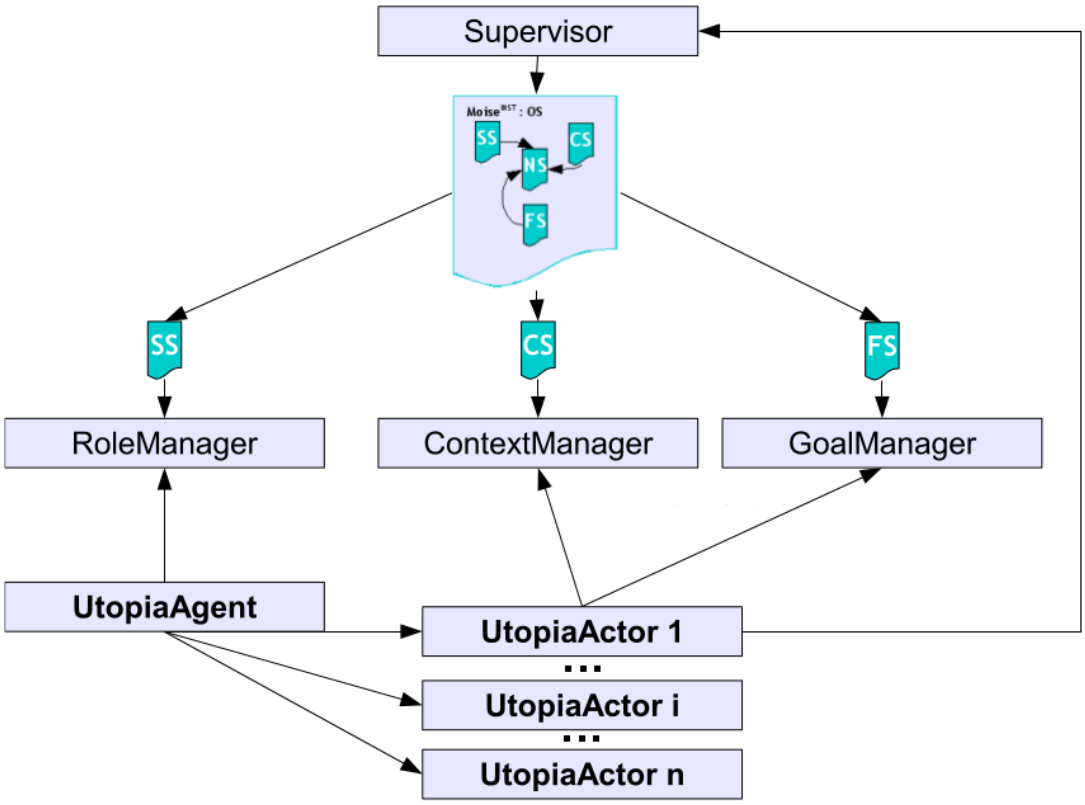


FIGURE 4.10 – Architecture du framework de programmation orientée institution Utopia [Schmitt 10].

— trouver une répartition des rôles (quels rôles seront joués par quels agents) et retourner le nombre d'agents nécessaires pour instancier la spécification structurelle.
— fournir tous les rôles qu'un agent doit jouer ou refournir la liste même encore une fois si un agent donné le demande à plusieurs reprises.
— indiquer si tous les agents jouent les rôles appropriés.

Trouver une bonne répartition des rôles implique de résoudre le problème (ISS [1]) :

1. Instanciation de la Spécification Structurelle.

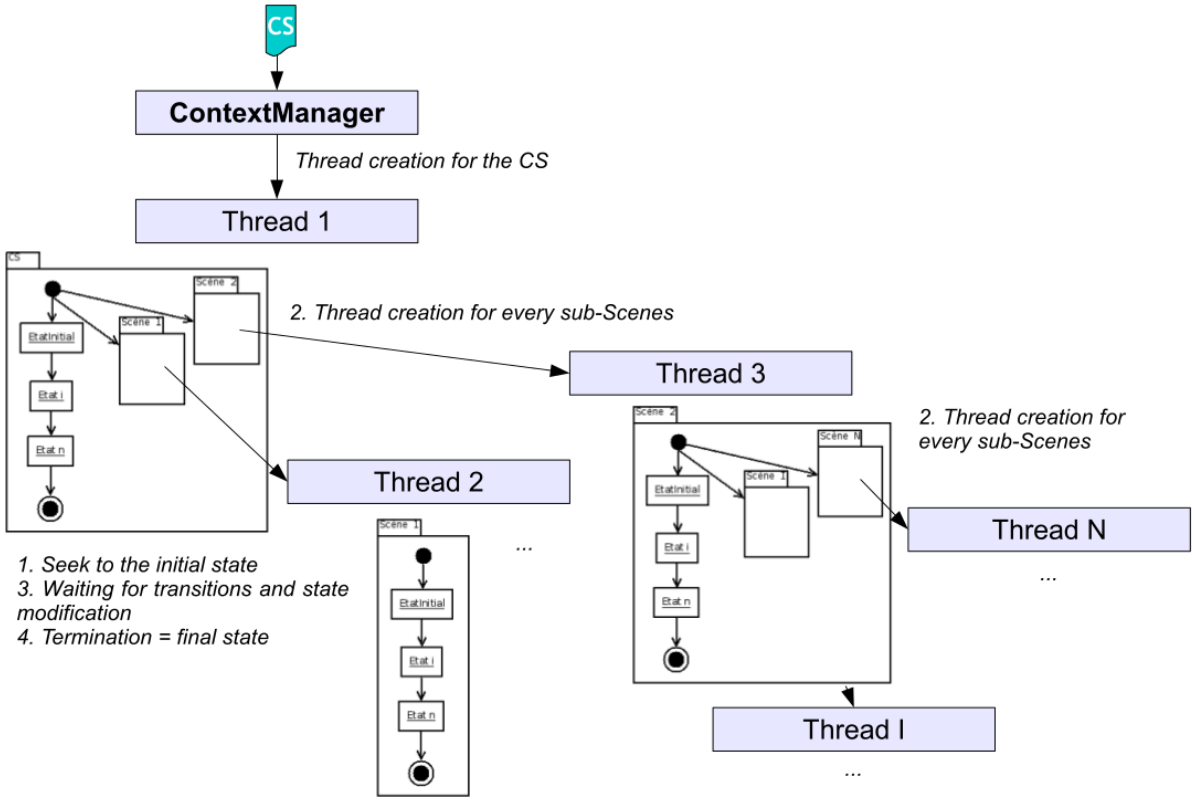


FIGURE 4.11 – Principe de fonctionnement du *ContextManager*.

$$
\begin{cases}
minA(X_G) \\
u.c \\
(1)\ \forall r \in \{1,..,n_G\}, l_r \leq \sum_{a=1}^{p} x_{(r,a)_G} \leq h_r \\
(2)\ \forall r_1 \in \{1,..,n_G\}, \forall r_2 \in \{1,..,n_G\}, \forall a \in \{1,..,p\}, \\
x_{(r_1,a)_G} x_{(r_2,a)_G} c_{(r_1,r_2)_G} - x_{(r_1,a)_G} x_{(r_2,a)_G} = 0 \\
(3)\ l_G \leq A(X_G) + \sum_{i=1}^{q_G} minA(X_{G_i}) \leq h_G \\
(i)\ \forall r \in \{1,..,n_G\}, l_{rG} \in \mathbb{N}, h_{rG} \in \mathbb{N} \\
(ii)\ l_G \in \mathbb{N}, h_G \in \mathbb{N} \\
(iii)\ \forall r \in \{1,..,n_G\}, \forall a \in \{1,..,p\}, x_{(r,a)_G} \in \mathbb{N}, 0 \leq x_{(r,a)_G} \leq 1
\end{cases}
$$

Où $A(X) = \sum_{a=1}^{p} O(x_{1,a},...,x_{n,a})$ représente le nombre d'agents jouant un rôle, X une matrice binaire où les rôles sont stockés dans les lignes et les agents dans les

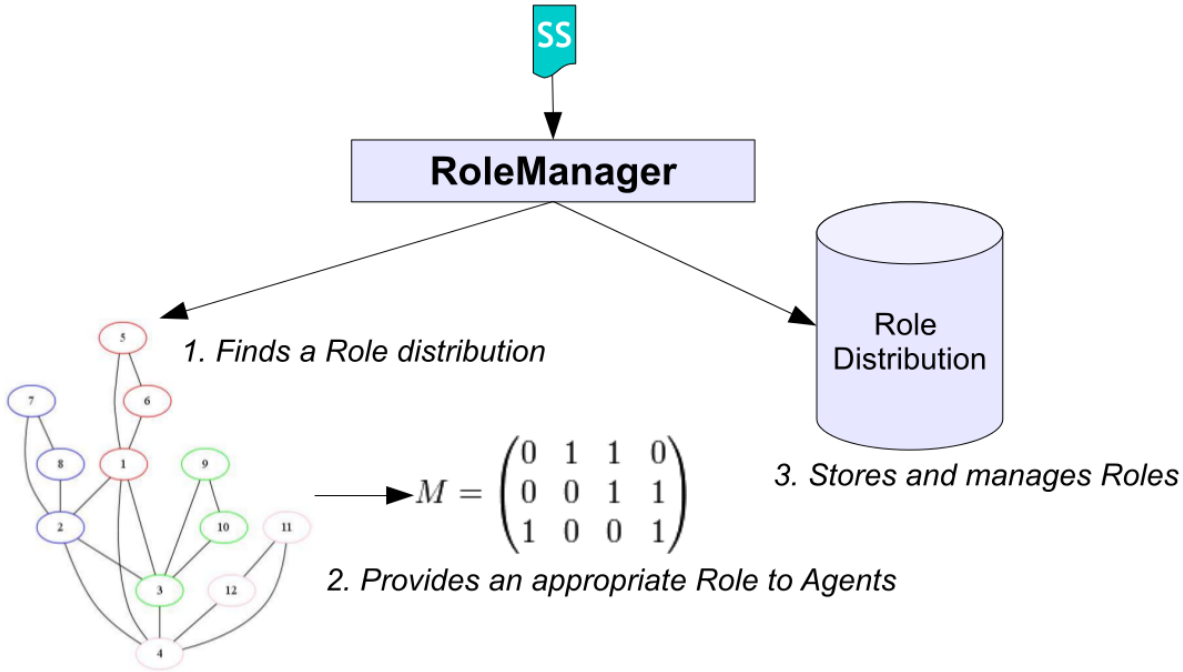


FIGURE 4.12 – Principe de fonctionnement du *RoleManager*.

colonnes, O une fonction renvoyant le résultat d'un OU logique entre plusieurs variables binaires :

$$\begin{cases} O(x_1, ..., x_n) = x_n + (O(x_1, ..., x_{n-1})(1 - x_n)) \\ O(x_1, x_2) = x_1 + x_2 - x_1.x_2 \end{cases}$$

Les contraintes sont les suivantes :

— le numéro de l'instanciation de chacun des n_G rôles, r doit être compris entre l_r et h_r (la cardinalité associée à r).
— un agent doit jouer un rôle unique, ou toutes les paires de rôles (r_1, r_2) qu'il joue doit avoir un lien de compatibilité $c_{(r_1,r_2)_G} = 1$.
— le nombre total d'agents jouant un rôle dans le groupe G et tous ses sous-groupes q_G doit respecter la cardinalité du G-Groupe (l_G, h_G).

Ce problème mathématique a été résolu par une décomposition polynomiale en un problème à k-coloration récursif [Beigel 89], ce qui permet de maximiser le nombre de rôles joué en parallèle par un agent.

4.2.3 Le *GoalManager*

Le $GoalManager$ exerce principalement une correspondance entre l'ensemble des goals de la spécification fonctionnelle et les instances concrètes des classes implémentant les actions associées à chaque goal dans un environnement Java (cf. figure 4.13). Lorsque ce mappage est effectué, il peut fournir des instances de classes après les requêtes de chaque agent.

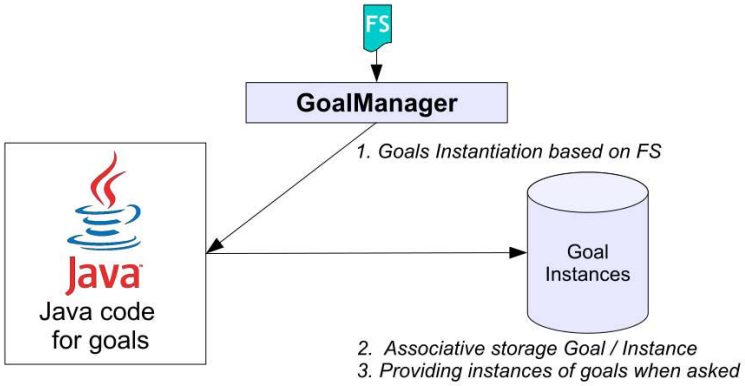


FIGURE 4.13 – Principe de fonctionnement du $GoalManager$.

4.2.4 Le *Superviseur*

Le superviseur est au sommet de la hiérarchie du système (cf. figure 4.14), ses objectifs sont les suivants :

— création d'un graphe récursif basé sur la spécification de l'organisation (SO) (structure XML).
— création de trois managers : $ContextManager$, $RoleManager$ et $GoalManager$.
— envoi de sous-graphes de la SO aux managers correspondants, principalement la SC pour le $ContextManager$, la SS pour le $RoleManager$ et la FS pour le $GoalManager$.
— faire le lien avec la spécification normative (SN) qui est nécessaire à la gestion de l'ensemble de la spécification de l'organisation.

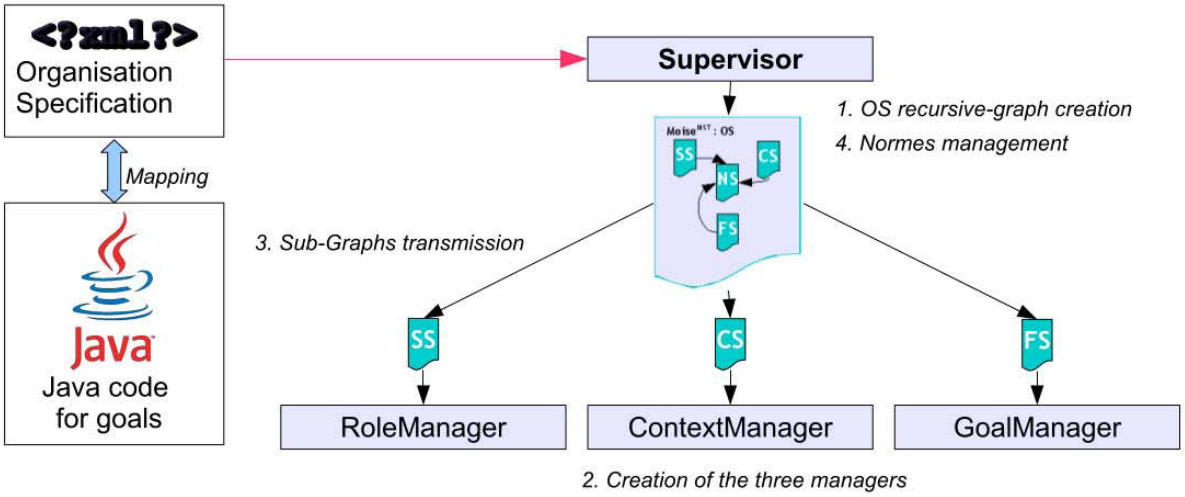


FIGURE 4.14 – Principe de fonctionnement du *Superviseur*.

4.2.5 L'*Agent Utopia*

L'Agent Utopia est un agent générique (cf. Figure 4.15) susceptibles de se spécialiser de façon indépendante afin de se comporter différemment selon le cas d'utilisation, cela permet d'obtenir des comportements intelligents. Pour arriver à cette fin, cet agent récupère un rôle et agit (en exécutant les goals appropriés grâce au *GoalManager*) selon les contextes de l'institution et des normes qui lui sont appliquées, ce qui donne une auto-organisation des agents.

De cette manière Utopia permet une instanciation facile d'une organisation afin d'obtenir une entité dynamique dans laquelle les agents peuvent évoluer (les règles de l'organisation restent statiques). En effet, grâce aux graphes récursifs, toutes les données homogènes sont stockées dans une unique structure récursive, permettant de distribuer facilement le partage d'informations entre les agents d'Utopia.

4.3 Conclusion

La particularité de notre architecture de contrôle réside en sa hiérarchisation et surtout en la méthode utilisée dans son haut niveau, pour permettre une bonne autonomie et une bonne organisation du système multi-robots. En effet, on l'a organisé en deux principaux niveaux, le haut niveau appelé *Niveau Organisation*

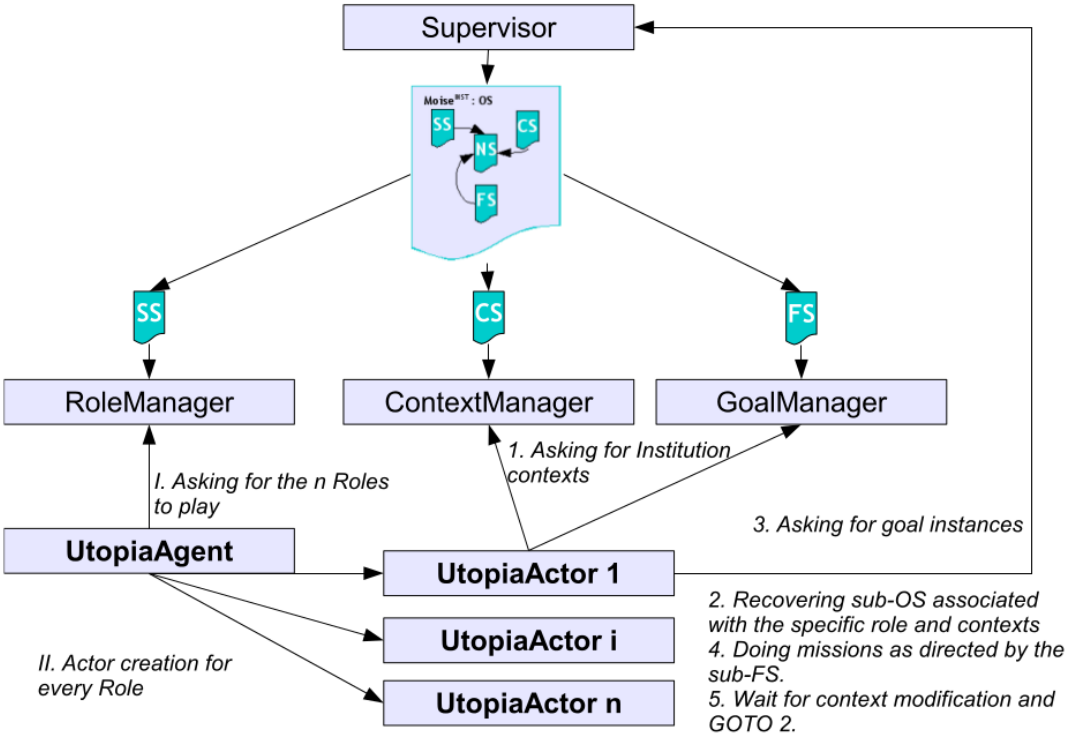


FIGURE 4.15 – Principe de fonctionnement de l'*AgentUtopia*.

et le bas niveau appelé *Niveau de Contrôle*, cette hiérarchisation nous a permis d'implémenter un système puissant de coordination basé sur les systèmes multi-agents, c'est en fait un modèle d'organisation et d'institution d'agent $\mathcal{M}$OISE^{Inst} permettant une facilité et une grande flexibilité du contrôle et de coordination des différents agents. Grâce à ses différentes spécifications on a pu décrire avec précision les rôles, les goals, les contextes les fonctionnalités et les normes propres à chaque agent de l'organisation.

On a pu ainsi associer chaque agent de l'organisation à un robot du système multi-robots mis à notre disposition, tous les agents/robots on la même architecture de contrôle précédemment décrite, avec les mêmes spécifications, normes, contextes et fonctionnalités, la seule différence qu'il peut y avoir réside dans le mode de fonctionnement de chacun.

Dans le chapitre suivant nous présenterons une illustration du fonctionnement de l'architecture de contrôle proposée, nous verrons toutes les interactions existantes entre le niveau de contrôle/commande et le niveau d'organisation, ainsi que les différents contextes de fonctionnements possibles et comment y aboutir.

Chapitre 5

Illustration des différents modes de fonctionnement de l'architecture de contrôle/commande proposée

Ce chapitre représente la viabilité de la mise en commun des différentes composantes de l'architecture de contrôle proposée (cf. chapitres 3 et 4), soit le haut et le bas niveau, on y verra la relation qui existe entre le niveau contrôle/commande et le niveau organisation.

Il présente aussi l'interconnexion entre de bas et le haut niveau, ce qui permet l'exécution de l'architecture de contrôle/commande proposée dans différents contextes.

5.1 Introduction

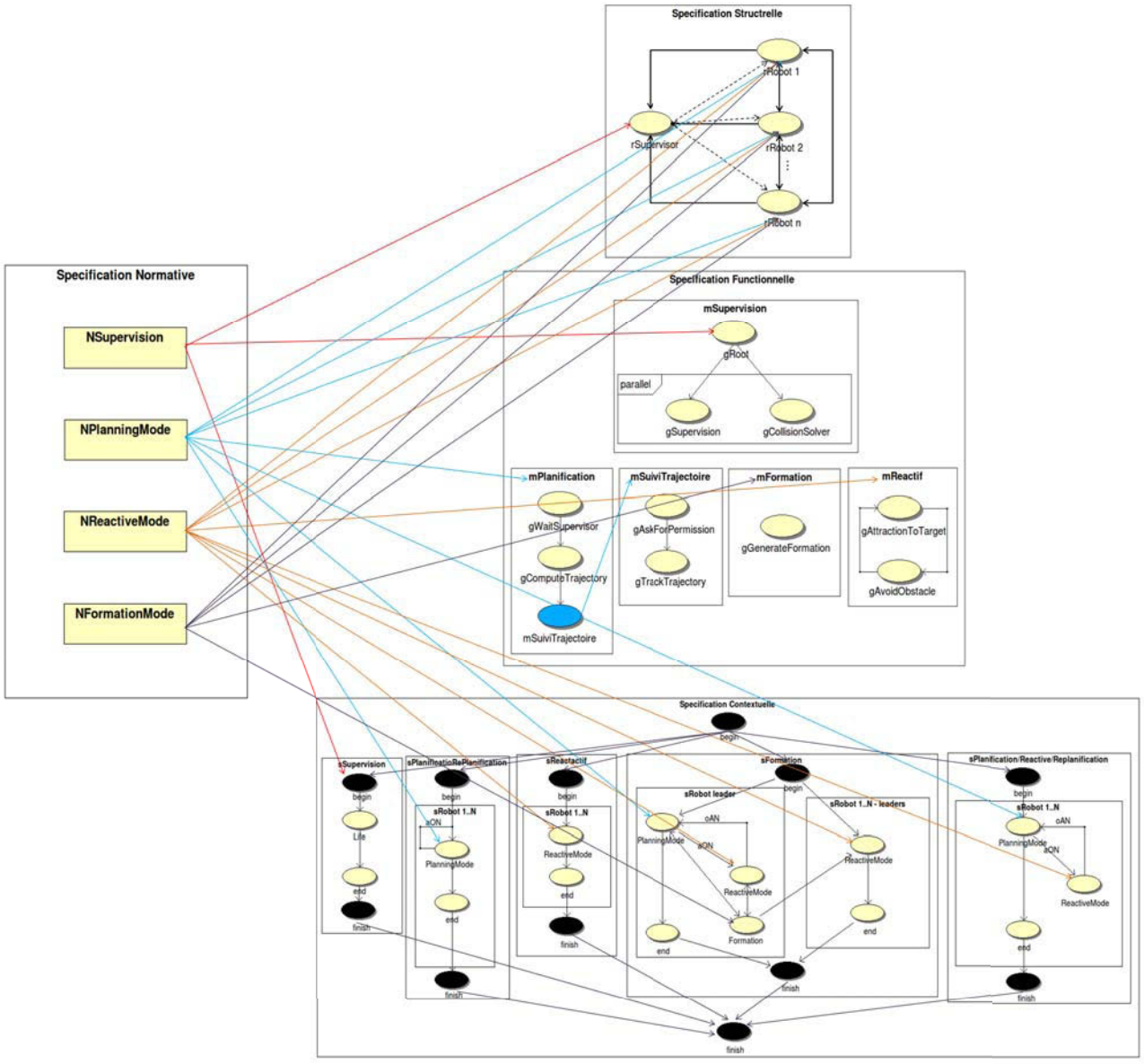


FIGURE 5.1 – Vue globale de la spécification de l'organisation du niveau organisation.

Pour pouvoir comprendre le fonctionnement de l'architecture de contrôle/-commande proposée ainsi que les différentes interconnexions qui y existent, on va

représenter tous les liens existants entre les différentes spécifications du niveau organisation par la figure 5.1, où on voit bien que la spécification normative est l'acteur principal qui relie toutes les spécifications de l'organisation (cf. chapitre 4), ce qui nous permet de contrôler le type de navigation qu'on veut obtenir.

Grâce à cette structuration, le niveau organisation envoi les informations de navigation au niveau du contrôle/commande, et spécialement au module de sélection d'action hiérarchique qui se charge de faire appel aux modules et contrôleurs concernés pour la navigation autonome du système multi-robots (cf. figure 3.6). L'architecture de contrôle proposée présente alors un genre de framework ouvert qui peut s'adapter à tout type de navigation mono ou multi-robots.

Dans le cas de notre exemple d'application de l'architecture de contrôle/commande proposée dans l'industrie, on a pris celui du transport et déplacement de marchandise dans un entrepôt de stockage, notre architecture pourra avoir trois types de fonctionnements différents :

— centralisées : dans le cas où l'entrepôt est assez encombré, et où les situations de conflits éventuels sont peu fréquentes et sont gérable par le superviseur, il est donc important de planifier les missions du début à la fin pour plus de sécurité.
— distribuées : dans le cas où notre entrepôt est peu rempli avec un petit trafic de robots ou encore très encombré, ces des missions assez facile à exécuter en mode totalement distribué qui permet de gagner en énergie.
— centralisées/distribuées : dans le cas où on a un grand trafic des robots dans l'entrepôt encombré (environnement hautement dynamique), et où les situations de conflits sont très fréquentes, on a donc besoin de sécuriser les missions par la planification mais aussi d'alléger la tâche du superviseur en munissant les robots d'un contrôle distribué qui leurs permet de gérer les événements inattendu.

En outre, grâce à l'efficacité et la souplesse du modèle d'organisation multi-agents proposé, on a pu tester plusieurs exemples représentant chaque type de navigation réalisable par l'architecture de contrôle proposée, soit :

— navigation centralisée : algorithme utilisant la planification et la re-planification supervisée.
— navigation centralisée/distribuée :
  — algorithme utilisant la planification supervisée, et l'évitement réactif d'obstacles (non supervisé) pour gérer les événements inattendus.
  — algorithme de navigation en formation, avec un robot leader utilisant la planification supervisée par l'agent superviseur, et des robots suiveurs en mode navigation réactive utilisant le contrôleur de suivi de cible dynamique (cf. section 3.3.2).

— navigation distribuée : Algorithme de navigation réactive avec cible statique, muni du contrôleur d'évitement réactif d'obstacles.

Nous décrirons dans ce qui suit le fonctionnement des différents types de navigation ainsi que celui de chacun de ces algorithmes, en mettant en évidence les parties mises à contribution de l'architecture de contrôle proposée. Pour simplifier cette description nous avons choisi un système multi-robots à trois entités robotiques.

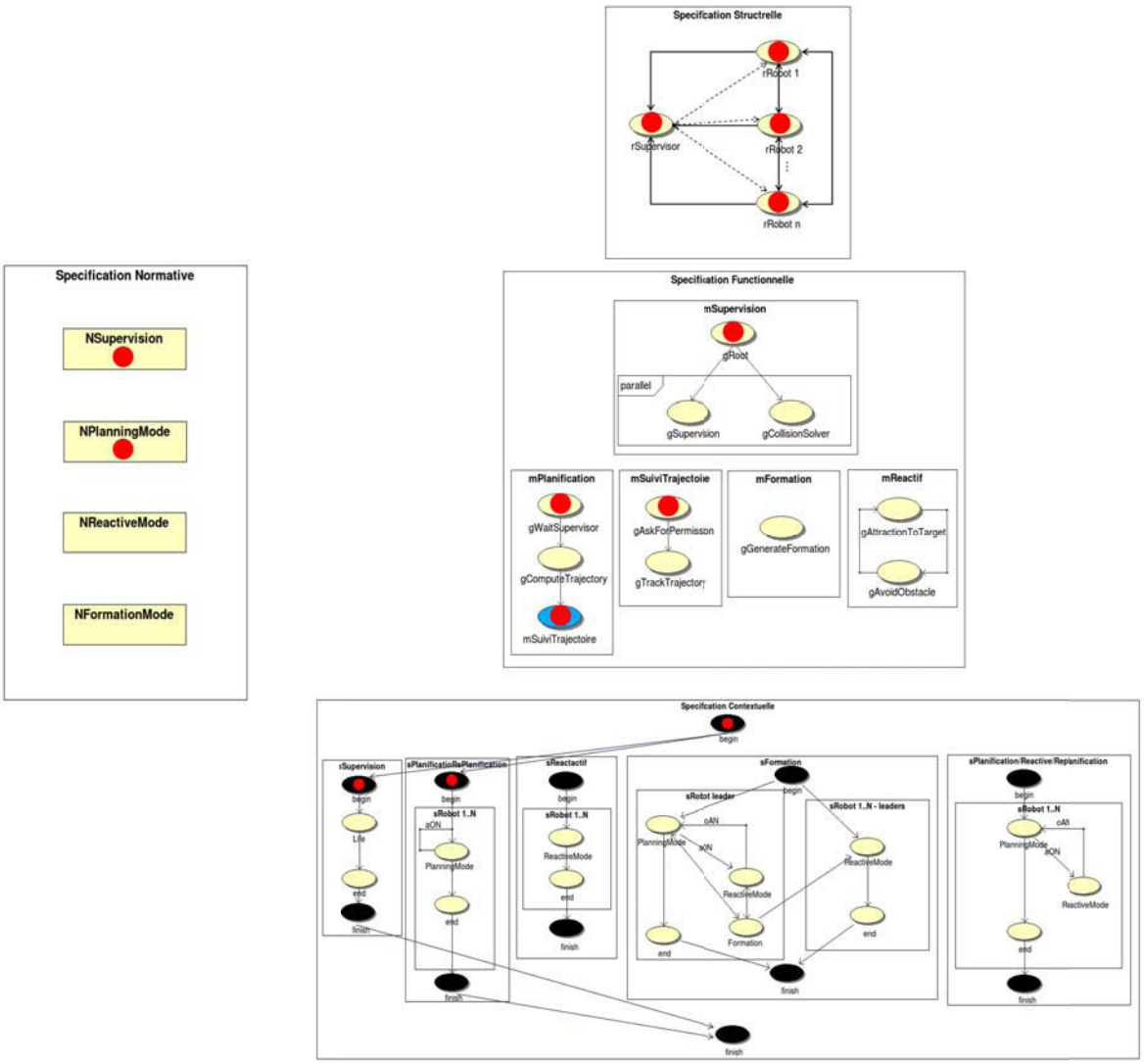


FIGURE 5.2 – Fonctionnement du niveau organisation dans le contexte d'une navigation centralisée (les points rouges représentent les blocs actifs).

5.2 Navigation de plusieurs robots en milieu confiné : Approche centralisée

Comme précédemment mentionné, le comportement centralisé est représenté, dans l'architecture de contrôle proposée, par l'algorithme de planification et de re-planification supervisé par l'agent superviseur.

La figure 5.2 représente les parties en activité dans le niveau organisation. En effet, comme on peut le constater on a deux normes de la spécification normative qui ont une activité parallèle :

— la norme $NSupervision$ (cf. figure 5.2) : elle est responsable du comportement de l'agent superviseur dans le contexte $sSupervision$ de la spécification contextuelle, après la création et l'initialisation du superviseur. Elle lui donne le rôle $rSuperviseur$ de la spécification structurelle et le force à exécuter la mission $mSupervision$ de la spécification fonctionnelle, en ayant comme deux objectifs exécutés en parallèle les goals : $gSupervision$ et $gCollisionSolver$.

— la norme $NPlanningMode$ (cf. figure 5.2) : qui est responsable du comportement de tous les autres agents/robots du système multi-agents/robots. Après leur activation et initialisation, elle se connecte à chacun d'eux et les met dans le contexte $sPlanificationRePlanification$, elle les force ainsi à exécuter la mission $mPlanification$, ils se mettent alors en relation avec l'agent superviseur à travers le goal $gWaitSupervisor$ pour superviser leur navigation.

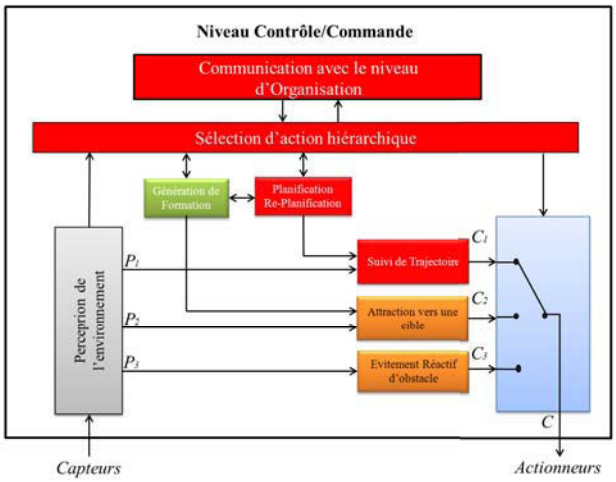


FIGURE 5.3 – Fonctionnement du niveau de contrôle dans le contexte d'une navigation centralisée (les blocs en rouge sont les blocs actifs).

Chaque agent/robot, une fois mis en relation avec le superviseur, active le reste des goals de sa mission $mPlanification$ en activant la planification de sa trajectoire prenant en compte les obstacles existants dans l'environnement de navigation. Dès qu'il a l'autorisation du superviseur il la suit en activant la mission $mSuiviTrajectoire$. Si un agent/robot reçoit le signal de la détection d'un obstacle inattendu, il refait aussitôt appel à la mission $mPlanification$ pour trouver une autre trajectoire plus sûre, qui sera à son tour supervisée par l'agent superviseur. La figure 5.4 montre une simulation de ce type de navigation, où on voit une navigation de trois robots mobiles avec une gestion centralisée des obstacles inattendus. Les simulations des autres types de fonctionnement de l'architecture de contrôle/commande proposée en mode centralisé sont présentées dans le chapitre 7.

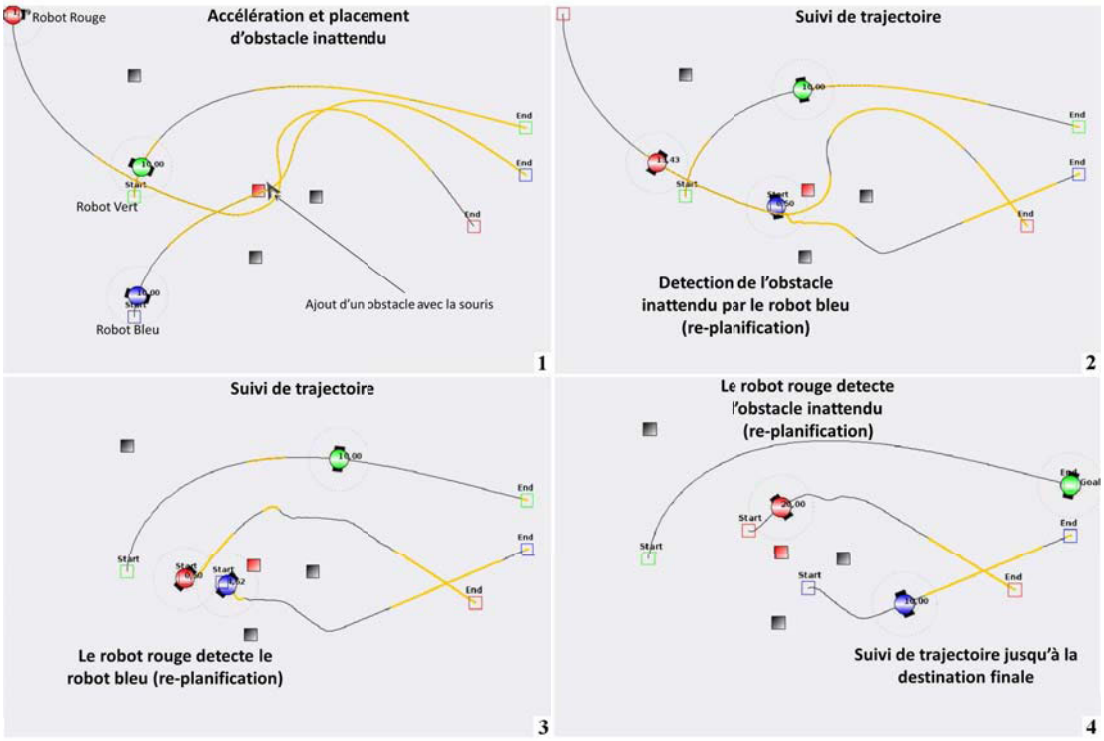


FIGURE 5.4 – Simulation du scénario Navigation avec événements inattendus.

Concernant le niveau de contrôle de l'architecture de contrôle proposée, il est en lien direct avec le niveau organisation. En effet, le niveau contrôle envoi en temps réel la liste des goals à être exécutés au module de sélection d'action hiérarchique (cf. figure 3.6), qui se charge d'activer les contrôleurs et les modules

en question. Dans le contexte de la navigation centralisée, le module de sélection d'action hiérarchique active le module de planification de trajectoire, une fois la trajectoire validée, il active le contrôleur de suivi de trajectoire pour pouvoir la suivre (cf. figure 5.3), et ainsi de suite.

5.3 Navigation de plusieurs robots en milieu confiné : Approche distribuée

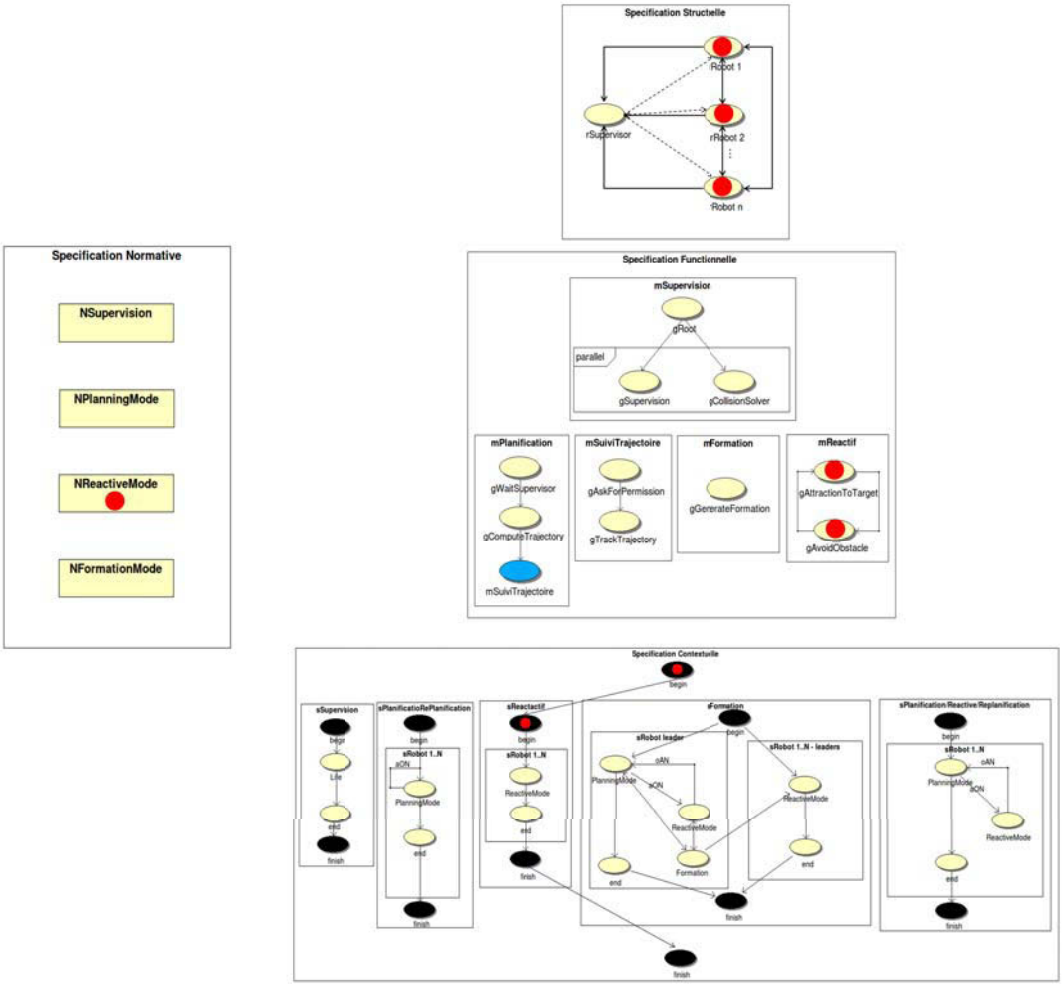


FIGURE 5.5 – Fonctionnement du niveau organisation dans le contexte d'une navigation distribuée.

Dans l'architecture de contrôle proposée (cf. figure 3.6), le comportement de la navigation distribuée est représenté par un algorithme de navigation réactive, où tous les agents/robots de l'organisation sont indépendants, pas d'agent superviseur ni de priorité entre les robots (pas de leader ni de suiveurs). Les robots sont donc complètement autonomes en mode navigation réactive avec une cible statique à atteindre dans l'environnement.

On a dans ce cas de figure une seule norme qui rentre en jeu (cf. figure 5.5), c'est la norme $NReactiveMode$, qui est reliée à tous les agents/robots de l'organisation. Elle les force donc à exécuter la mission $mReactif$ dans le contexte $sReactif$. On a alors une attraction des agents/robots vers la cible statique propre à chacun par l'exécution du goal $gAttractionToTarget$, puis pour résoudre les conflits éventuels qui peuvent survenir pendant la navigation (soit des risques de collision entre les robots, soit les risques liés à l'apparition d'un obstacle inattendu dans l'environnement). Ils font appel au goal $gAvoidObstacle$, pour résoudre ces deux types de conflits possibles. Une fois que le conflit résolu on repasse au goal $gAttractionToTarget$ pour que l'agent/robot continu son chemin vers sa cible.

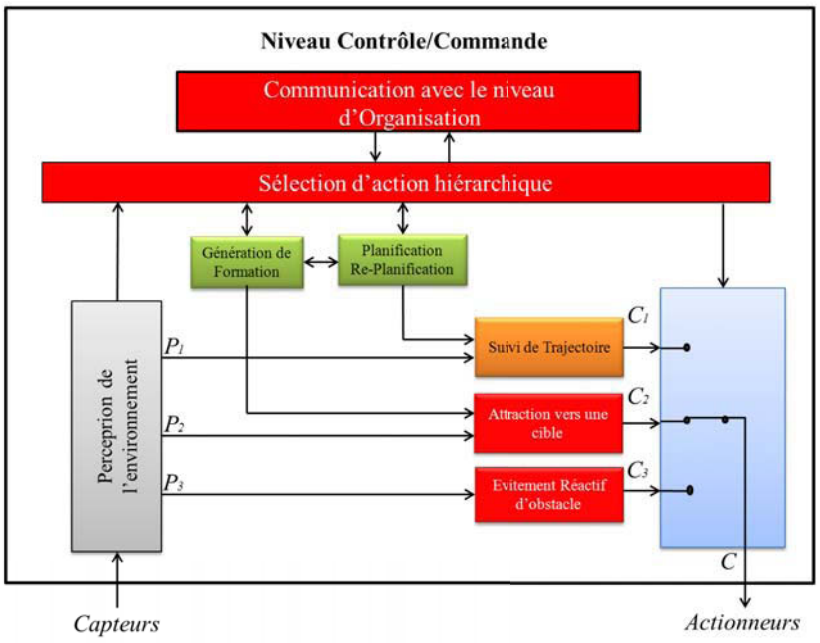


FIGURE 5.6 – Fonctionnement du niveau de contrôle dans le contexte d'une navigation distribuée.

Comme déjà expliqué plus haut, ces informations haut (activation des différentes missions et goals, etc.) niveau sont envoyés en temps réel au module de

sélection d'action hiérarchique du niveau de contrôle, qui se charge d'activer les modules et/ou les contrôleurs correspondant à ce type de navigation. En l'occurrence dans le cas de la navigation distribuée (cf. figure 5.6), ce module active le contrôleur de suivi de cible avec lequel le robot commence la navigation, puis dès qu'il y a détection d'obstacles inattendus ou d'un robot gênant son passage, le module de sélection d'action hiérarchique bascule sur le contrôleur d'évitement réactif d'obstacle, puis une fois cette tâche terminée il rebascule sur le contrôleur d'attraction vers une cible jusqu'à atteindre cette dernière.

On obtient ainsi une navigation distribuée de plusieurs robots au sein de l'architecture de contrôle proposée, ces robots sont complètement autonomes dans un environnement inconnu avec des destinations différentes, et qui doivent les atteindre de la manière la plus sûre qu'il soit. Plus de résultats dans un contexte distribué seront présentés dans le chapitre 7.

5.4 Navigation de plusieurs robots en milieu confiné : Approche centralisée/distribuée

Le comportement centralisé/distribué est représenté, dans l'architecture de contrôle proposée, par deux comportements : le comportement de planification et de re-planification supervisée par l'agent superviseur avec évitement réactif d'obstacles, et le comportement de navigation en formation avec un leader en mode planification et re-planification supervisé et des followers en mode navigation réactive.

Dans ce qui suit, nous commençons par présenter le premier comportement représentant la navigation centralisée/distribuée au sein de l'architecture de contrôle/commande. Le comportement planification et re-planification supervisée avec évitement réactif d'obstacles comporte deux parties, l'une est centralisée représentée par la planification et la re-planification supervisées, et l'autre distribuée représentée par l'évitement réactif d'obstacles inattendus.

Planification et re-planification supervisée avec évitement réactif d'obstacles

La figure 5.7 représente les parties en activité dans le niveau organisation pour l'exécution de ce comportement. En effet, comme on peut le constater on a trois normes en activité dans la spécification normative :

— la norme *NSupervision* : elle a le même comportement précédemment décrit (cf. section 5.2), elle fait donc le lien entre l'agent superviseur et

les composantes de toutes les spécifications de l'organisation. Cette norme permet donc d'exécuter la mission *mSupervision* qui lui permettra de superviser la navigation des agents/robots et résoudre les conflits avec les goals *gSupervision* et *gCollisionSolver* (cf. section 4.1.2).

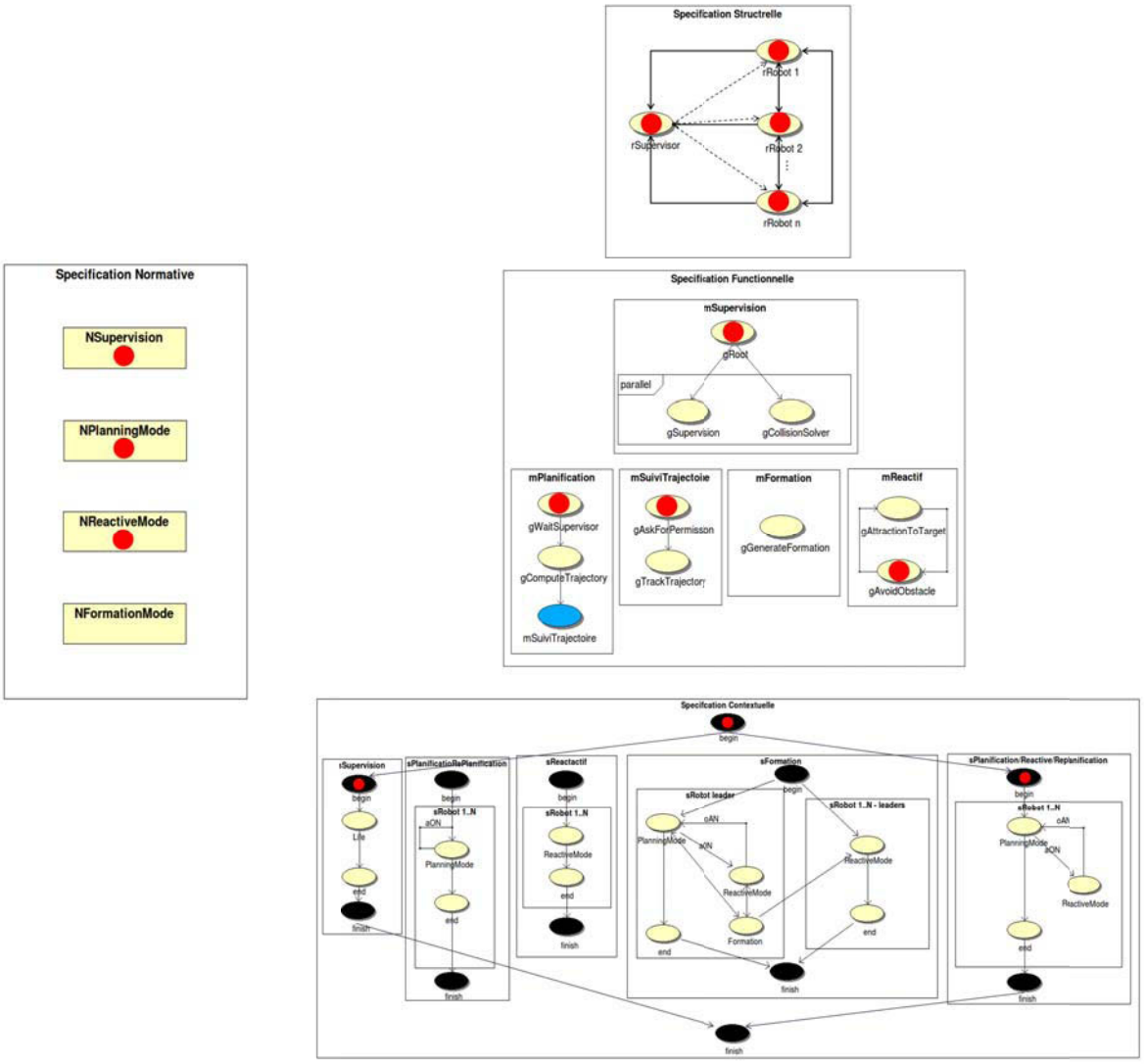


FIGURE 5.7 – Fonctionnement du niveau organisation dans le contexte d'une navigation centralisée/distribuée.

— la norme $NPlanningMode$: est connectée à tous les agents/robots et les force à se mettre dans le contexte $sPlanificationReactiveReplanification$. Les agents/robot commencent donc par faire appel à la mission $mPlanification$

comme précédemment expliqué (cf. section 5.2), ils se mettent alors en relation avec le superviseur lui transmettant leurs trajectoires respectives pour la traiter et les autoriser ou non à la suivre, en exécutant la mission $mSuiviTrajectoire$. Dans le cas où un ou plusieurs agents/robots rencontrent un obstacle inattendu, la norme $NReactiveMode$ rentre en jeu, après que la transition aON (Avoid Obstacle N) (cf. figure 5.7) soit passée.

— La norme $NReactiveMode$: est aussi connectée à tous les agents/robots, elle leurs permet d'éviter l'obstacle inattendu en les forçant à exécuter la mission $mReactif$ et plus précisément le goal $gAvoidObstacle$ dans le contexte $sPlanificationReactiveReplanification$.

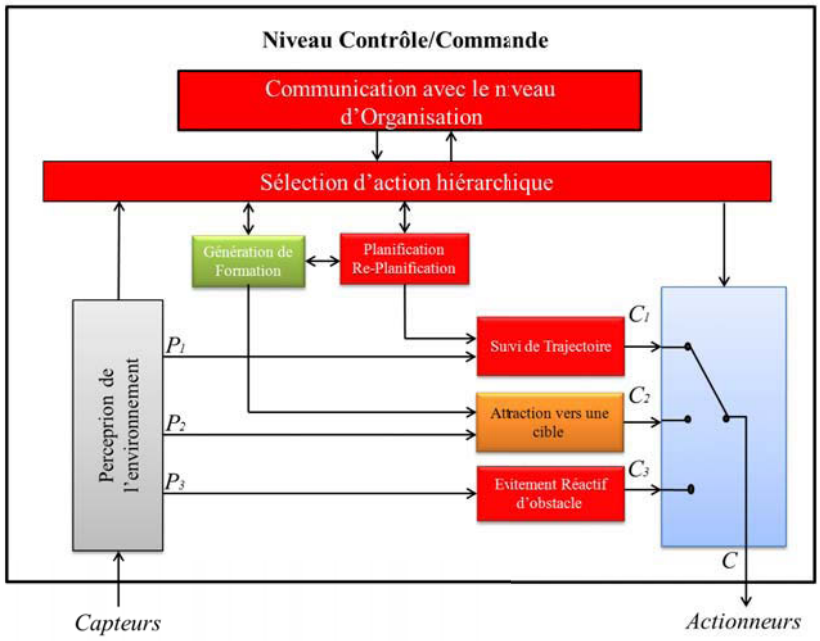


FIGURE 5.8 – Fonctionnement du niveau de contrôle dans le contexte d'une navigation centralisée/distribuée.

Une fois cette tâche finie une autre transition se fait (oAN Obstacle Avoided N) pour passer à la re-planification. On a alors la norme $NPlanningMode$ qui permet à l'agent/robot en question de refaire une deuxième planification en collaboration avec le superviseur pour qu'il puisse continuer son chemin vers sa cible.

La figure 5.8 représente les modules et les contrôleurs activés par le module de sélection d'action hiérarchique, dans ce type de navigation. Cet algorithme présente alors un comportement centralisé et distribué en même temps. Dans le

cas où les agents/robots ne rencontrent aucun obstacle inattendu on obtiendra un comportement purement centralisé.

Le deuxième comportement centralisé/distribué couvert par notre architecture de contrôle/commande proposée est celui de la navigation en formation, il comporte une partie centralisée représentée par le comportement de l'agent/robot leader qui effectue une planification et re-planification supervisée, et une partie distribuée représentée par le comportement des agents/robots suiveurs qui sont en mode réactif. Dans ce qui suit nous présentons le fonctionnement détaillé de ce mode de fonctionnement.

Navigation en formation

Pour effectuer une navigation en formation par l'architecture de contrôle proposée, on fait appel à toutes les normes existantes dans notre modèle d'organisation, soit quatre normes, pour gérer au mieux notre système multi-robots. On peut alors différencier entre deux catégories d'agents/robots :

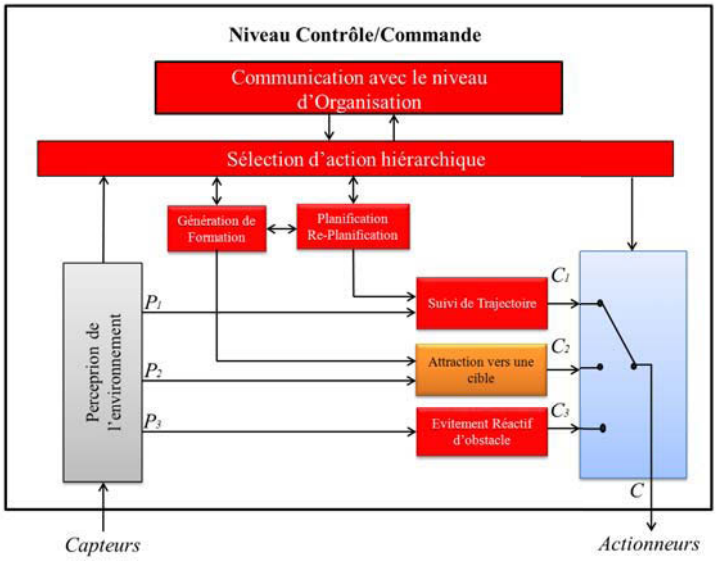


FIGURE 5.10 – Fonctionnement du niveau de contrôle dans le contexte d'une navigation centralisée/distribuée de l'agent/robot leader.

1. Les agents/robots leader avec quatre normes en activité (cf. figure 5.9) :
   — la norme *NSupervision* qui a le même comportement précédemment décrit (cf. section 5.2), elle permet à l'agent superviseur de superviser la navigation de l'agent robot leader dans le contexte *sFormation*.

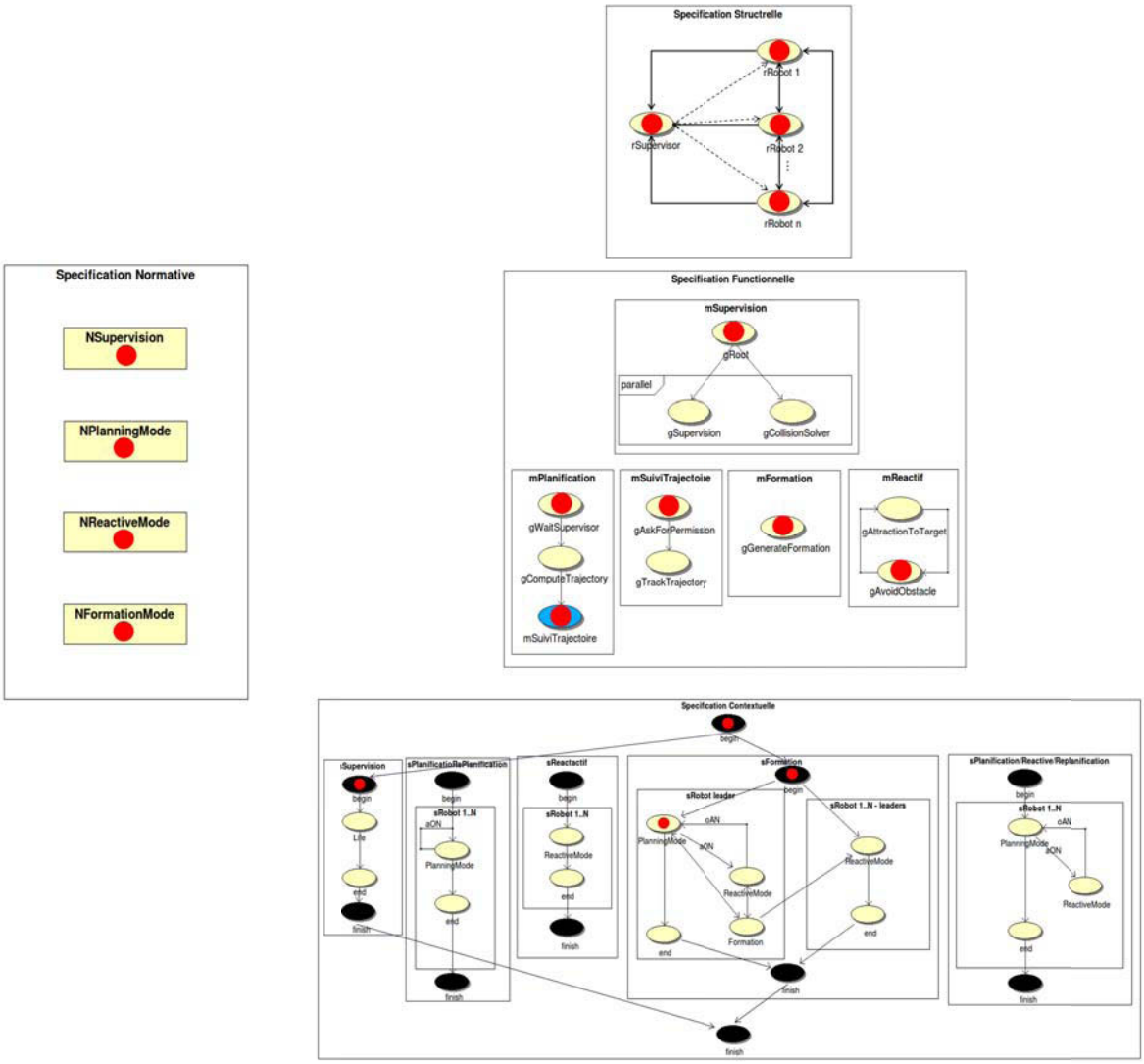


FIGURE 5.9 – Fonctionnement du niveau organisation dans le contexte d'une navigation centralisée/distribuée de l'agent/robot leader.

— la norme $NPlanningMode$ qui permet au leader de planifier ou replanifier sa trajectoire (cf. section 5.2) et de la suivre en étroite collaboration avec l'agent superviseur dans le contexte $sFormation$.
— la norme $NFormationMode$ est reliée à l'agent/robot leader le mettant dans le contexte $sFormation$, et le forçant à exécuter la mission $mFormation$. Le goal $gGenerateFormation$ lui permet de générer les positions de formation des robots suiveurs pour chaque position du robot

leader.

— la norme $NReactiveMode$ permet à l'agent/robot leader d'éviter d'une manière réactive des obstacles inattendus.

La figure 5.10 représente une illustration du niveau de contrôle du robot leader ainsi que ses différents modules en activité.

2. les agents/robots suiveurs avec deux normes activés (cf. figure 5.11) :

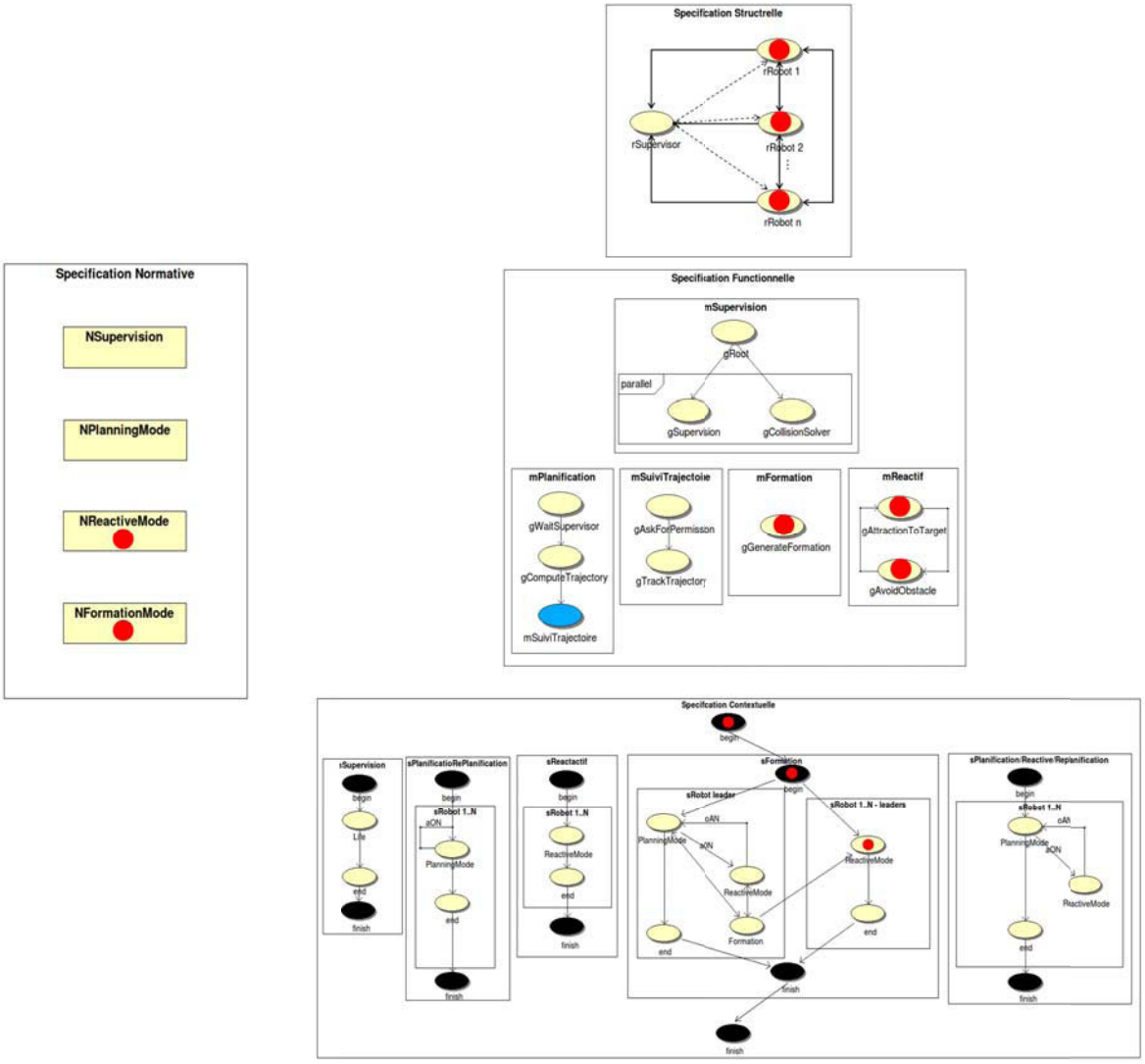


FIGURE 5.11 – Fonctionnement du niveau organisation dans le contexte d'une navigation centralisée/distribuée des agents/robots suiveurs.

— la norme $NFormationMode$ met les agents/robots dans le contexte

$sFormation$, puis grâce au goal $gGenerateFormation$ leur envoi à chaque évolution de l'agent/robot leader la position de la nouvelle cible à suivre.

— la norme $NReactiveMode$ est connectée à tous les agents/robots suiveurs, dans le contexte $sFormation$, pour les forcer à effectuer la mission $mReactif$. On obtient alors un suivi de cible dynamique (grâce à $gAttractionToTarget$) envoyé par l'agent leader, pour atteindre leurs positions dans la formation, et grâce à $gAvoidObstacle$ ils peuvent éviter les obstacles dans leur chemin.

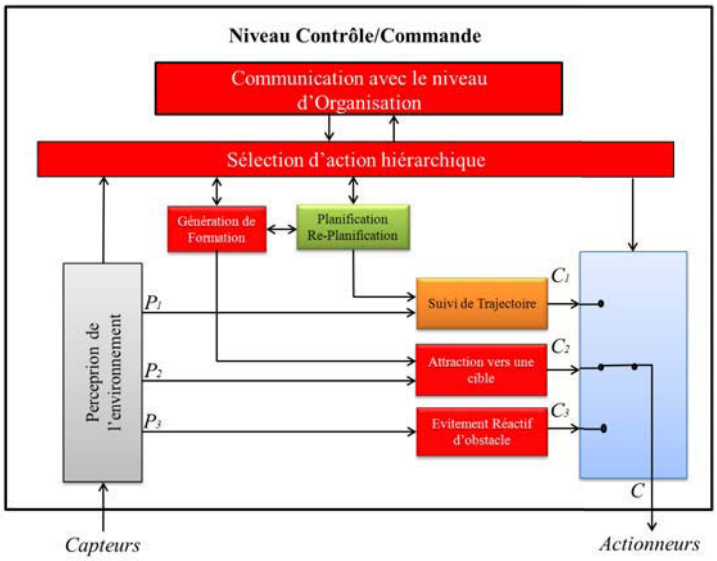


FIGURE 5.12 – Fonctionnement du niveau de contrôle dans le contexte d'une navigation centralisée/distribuée des agents/robots suiveurs.

La figure 5.12 représente une illustration du niveau de contrôle des robots suiveurs ainsi que les différents modules en activité. Ainsi, en regroupant les architectures des deux catégories d'agents/robots, on obtient une navigation en formation performante d'un point de vue flexibilité et simplicité de reconfiguration tant dans le maintient de la formation que dans la navigation multi-robots, contenant un robot leader suivi par un ou plusieurs suiveurs. Le tout représentant une forme géométrique précise (triangle, carré, ligne, etc.). Ce fonctionnement démontre non seulement l'aspect centralisé/distribué de l'architecture de contrôle proposée, mais aussi l'aspect **hétérogène** de celle-ci, qui peut gérer deux types de navigations complètement différents au sein de la même organisation. Nous verrons dans le chapitre 7 jusqu'où peut aller cette hétérogénéité, des résultats concrets y sont présentés.

5.5 Conclusion

L'association de chaque agent de l'organisation multi-agents à un robot du système multi-robots a permis à tous les agents/robots d'avoir la même architecture de contrôle précédemment décrite dans les Chapitres 3 et 4, avec les mêmes spécifications, normes, contextes et fonctionnalités, mais aussi avec des modes de fonctionnements différents. Le système multi-robots avec toutes ses composantes, peut être contrôlé (grâce à l'architecture de contrôle proposée) suivant trois différents types d'exécutions : l'exécution *Centralisée*, l'exécution *Centralisée/Distribuée* et l'exécution *Distribuée*, selon le besoin. Ces différents types de navigations s'effectuent grâce à deux types de traitements *Cognitif* (représenté par les planifications et re-planifications) et *Réactif* (représenté par l'évitement réactif et l'attraction vers une cible).

On a ainsi obtenu une architecture sous forme de framework ouvert. En effet, grâce à l'efficacité et la souplesse du modèle organisationnel multi-agents proposé, l'architecture peut être modifiée et étendue très facilement, par exemple en modifiant les algorithmes de planification, d'évitement d'obstacles ou de génération de formation utilisés, ou encore en rajoutant d'autres modules pour améliorer ou s'adapter à l'exécution des missions en munissant les robots de caméras, bras manipulateurs, etc.

Conçue de cette manière, l'architecture de contrôle/commande proposée répond parfaitement aux exigences que nous nous sommes imposés en section 1.4.

Maintenant que cette structure logicielle est développée, elle peut être utilisée pour déployer des robots dans un environnement physique. Pour ce faire, nous présenterons dans ce qui suit, une description détaillée du simulateur *ROBOTOPIA* développé pour expérimenter l'architecture de contrôle proposée.

Chapitre 6

Implémentation de *ROBOTOPIA* : simulation de la coordination multi-agents et de la commande des robots mobiles

Dans ce chapitre, nous nous intéresserons à l'implémentation de l'architecture de contrôle proposée à travers le développement du logiciel de simulation de la coordination multi-agents *ROBOTOPIA*. Le simulateur est destiné à simuler divers scénarios de coordination d'un système multi-robots dans un environnement encombré. Dans ce chapitre nous décrirons les différentes composantes et modèles d'implémentation de ROBOTOPIA, qui ont permis la mise en œuvre de l'architecture de contrôle/commande proposée.

6.1 Introduction de l'architecture de ROBOTOPIA

La conception *ROBOTOPIA* repose sur le framework de programmation Utopia pour implémenter les quatre spécifications de $\mathcal{M}$OISE^{Inst} : la Spécification Structurelle (SS), la Spécification Fonctionnelle (SF), la Spécification Contextuelle (SC) et la Spécification Normative (SN), avec comme concept de base les "graphes récursifs". En effet, ROBOTOPIA est organisé de la même façon qu'Utopia avec la même architecture (cf. figure 4.14) et les mêmes composantes. La différence réside dans l'adaptation des spécifications incluses dans les différents managers au cas de la robotique mobile collective, comme expliqué dans le chapitre 4 section 4.1.

Pour gérer avec des objets informatiques la Spécification de l'Organisation (SO) telle que nous l'avons décrite, nous utilisons différents "parsers [1]" qui ont la charge de parcourir la structure XML [2] décrivant l'organisation. Ce fichier contient les quatre spécifications de $\mathcal{M}$OISE^{Inst} (cf. Chapitre 4) : SS, SF, SC et SN.

Nous utilisons le graphe récursif présenté dans la figure B.6 pour stocker toutes ces spécifications. Les classes OS, SS, FS, CS et NS héritent donc de *RecursiveGraph* et définissent les objets atomiques qu'elles utilisent : *Role*, *Goal*, *Context* et *Norm*. Finalement le graphe récursif OS comporte quatre sous-graphes composés des spécifications et nous permet d'avoir un stockage global et harmonieux de la spécification de l'organisation. Cet objet nous sera très utilise pour implémenter l'instanciation concrète de l'organisation par des agents comme nous allons le voir dans la partie suivante. La figure 6.1 représente un schéma UML (cf. annexe D) de la spécification de l'organisation où on voit bien les différents liens entre les quatre spécifications de l'organisation ainsi que leurs parsers respectifs.

6.2 La spécification des modèles d'organisation dans ROBOTOPIA : cas d'utilisation

L'implémentation de l'architecture de contrôle proposée dans le simulateur ROBOTOPIA, réside en la transcription de toutes les spécifications de l'organisation du modèle $\mathcal{M}$OISE^{Inst}, dans une structure XML et le codage du contenu de chaque spécification. Dans le chapitre 4 nous avons décrit en détail toutes

1. Le parser est un algorithme qui permet d'analyser un texte et d'en déterminer sa structure syntaxique afin d'effectuer divers traitements, comme par exemple une compilation
2. eXtensible Markup Language

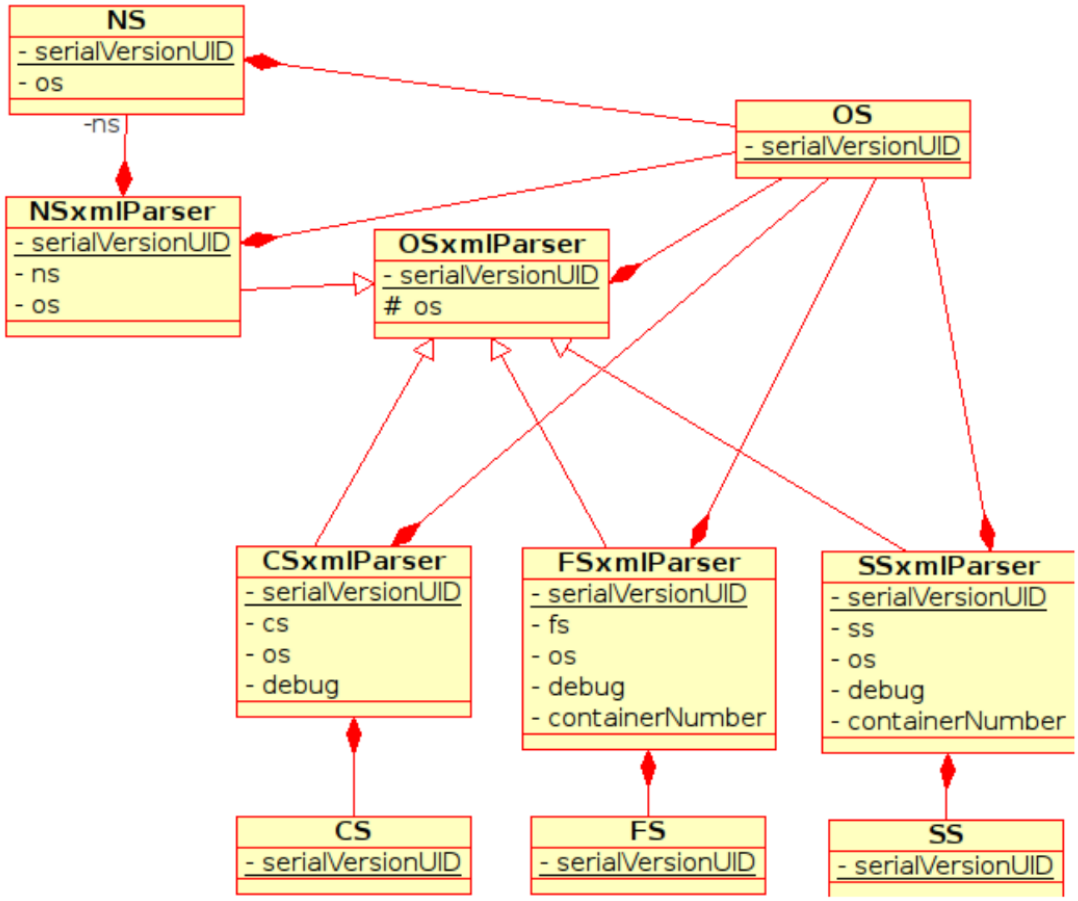


FIGURE 6.1 – UML : Spécification de l'organisation et parser XML.

les spécifications (cf. section 4.1), en décrivant chacun des rôles, goals, missions, contextes, et scènes ainsi que tous les liens qui les relient. Dans cette partie nous nous contenterons de décrire la spécification de l'organisation à travers la structure XML correspondant à une navigation en mode planification et re-planification, de trois robots autonomes évoluant dans le même environnement pour illustrer le fonctionnement de ROBOTOPIA (cet exemple n'est évidemment pas figé, il est modifiable à l'infini selon le comportement de l'architecture qu'on souhaite avoir).

Spécification Structurelle

Nous avons trois robots qui vont être représentés par trois agents. Dans notre spécification structurelle (cf. figure 6.2). Ils auront chacun un rôle $rRobot - i$ (pour un robot d'indice i). Tous les rôles sont les mêmes car ils vont effectuer les

mêmes actions, d'où la cardinalité de 1 pour qu'il y est qu'un seul agent affecté à chaque rôle. On a ajouté un autre agent de supervision avec un rôle *rSupervisor* dont la cardinalité est aussi égale à 1.

```
<StructuralSpecification>
        <Group id="Robotopia" min="4" max="4">
                <Role id="rRobot-1" min="1" max="1"/>
                <Role id="rRobot-2" min="1" max="1"/>
                <Role id="rRobot-3" min="1" max="1"/>
                <Role id="rSupervisor" min="1" max="1"/>
                <Link source="rRobot-1" destination="rSupervisor"
                 type="communication" scope="intra-group"
                 extendsToSubGroups="true" symmetric="true"/>
                <Link source="rRobot-2" destination="rSupervisor"
                 type="communication" scope="intra-group"
                 extendsToSubGroups="true" symmetric="true"/>
                <Link source="rRobot-3" destination="rSupervisor"
                 type="communication" scope="intra-group"
                 extendsToSubGroups="true" symmetric="true"/>
        </Group>
</StructuralSpecification>
```

FIGURE 6.2 – Représentation du code XML de la Spécification Structurelle.

Dans la figure 6.2 on a organisé notre code XML en définissant en premier lieu les différents rôles des différents agents, puis on a fini par définir les liens entre eux.

Spécification Fonctionnelle

Cette spécification (cf. figure 6.3) est représentée par quatre missions contenant divers goals : une mission *mPlan* pour la planification de trajectoire et le suivi de celle-ci, *mCalc* pour la re-planification de trajectoire dans le cas où on rencontre un obstacle inattendu, *mReact* pour éviter réactivement l'obstacle et demander une nouvelle planification une fois terminée, et la mission *mSupervisor* pour permettre la résolution de conflits entre les agents/robots.

```
<FunctionalSpecification>
        <Mission id="mPlan" headGoal="gRoot">
                <GoalId>gRoot</GoalId>
                <Plan headGoal="gRoot" operator="sequence">
                        <GoalId>gReachTarget</GoalId>
                        <GoalId>gAskForPermission</GoalId>
                        <GoalId>gComputeTrajectory</GoalId>
                        <GoalId>gWaitSupervisor</GoalId>
                </Plan>
        </Mission>

        <Mission id="mCalc" headGoal="gRoot">
                <GoalId>gRoot</GoalId>
                <Plan headGoal="gRoot" operator="sequence">
                        <GoalId>gCalculateTrajectory</GoalId>
                        <GoalId>gForward</GoalId>
                </Plan>
        </Mission>

        <Mission id="mSupervisor" headGoal="gRoot">
                <GoalId>gRoot</GoalId>
                <GoalId>gSupervisor</GoalId>
                <GoalId>gSupervisorGUI</GoalId>
                <GoalId>gSupervisorCollisionSolver</GoalId>
                <GoalId>gRadarSimulation</GoalId>
                <Plan headGoal="gRoot" operator="parallel">
                        <GoalId>gSupervisor</GoalId>
                        <GoalId>gSupervisorGUI</GoalId>
                        <GoalId>gSupervisorCollisionSolver</GoalId>
                        <GoalId>gRadarSimulation</GoalId>
                </Plan>
        </Mission>

        <Mission id="mReact" headGoal="gRoot">
                <GoalId>gRoot</GoalId>
                <Plan headGoal="gRoot" operator="sequence">
                        <GoalId>gReturnOnTrajectory</GoalId>
                        <GoalId>gAvoidObstacleNP</GoalId>
                </Plan>
        </Mission>

</FunctionalSpecification>
```

FIGURE 6.3 – Représentation du code XML de la Spécification Fonctionnelle.

Le code XML présenté dans la figure 6.3, commence par définir une mission à la fois, elle contient les différents goals qui la constituent ainsi que la façon avec

laquelle ces goals seront exécutés (séquentiellement ou en parallèle).

Spécification Contextuelle

La spécification contextuelle donne à chaque agent/robot la même scène et à l'agent superviseur une scène différente, soit quatre scènes représentant les contextes et les transitions existantes dans la spécification contextuelle.

```
<ContextualSpecification>
        <Scene id="sSupervisor" initialCtxt="life" finalCtxt="dead">
                <ContextDesc id="dead"/>
                <ContextDesc id="life"/>
                <Transition id="t1" source="life" target="dead" eventId="finish"/>
        </Scene>

        <Scene id="sRobot1" initialCtxt="planningMode" finalCtxt="dead">
                <ContextDesc id="dead"/>
                <ContextDesc id="planningMode"/>
                <ContextDesc id="calculMode"/>
                <Transition id="t1" source="planningMode" target="dead" eventId="finish"/>
                <Transition id="t2" source="planningMode" target="calculMode" eventId="aO1"/>
                <Transition id="t3" source="calculMode" target="dead" eventId="finish"/>
                <Transition id="t4" source="calculMode" target="calculMode" eventId="aO1"/>
                <Transition id="t5" source="calculMode" target="planningMode" eventId="AA1"/>
        </Scene>

        <Scene id="sRobot2" initialCtxt="planningMode" finalCtxt="dead">
                <ContextDesc id="dead"/>
                <ContextDesc id="planningMode"/>
                <ContextDesc id="calculMode"/>
                <Transition id="t1" source="planningMode" target="dead" eventId="finish"/>
                <Transition id="t2" source="planningMode" target="calculMode" eventId="aO2"/>
                <Transition id="t3" source="calculMode" target="dead" eventId="finish"/>
                <Transition id="t4" source="calculMode" target="calculMode" eventId="aO2"/>
                <Transition id="t5" source="calculMode" target="planningMode" eventId="AA2"/>
        </Scene>

        <Scene id="sRobot3" initialCtxt="planningMode" finalCtxt="dead">
                <ContextDesc id="dead"/>
                <ContextDesc id="planningMode"/>
                <ContextDesc id="calculMode"/>
                <Transition id="t1" source="planningMode" target="dead" eventId="finish"/>
                <Transition id="t2" source="planningMode" target="calculMode" eventId="aO3"/>
                <Transition id="t3" source="calculMode" target="dead" eventId="finish"/>
                <Transition id="t4" source="calculMode" target="calculMode" eventId="aO3"/>
                <Transition id="t5" source="calculMode" target="planningMode" eventId="AA3"/>
        </Scene>
</ContextualSpecification>
```

FIGURE 6.4 – Représentation du code XML de la Spécification Contextuelle.

Dans la figure 6.4 on définit des scènes pour chaque agent/robot contenant les différents modes de fonctionnement ainsi que les différentes transitions qui les régissent.

Spécification Normative

Dans cette spécification, on assigne à chaque agent de l'organisation des normes pour le forcer à exécuter le comportement voulu, soit trois normes pour chaque agent/robot pour exécuter les missions $mPlan$, $mReact$ et $mCalc$, et une norme pour l'agent superviseur pour exécuter la supervision (on a en tout dix normes, trois pour chaque agent robot et une norme pour le superviseur, présentés dans la figure 6.5). La spécification des modèles d'organisation Le fichier XML ainsi constitué fourni a ROBOTOPIA toutes les informations utiles pour l'organisation et la coordination du système multi-agents. Les agents seront alors créés selon le nombre de robots fourni, ils exécuteront les différents goals et missions qui leurs sont propres pour arriver à mener à bien la mission et le comportement souhaité.

```
<NormativeSpecification>

        <Norm id="N1" weight="1" operator="O" bearer="rRobot-1" issuer="rRobot-1"
        context="/OS/CS/sRobot1/planningMode" actionT="mission" action="mPlan"/>

        <Norm id="N2" weight="1" operator="O" bearer="rRobot-2" issuer="rRobot-2"
        context="/OS/CS/sRobot2/planningMode" actionT="mission" action="mPlan"/>

        <Norm id="N3" weight="1" operator="O" bearer="rRobot-3" issuer="rRobot-3"
        context="/OS/CS/sRobot3/planningMode" actionT="mission" action="mPlan"/>

        <Norm id="N4" weight="1" operator="O" bearer="rSupervisor" issuer="rSupervisor"
        context="/OS/CS/sSupervisor/life" actionT="mission" action="mSupervisor"/>

        <Norm id="N5" weight="1" operator="O" bearer="rRobot-1" issuer="rRobot-1"
        context="/OS/CS/sRobot1/reactiveMode" actionT="mission" action="mReact"/>

        <Norm id="N6" weight="1" operator="O" bearer="rRobot-2" issuer="rRobot-2"
        context="/OS/CS/sRobot2/reactiveMode" actionT="mission" action="mReact"/>

        <Norm id="N7" weight="1" operator="O" bearer="rRobot-3" issuer="rRobot-3"
        context="/OS/CS/sRobot3/reactiveMode" actionT="mission" action="mReact"/>

        <Norm id="N8" weight="1" operator="O" bearer="rRobot-1" issuer="rRobot-1"
        context="/OS/CS/sRobot1/calculMode" actionT="mission" action="mCalc"/>

        <Norm id="N9" weight="1" operator="O" bearer="rRobot-2" issuer="rRobot-2"
        context="/OS/CS/sRobot2/calculMode" actionT="mission" action="mCalc"/>

        <Norm id="N10" weight="1" operator="O" bearer="rRobot-3" issuer="rRobot-3"
        context="/OS/CS/sRobot3/calculMode" actionT="mission" action="mCalc"/>

</NormativeSpecification>
```

FIGURE 6.5 – Représentation du code XML de la Spécification Normative.

6.3 *ROBOTOPIA : implémentation et instanciation des modèles Utopia*

Dans cette section, nous allons présenter plus en détails l'implémentation des différentes constituantes de ROBOTOPIA, à savoir les différents managers, le superviseur, et les agents.

Dans ROBOTOPIA, les différents agents du système héritent d'une classe de base *GenericAgent* qui définit un ensemble de primitives essentielles et utilisées par tous (cf. figure 6.6). *GenericAgent* utilise principalement deux autres classes :

— $Mutex$, qui met en œuvre un sémaphore [3] d'exclusion mutuelle, pour gérer les soucis potentiels de synchronisation au sein d'un même agent.
— $CommandValidator$ est un objet qui sert à définir le protocole de communication de l'agent, à savoir, l'ensemble des commandes qu'il peut recevoir etc. Tous les messages systèmes de $ROBOTOPIA$ sont des messages ACL (de JADE cf. annexe D) dans lesquels nous avons encapsulé un objet $Command$ construit à partir de texte comme une commande $POSIX$ (cf. annexe B). Nous voyons donc un agent comme une sorte de « terminal » dans lequel un autre agent peut exécuter des commandes et en récupérer les résultats.

Maintenant que nous avons vu comment les agents s'organisaient de manière globale, nous allons présenter en détail les différents agents intervenant dans $ROBOTOPIA$ et la manière dont ils interagissent.

6.3.1 ContextManager

Comme on a pu le voir dans la description d'Utopia (section 4.2), le *ContextManager* de $ROBOTOPIA$ a pour objectif de gérer l'ensemble des contextes (états) dans lesquels le système peut se trouver, conformément à la spécification contextuelle (cf. figure 6.7). On peut donc le voir comme une machine parallèle à états finis exécutant un thread pour chaque scène qu'il doit suivre.

Une scène étant composée d'un ensemble de contextes, de transitions et éventuellement d'autres sous-scènes (les différents contextes dans lesquels notre système multi-agents/robot peut rencontrer). Une scène définit en quelque sorte un diagramme d'état-transition récursif. Le thread chargé d'une scène commence par le sommet initial et avance sur l'état suivant si un message définissant une transition adéquate est reçue.

3. Un sémaphore est une variable, ou un type de donnée abstrait, qui constitue la méthode utilisée couramment pour restreindre l'accès à des ressources partagées.

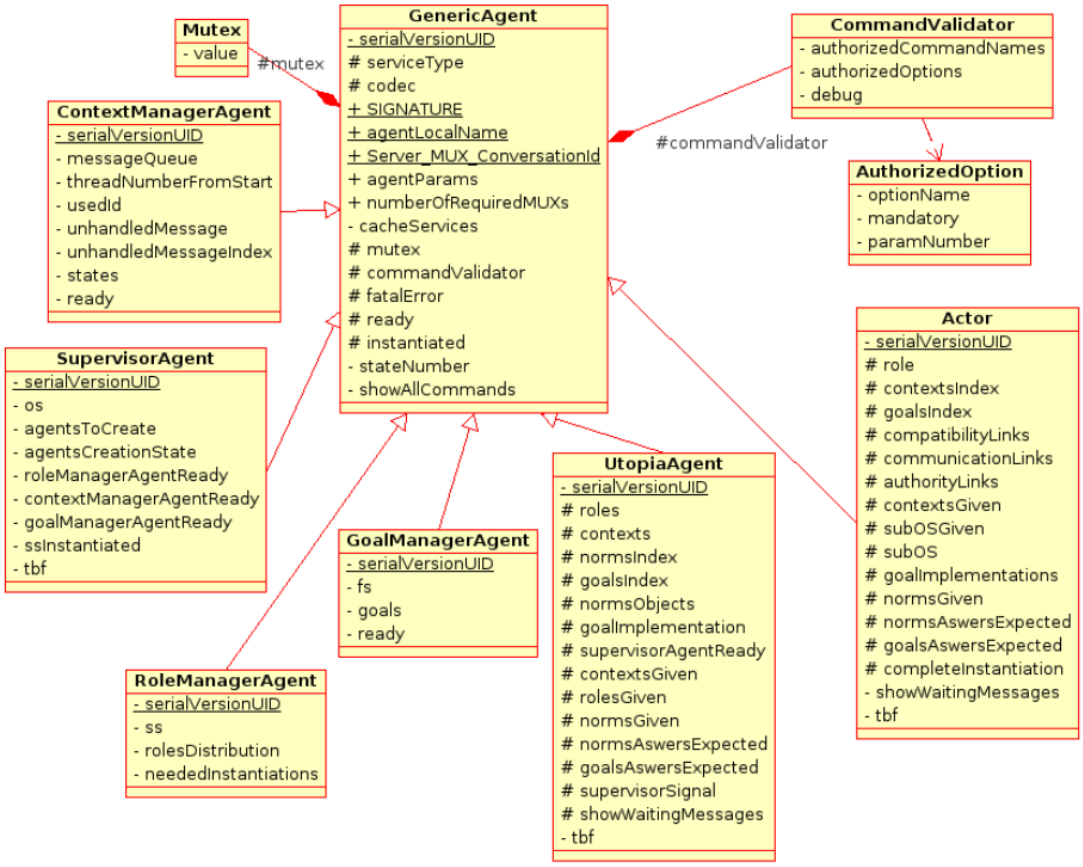


FIGURE 6.6 – Organisation générale des agents.

Les commandes gérées par le *ContextManager* sont principalement les transitions et les requêtes pour obtenir l'ensemble des états courants (initialisation). En outre, il informe l'ensemble des acteurs de tout changement de contexte. Finalement, quand tous les threads ont fini leur exécution (alors l'état final de la SC est atteint), il informe l'ensemble des agents que l'instanciation de la spécification est terminée, il est donc l'acteur qui permet l'arrêt du simulateur *ROBOTOPIA*.

Cet agent met en œuvre 3 comportements :

— *CMAMessageReceiverBehaviour* : dont l'exécution est cyclique, il reçoit et renseigne la *SC* (spécification contextuelle), puis traite tous les messages et commandes.
— *ContextManagerBehaviour* : gère les scènes. Initialement, le *ContextManager* crée ce comportement avec la *SC* comme paramètre puis se positionne sur l'état initial. Si plusieurs voisins existent alors la scène

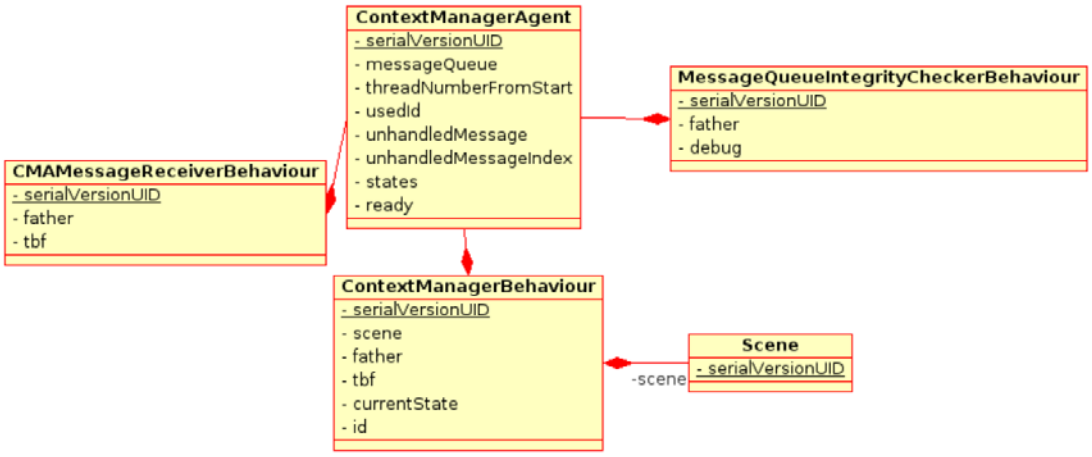


FIGURE 6.7 – Diagramme de classe du *ContextManager*.

fait des liens vers des sous-scènes. Dans ce cas le *Behaviour* ajoute au *ContextManager* autant d'autres *ContextManagerBehaviour* que nécessaire avec ces sous-scènes en paramètre. Cela nous permet de parcourir récursivement la *SC* via l'ajout successif de comportements. Dans notre cas on peut assimiler le fonctionnement du *ContextManagerBehaviour* au passage dans une scène d'une mission à l'autre, comme quand l'agent robot commence sa mission de planification dans la scène *sPlanificationReplanification*, il échange avec le superviseur en envoyant la trajectoire et recevant l'autorisation, il passe au suivi, etc.

— *MessageQueueIntegrityCheckerBehaviour* : sert à détecter les transitions qui sont totalement intégrés, pendant la surcharge de la file de JADE qui garantit que toute transition atteindra son destinataire.

6.3.2 RoleManager

Le *RoleManager* de *ROBOTOPIA* est chargé de la gestion des rôles (cf. figure 6.8) ce qui implique :

— déterminer la distribution des rôles (quels rôles seront joués par quels agents, dans notre cas soit le rôle d'un robot soit le rôle d'un superviseur) et retourner le nombre d'agents nécessaires à l'instanciation de la spécification structurelle.
— fournir l'ensemble des rôles qu'un agent doit jouer s'il en fait la demande, ou fournir à nouveau la même liste si la demande a déjà été effectuée.
— indiquer si tous les agents nécessaires à l'instanciation jouent un rôle.

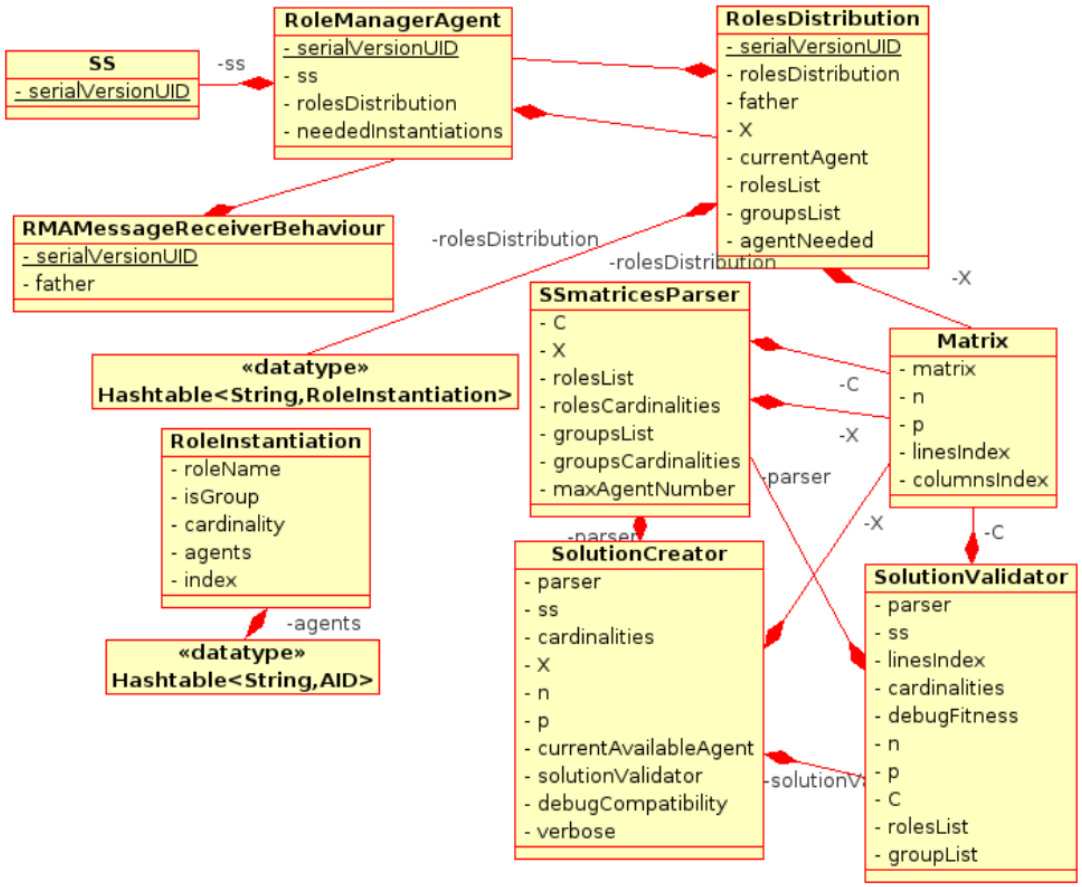


FIGURE 6.8 – Diagramme de classe du *RoleManager*.

De manière analogue au *ContextManager*, un comportement est dédié à la réception de la spécification structurelle (SS) et des messages, c'est le *RMAMessageReceiverBehaviour*. Dans la figure 6.8 c'est le seul comportement, il est en charge de gérer toutes les commandes et il s'appuie à cette fin sur plusieurs classes.

SolutionCreator (cf. figure 6.8) fournit une matrice binaire associant tous les rôles aux agents. Cette matrice est exploitée pour faire de la résolution inverse (associer un agent à un rôle) : quand un agent demande le rôle qu'il doit jouer, on regarde la colonne correspondant à l'agent courant i et on retourne tous les rôles (associés aux lignes) ou l'intersection comporte un 1.

Avec l'exemple du tableau 6.9, on indiquerait à l'agent 2 qu'il doit jouer les rôles « Rôle 2 » et « Rôle R ».

	Agent 1	Agent 2	...	Agent i	...	Agent A
Rôle 1	1	0	...	0	...	0
Rôle 2	0	1	...	1	...	0
...	...	...	...	...	...	...
Rôle R	0	1	...	1	...	0

FIGURE 6.9 – Exemple de matrice renvoyée par *SolutionCreator*.

RoleDistribution (cf. figure 6.8) mémorise l'ensemble des agents (avec leurs noms et leurs AID [4]) et les associent aux rôles qu'ils jouent. Si un agent demande plusieurs fois sa liste de rôle (en cas d'erreur de transmission par exemple) on sera alors capable de lui retourner la liste des rôles qui lui est prévu de jouer.

6.3.3 GoalManager

Au *GoalManager* est confié la gestion de la Spécification Fonctionnelle. La figure 6.10 représente une illustration du diagramme de classe de ce manager, il doit principalement réaliser un mapping (association) entre l'ensemble des buts (Goal) de la spécification et des instances concrètes des classes mettant en œuvre les actions associés à ces buts (dans notre cas il représente l'organisation d'une mission comme la planification, re-planification, évitement d'obstacles, etc.).

— *GMAMessageReceiverBehaviour* (cf. figure 6.10) reçoit et renseigne la *SF* puis traite les messages et les commandes.
— *GoalManagerAgent* (cf. figure 6.10) utilise l'objet *GoalsInstances* gérant le mapping <*Goal*, *GoalImplementation*> avec une table de hachage.
— *GoalImplementation* (cf. figure 6.10) est une classe abstraite qui permet, en l'implémentant, de spécifier les actions à exécuter pour atteindre un but en particulier.

6.3.4 Superviseur

Au sommet de la hiérarchie du système se trouve le superviseur (cf. figure 6.11), ses buts sont dans l'ordre :

— créer un graphe récursif de la spécification de l'organisation (*SO*) à partir d'une structure *XML* passée en paramètre.

4. AID est une classe d'identification d'agents dans JADE, elle est utilisée pour enregistrer les noms et les adresses des agents

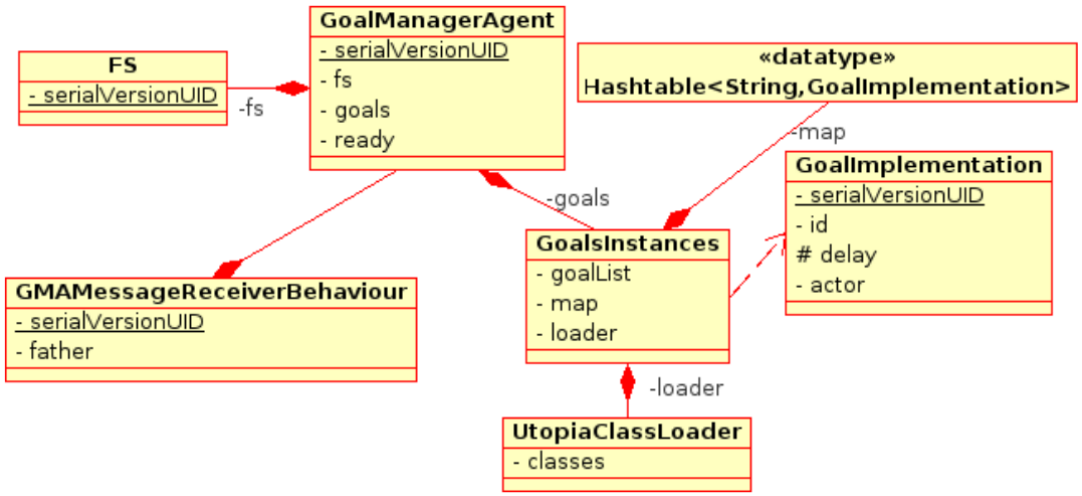


FIGURE 6.10 – Diagramme de classe du *GoalManager*.

— créer les 3 managers que nous venons de voir : *ContextManager*, *RoleManager* et *GoalManager*.
— envoyer les sous graphes de la *SO* aux managers correspondants à savoir la *SC* au *ContextManager*, la *SS* au *RoleManager* et enfin la *SF* au *GoalManager*.
— prendre en charge la spécification normative (*SN*) puisqu'en faisant des liens vers toutes les autres spécifications, la *SO* en entier est nécessaire pour la gérer.

SAMMessageReceiverBehaviour reçoit les messages et les commandes, *l'Agents LauncherBehaviour* est responsable de la création des managers, dont le résultat est donné grâce à *AgentCreationCheckerBehaviour*. Une fois tous les agents créés le *SAInitializationSequenceBehaviour* est lancé puis exécute en séquence les comportements correspondant aux dernières étapes de l'initialisation du superviseur, à savoir *WaitForAgentsInitializationBehaviour* (attend l'initialisation des 3 managers) puis *WaitForSSInstantiationBehaviour* (attend que le nombre adéquat d'agents/robots soit lancé).

6.3.5 ROBOTOPIA Agent

Un premier pas vers un comportement intelligent de la part des agents et de les concevoir de manière générique tout en faisant en sorte qu'ils se spécialisent de manière autonome pour que plusieurs agents, codés de la même manière, se

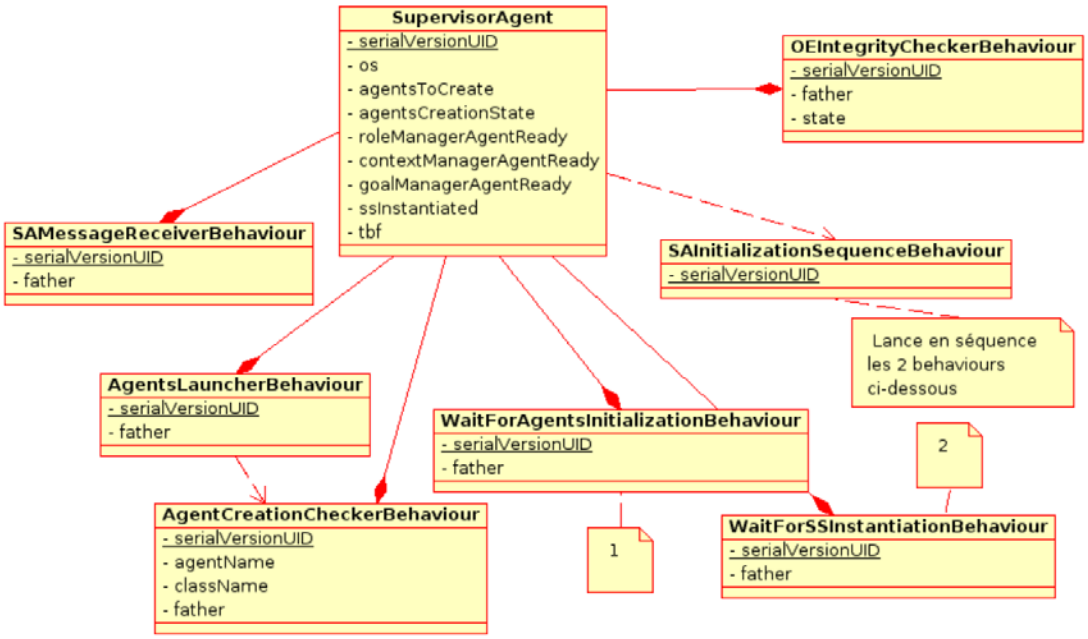


FIGURE 6.11 – Diagramme de classe du *Superviseur*.

comportent différemment. À cette fin, l'agent ROBOTOPIA est composé de deux entités : Agents principales et sous-agents appelés Acteurs.

Agent principal

Cet agent va s'approprier lui-même un rôle et agir en fonction des contextes de l'institution et des normes qui s'appliquent à lui (cf. figure 6.12). L'idée est donc d'obtenir une auto-organisation des agents par rapport aux spécifications.

MAInitializationSequenceBehaviour (cf. figure 6.12) exécute en séquence correspondantes à la phase d'initialisation d'un Agent ROBOTOPIA à savoir les comportements :

— *WaitForSupervisorBehaviour* (cf. figure 6.12) qui attend que le superviseur soit initialisé ce qui implique que tous les managers le soient.
— *GetRolesBehaviour* (cf. figure 6.12), récupère le ou les rôles que cet agent doit jouer.
— *CreateSubAgentsBehaviour* (cf. figure 6.12) va créer autant de sous-agents « Actor » qu'il a de rôle à jouer.

Acteurs

L'acteur, comme son nom l'indique, a pour fonction principale de « jouer » un rôle. A cette fin, il devra effectuer un ensemble d'actions en accord avec les spécifications. Pour déterminer ce qu'il doit, peut ou ne peut pas faire, il utilise

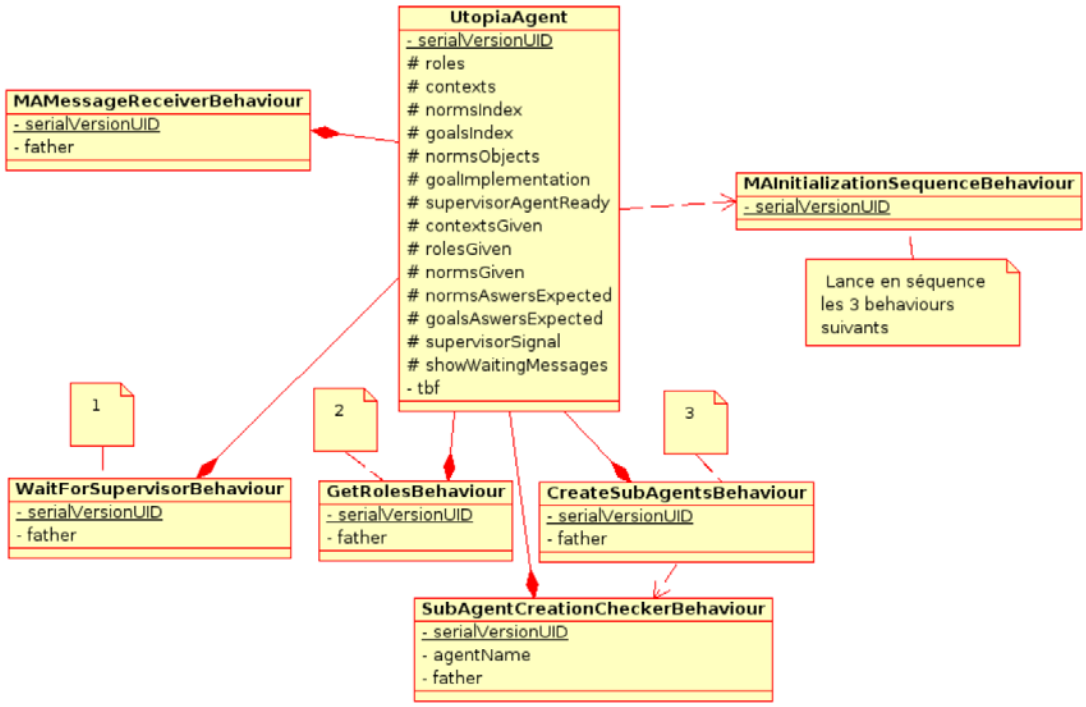


FIGURE 6.12 – Diagramme de classe d'un *ROBOTOPIAAgent*.

les contextes de l'organisation afin de retrouver l'ensemble des normes portant sur lui, la figure 6.13 présente le diagramme de classe d'un acteur.

Finalement, après un choix et l'exécution des actions consécutives à cette décision, il convient que l'acteur soit à l'écoute de tout changement de contexte éventuel pour effectuer, le cas échéant, un nouveau cycle de décision. Et cela tant que le *ContextManager* ne l'arrête pas après avoir atteint l'état terminal de la Spécification Contextuelle.

D'une manière similaire à tous les agents, un comportement est dédié à la réception des messages. Avec *GetContextsBehaviour* (cf. figure 6.13) on récupère les contextes initiaux de l'institution, puis on déclenche un cycle de vie : $LifeBehaviour$. Les actions suivantes vont alors être réalisées séquentiellement (cf. figure 6.13) :

— *GetSubOsBehaviour* : récupère à partir de la *SO* le sous-graphe récursif induit par le rôle de l'acteur et les contextes. L'acteur peut alors déterminer l'ensemble des missions sur lesquelles il peut s'engager (via la sous spécification normative) la manière d'exécuter une telle mission (via la sous spécification fonctionnelle) mais aussi ses relations avec les autres acteurs

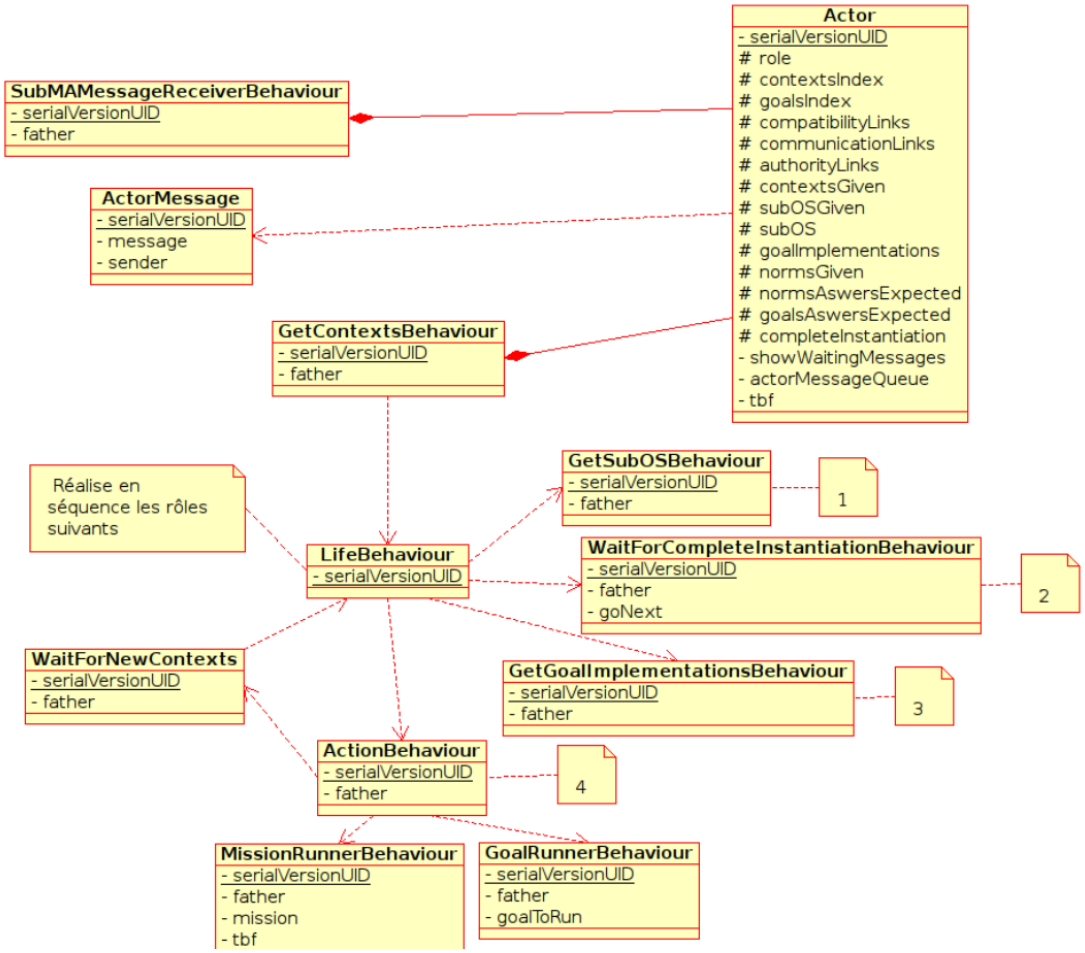


FIGURE 6.13 – Diagramme de classe d'un *Acteur*.

(via la sous spécification structurelle).

— *WaitForCompleteInstantationBehaviour* : permet d'attendre que les agents nécessaires pour instancier l'institution aient pris possession d'un rôle, avant de commencer à effectuer des actions.

— *GetGoalImplementationBehaviour* : récupère toutes les implémentations de « Goal » que l'agent pourrait réaliser.

— *ActionBehaviour* : permet de s'engager sur les missions / buts à réaliser.

Les actions sont exécutées via *MissionRunnerBehaviour* composé de *GoalRunnerBehaviour* (exécutants les méthodes Run() des implémentations de goal associées) et éventuellement d'autres *MissionRunnerBehaviour* en fonction du caractère récursif de la mission. Le comportement composé l'est de telle manière à respecter la spécification fonctionnelle, en d'autres termes, nous veillons à bien

effectuer les actions en séquence, en parallèle, ou à faire des choix comme indiqué dans la spécification fonctionnelle (cf. section 4.1).

En parallèle, $WaitForNewContext$ est appliqué à l'acteur afin de détecter d'éventuels changements de contexte. Si tel est le cas, un nouveau $LifeBehaviour$ sera ajouté et l'acteur réalisera un autre cycle décisionnel.

6.4 Conclusion

La structure de données que nous avons présentée : les graphes récursifs (à savoir des graphes pouvant être des graphes de graphes) ; différents types d'arêtes ont permis de faciliter la gestion orientée objet de la spécification de l'organisation.

Malgré la difficulté de l'implémentation liée à la récursivité omniprésente de la structure de donnée et de la gestion mémoire, cette structure nous a aidée par la suite à faire abstraction de ce caractère récursif pour nous concentrer sur les détails des quatre spécifications.

Le fait de pouvoir gérer toute la spécification de l'organisation avec un seul objet décomposable facilement (notion de sous-graphe) fut un point clé dans la collaboration des agents, notamment en simplifiant le transport et donc le partage des informations.

ROBOTOPIA (comme Utopia) a été programmé comme un framework dont l'objectif est d'interpréter une spécification d'organisation, de déterminer la manière la plus intelligente d'engager des agents sur les rôles de l'institution, d'automatiser leur déploiement comme leurs actions. Chaque agent se spécialise en fonction du rôle qu'il joue et des contextes de l'organisation puis exécute à la volée les actions appropriées, telles que prévus dans le scénario de navigation.

Ces utilisations dans la version actuelle sont déjà vastes : parallélisation d'algorithmes classiques en décomposant les fonctions grâce à la spécification fonctionnelle, déploiement simple d'un système multi-agents complexe, calcul distribué, etc.

On a pu ainsi simuler plusieurs scénarios de fonctionnement d'un système multi-robots d'une manière souple et efficace. Dans le chapitre suivant nous présenterons les différents simulations effectuées et leurs résultats respectifs.

Chapitre 7

Évaluations multi-configurations de l'architecture ACO^2DA

Après avoir présenté le logiciel de simulation ROBOTOPIA ainsi que l'implémentation de l'architecture de contrôle proposée ACO^2DA. Nous présentons dans ce chapitre plusieurs simulations de cette architecture en utilisant ROBOTOPIA (cf. Chapitre 6), dans différents contextes de navigation du système multi-robots.

Dans le chapitre 5, nous avons présenté les différents modes de fonctionnement de l'architecture de contrôle proposée, soit : *Navigation Centralisée*, *Navigation Distribuée* et *Navigation Centralisée/Distribuée*. Dans le cadre de la simulation de cette architecture de contrôle, nous nous proposons de simuler différents scénarios de chacun de ces modes de fonctionnement.

Nous présenterons alors pour chaque type de navigation, les scénarios de simulations associés et leurs résultats. Dans toutes ces simulations, les conditions initiales de l'environnement (robots, obstacles) sont identiques. Afin de mettre en exergue l'influence du mode de fonctionnement choisi pour l'architecture flexible proposée sur la navigation des robots.

7.1 Navigation Centralisée

Comme nous l'avons déjà expliqué (cf. chapitre 5) le principe de fonctionnement de ce type de navigation repose sur l'activation de deux normes du modèle organisationnel utilisé, soit la norme $NSupervision$ et $NPlanningMode$. Les robots sont donc en mode planification supervisée par l'agent superviseur. Ce superviseur est créé en même temps que les agents/robots il est dissocié d'eux, et se situe à l'extérieur des robots tout en exerçant un lien d'autorité sur eux (comme exemple de ce genre de systèmes, on peut citer les SMR supervisés dans les entrepôts, où un superviseur humain ou autre gère leur navigation). La figure 7.1 représente une illustration simplifiée du fonctionnement de ce genre de navigation.

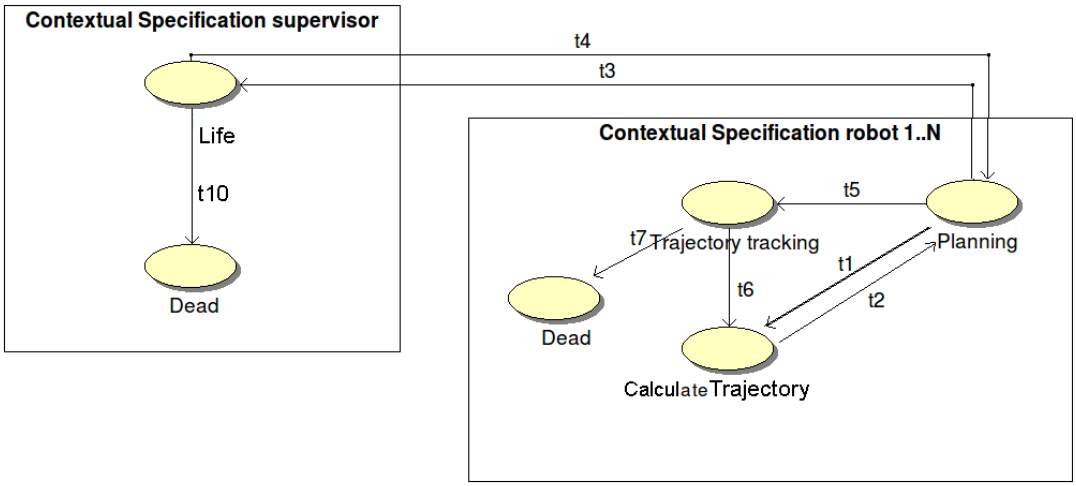


FIGURE 7.1 – Représentation simplifiée du fonctionnement de la navigation Centralisée.

Dans cette figure, on y voit que chaque robot du SMR commence par une planification de sa propre trajectoire (transitions t_1, t_2), puis la transmet au superviseur (transition t_3) qui l'analyse en modifiant la vitesse du robot si nécessaire, puis l'autorise à la suivre ou pas (en modifiant sa vitesse) par la transition t_4, le robot procède donc au suivi de cette trajectoire grâce à la transition t_5. Pendant ce suivi de trajectoire par le robot, il peut y avoir un obstacle inattendu ou non pris en compte initialement pendant la première planification (c'est le cas des environnements incertains ou dynamiques). Ainsi, dès que le robot le détecte, il procède à la transition t_6 pour re-planifier une nouvelle trajectoire prenant en considération : le nouvel obstacle en plus de ceux connus initialement par le

robot ainsi que la position actuelle du robot. Ces informations sont récupérés dans *CalculateTrajectory* et envoyés à *Planning* grâce à t_2, puis la transmet à nouveau au superviseur et ainsi de suite.

Un schéma séquentiel illustrant, étape par étape, le fonctionnement des robots se trouvant en mode navigation centralisée est présenté sur la figure 7.2.

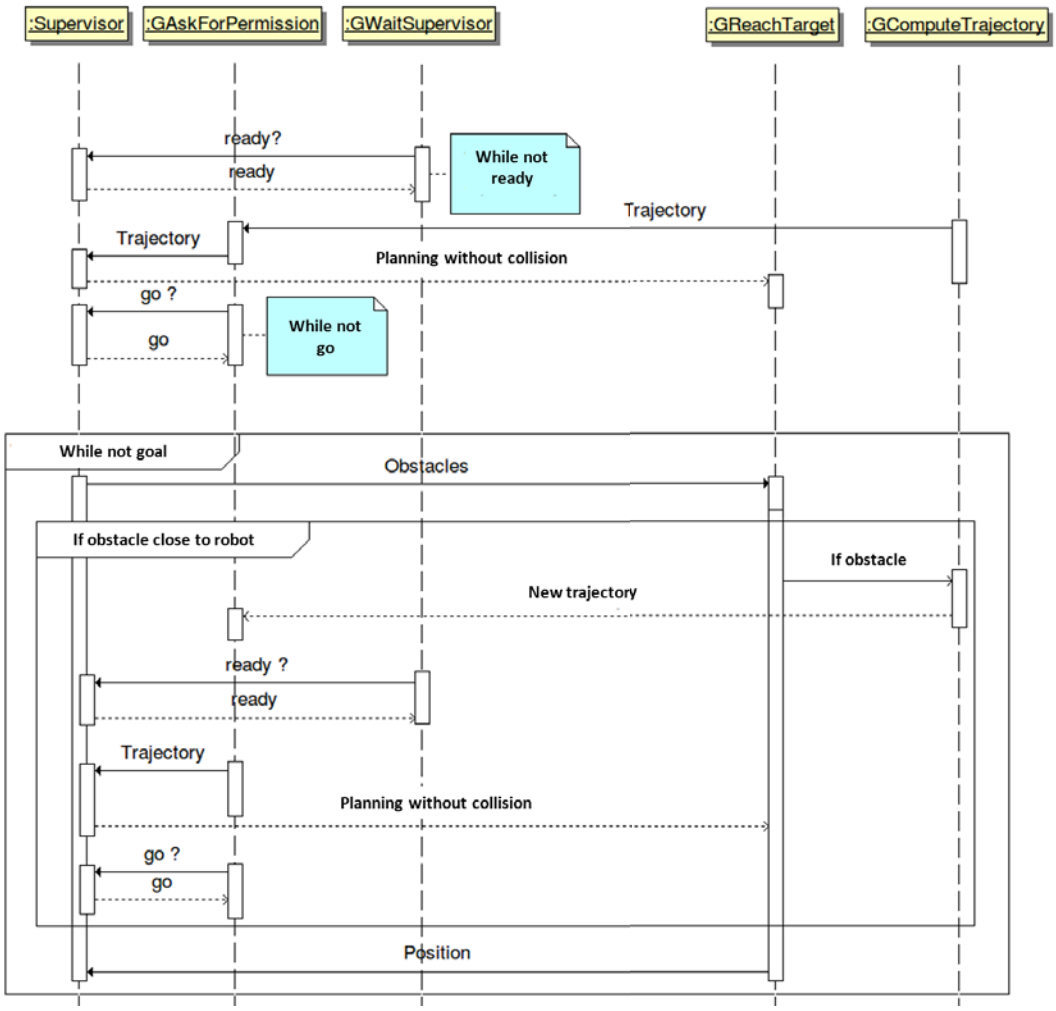


FIGURE 7.2 – Diagramme de séquence de la navigation Centralisée.

Nous avons utilisé deux scénarios pour illustrer le fonctionnement de la navigation centralisée. Ces scénarios consistent à simuler dans ROBOTOPIA la navigation d'un système multi-robots constitué par trois robots mobiles, dans un environnement en présence d'obstacles statiques et dynamiques, pour atteindre trois cibles différentes :

— navigation sans événements inattendus : trois robots mobiles navigant en coordination dans un environnement contenant des obstacles **statiques**, pour atteindre trois destinations différentes.
— navigation avec événements inattendus : trois robots mobiles navigant en coordination dans un environnement contenant des obstacles **statiques** et **dynamiques**, pour atteindre trois destinations différentes.

Dans ce qui suit nous allons présenter les résultats de simulation de ces deux scénarios.

7.1.1 Navigation sans événements inattendus

Grâce à ROBOTOPIA nous avons pu simuler ce scénario, soit la simulation de trois robots mobiles collaborant pour atteindre leurs destinations finales respectives. L'environnement ici contient que des obstacles statiques. Pour différencier nos trois robots nous leurs avons attribué trois couleurs, et trois vitesses de navigation v_{max} :

— robot rouge avec une vitesse $v_r = 20m/s$.
— robot vert avec une vitesse $v_v = 10m/s$.
— robot bleu avec une vitesse $v_b = 10m/s$.

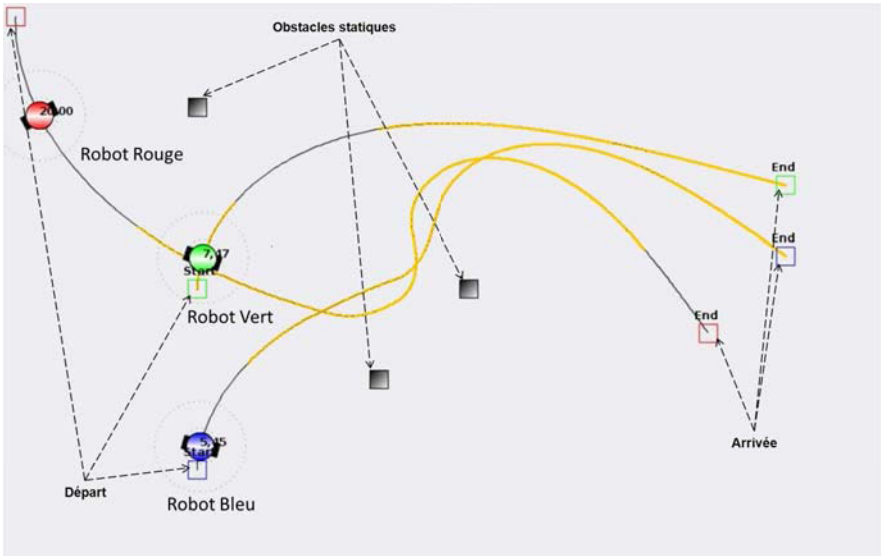


FIGURE 7.3 – Chemins obtenus après planification et modification des vitesses.

Comme on l'a précédemment expliqué, dès que les agents/robots sont créés et initialisés ils commencent par planifier leurs chemins puis entrent en contact

avec le superviseur qui à son tour les autorise soit à suivre leurs trajectoires avec leur vitesse max, soit leur change la vitesse pour éviter les conflits entre eux (cf. algorithme 2). Dans la figure 7.3 on voit que le superviseur autorise le robot rouge à suivre sa trajectoire avec sa vitesse max ($v_r = 20m/s$), mais change les vitesses v_{max} des robots bleu et vert respectivement par $v_b = 5.45m/s$ et $v_v = 7.47m/s$. Ce changement et dû à l'ordre initial de l'envoi des requêtes par les agents/robots de type FIFO (First In First Out), les zones en jaune (ligne épaisse) présentes dans les trajectoires représentent les zones de conflits potentiels détectés par le superviseur, elles correspondent à toutes les positions temporelles des robots où il y a intersections de leurs zones de sécurité (rayon de $70cm$ autour des robots).

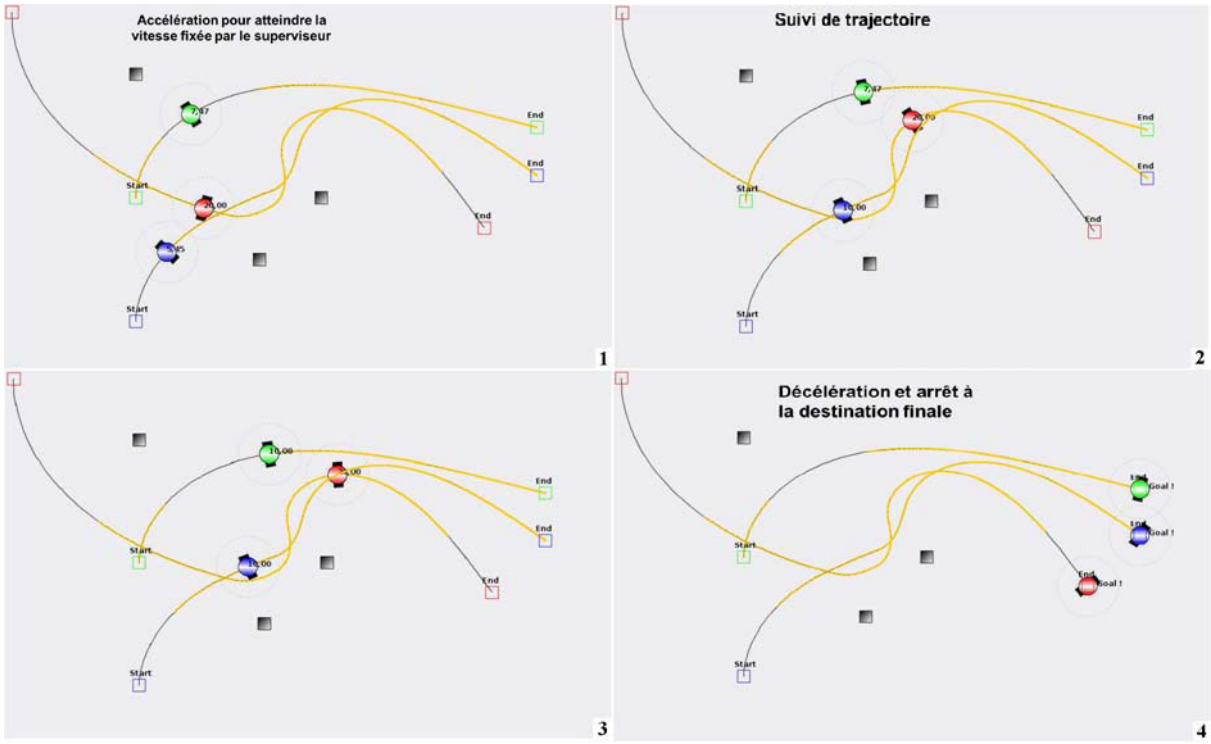


FIGURE 7.4 – Simulation du scénario Navigation sans événements inattendus.

Les figures 7.4 (1) et (2) représentent le début de la navigation des robots (accélération du départ) suivant leurs chemins respectifs. Dans la figure 7.4 (3) on remarque que les vitesses d'évolution des robots vert et bleu ont changés ($v_b = v_v = 10m/s$). En effet, le superviseur a restreint ces deux vitesses car il avait détecté un conflit entre ces deux robots (vert et bleu) et le robot rouge, et comme on le remarque dans cette même figure le robot rouge a dépassé le vert et le bleu et il a comme vitesse $v_r = 20m/s$, ce qui signifie que le superviseur a

résolu ce conflit. Par conséquent, les deux robots vert et bleu peuvent continuer leurs chemins avec leurs vitesses assignées et atteindre ainsi leurs destinations sans conflits (cf. figure 7.4 (4)).

7.1.2 Navigation avec événements inattendus

Dans ce deuxième scénario nous allons réaliser la même simulation que celle présentée dans la section précédente, sauf que cette fois nous allons rajouter des obstacles inattendus en plus des obstacles statiques présents initialement dans l'environnement.

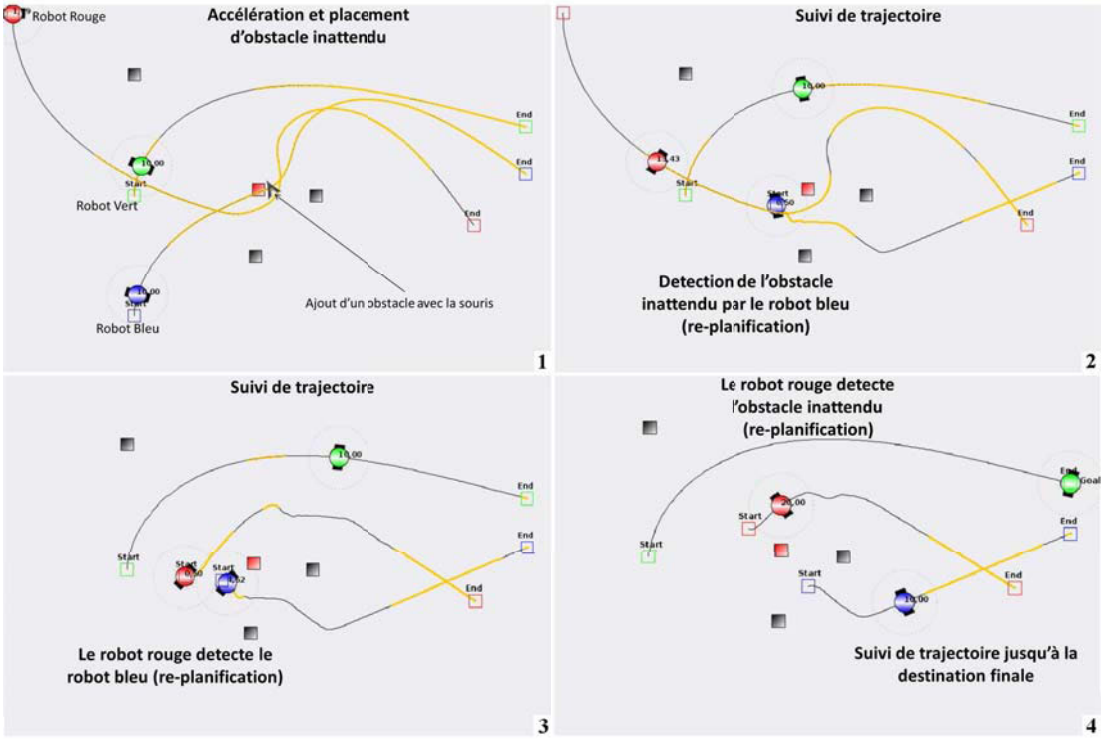


FIGURE 7.5 – Simulation du scénario Navigation avec événements inattendus.

La figure 7.5(1) représente le début de la navigation des trois robots dans leur environnement. Ils ont la même configuration que dans le premier scénario. On a placé néanmoins manuellement un obstacle inattendu dans le chemin des robots bleu et rouge.

On remarque que les vitesses attribuées par le superviseur sont différentes du premier scénario, les robots vert et bleu évoluent avec leurs vitesses max v_b =

$v_v = 10m/s$, tandis que la vitesse du robot rouge a été diminuée $v_r = 13.43m/s$. En effet, le superviseur a reçu en premier les requêtes des deux robots vert et bleu (grâce aux communications asynchrones émises par les robots comme dans le cas réel), et comme il n'a pas détecté de conflits entre eux, il les a laissé suivre leurs trajectoires avec leurs vitesses initialement assignées, la requête du robot rouge arrivant dernière, le superviseur en détectant des conflits entre le robot rouge et les deux autres a diminué sa vitesse pour les éviter.

Les trois robots continuent sur leurs trajectoires respectives jusqu'à la détection de l'obstacle inattendu par le robot bleu. Au moment de la détection de l'obstacle par le robot bleu (cf. figure 7.5(2)), refait une planification d'une nouvelle trajectoire en prenant en compte ce nouvel obstacle, celle-là va être envoyée au superviseur pour être analysée afin d'autoriser ou non le robot à la suivre.

Les autres robots pendant ce temps suivent leurs trajectoires respectives pour atteindre leurs destinations. Dans la figure 7.5(3) on voit que le robot rouge s'approche dangereusement du robot bleu (qui attend l'autorisation du superviseur). En le détectant, il le considère comme un obstacle inattendu et il va donc se comporter comme le robot bleu en faisant une re-planification supervisée par l'agent superviseur pour qu'il puisse continuer vers sa cible.

Une fois ces deux conflits résolus par la re-planification, les robots de l'organisation continuent sur leurs nouvelles trajectoires et atteignent finalement leurs destinations respectives (cf. figure 7.5(4)).

7.2 Navigation Distribuée

La navigation Distribuée est dédiée à des systèmes multi-robots évoluant dans des environnements dynamiques, avec un grand nombre d'obstacles ainsi qu'une forte incertitude et un grand nombre de robots dans un même environnement.

Dans notre cas de figure, la navigation distribuée est représentée par un comportement totalement détaché du superviseur, comme mentionné dans le chapitre 5, où les robots sont livrés à eux même en mode réactif, gérant les événements rencontrés lors de la navigation. Le principe de fonctionnement de ce type de navigation repose sur l'activation d'une seule norme proposée du modèle organisationnel (cf. chapitre 4), c'est la norme $NReactiveMode$.

Nous avons dédié un scénario pour illustrer le fonctionnement de la navigation distribuée, cela consiste en la navigation de trois robots se déplaçant en mode distribué dans un environnement en présence d'obstacles statiques et dynamiques, pour atteindre leurs destinations finales respectives.

Ce scénario a été simple à mettre en place et à simuler. On y voit trois robots évoluant dans l'environnement en mode réactif (cf. figure 7.6), en plus des obstacles statiques présents dans l'environnement, nous avons rajouté des obstacles inattendus. Les résultats obtenus sont représentés par les figures suivantes.

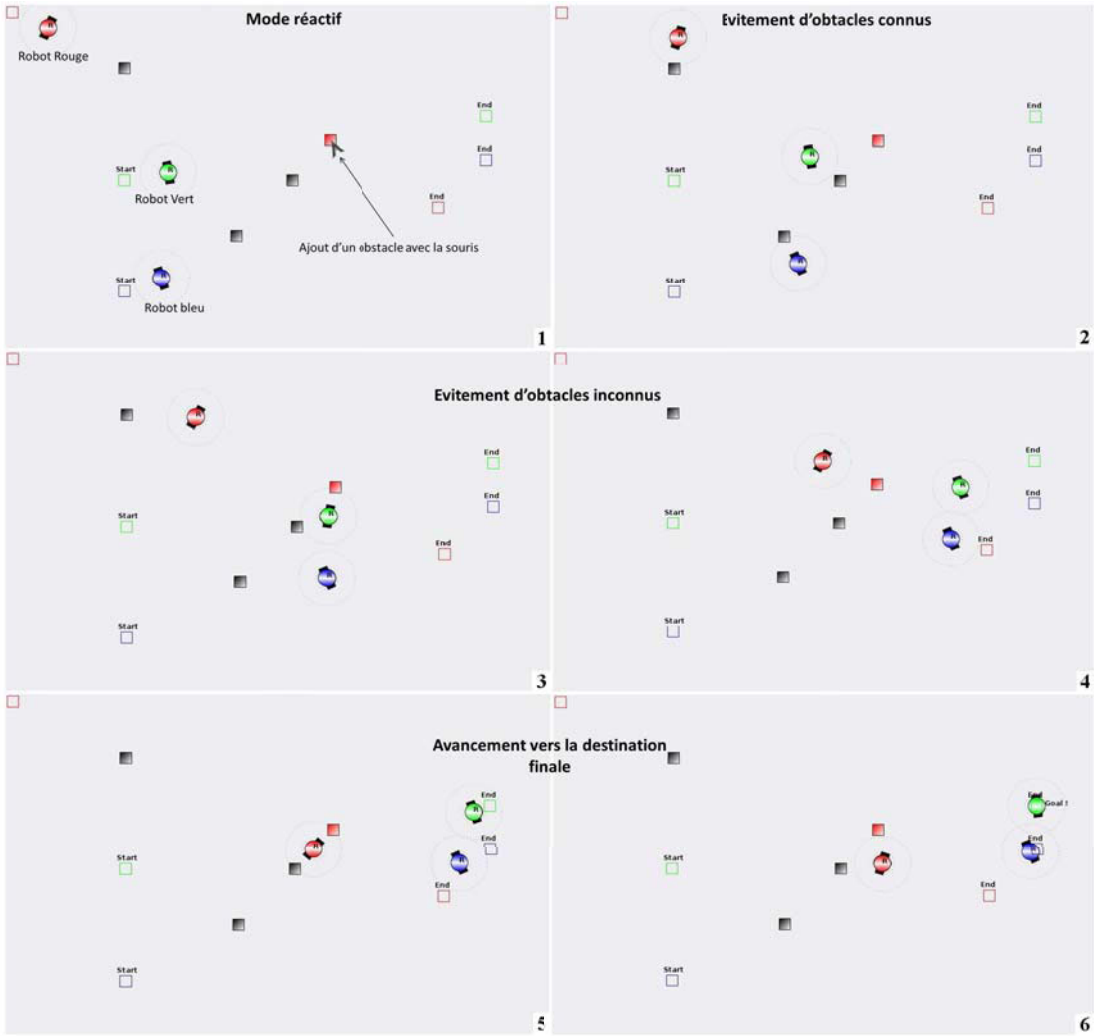


FIGURE 7.6 – Simulation du scénario de navigation Distribuée.

7.3 Navigation Centralisée/Distribuée

Le principe de fonctionnement de ce type de navigation repose sur l'activation de trois normes du modèle organisationnel utilisé, soit la norme *NSupervision*,

$NPlanningMode$ et $NReactiveMode$ (cf. chapitre 5). Les robots sont donc en mode planification supervisée par l'agent superviseur et les obstacles inattendus sont évités par le mode réactif suivi d'une re-planification de trajectoire. La figure 7.7 représente une illustration simplifiée du fonctionnement de ce genre de navigation centralisée/distribuée dans le contexte de la Planification et Re-Planification (PRP).

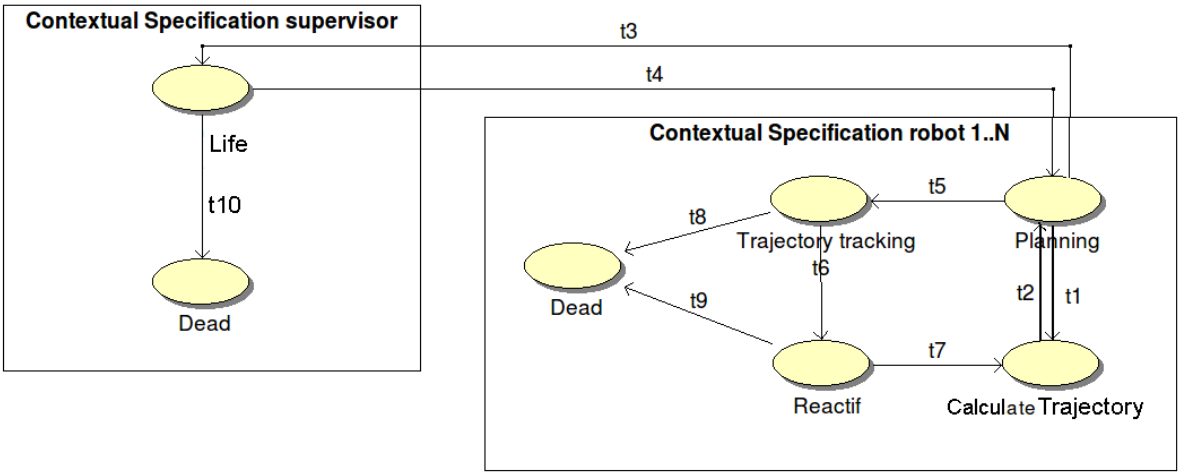


FIGURE 7.7 – Représentation simplifiée du fonctionnement de la navigation Centralisée/Distribuée PRP.

Dans la figure 7.7, chaque robot du SMR commence par une planification de sa propre trajectoire ($Planning$), puis la transmet au superviseur (transition t_3) qui l'analyse en modifiant la vitesse du robot si nécessaire, puis l'autorise ou pas à la suivre (transition t_4). Le robot procède donc au suivi de cette trajectoire grâce à la transition t_5. Dans le cas où le robot rencontre un obstacle inattendu pendant ce suivi de trajectoire, le robot va se mettre en mode réactif (transition t_6) pour l'éviter. Une fois l'obstacle évité le robot exécute la transition t_7 pour récupérer sa position actuelle ainsi que la liste actualisée des obstacles présents dans l'environnement et les envois (transition t_2) pour re-planifier une nouvelle trajectoire pour lui permettre d'atteindre sa destination, puis la transmet au superviseur (transition t_3) et ainsi de suite.

Ce mode de fonctionnement ayant recourt à l'évitement réactif, est utilisé principalement pour les environnements moyennement dynamiques où les obstacles sont assez prévisibles avec des mouvements peu aléatoires. Une fois l'environnement devient plus prévisible, on peut appliquer une re-planification.

Un schéma séquentiel illustrant étape par étape le fonctionnement des robots se trouvant en mode navigation centralisée/distribuée (PRP) dans ROBOTOPIA est présenté sur la figure 7.8.

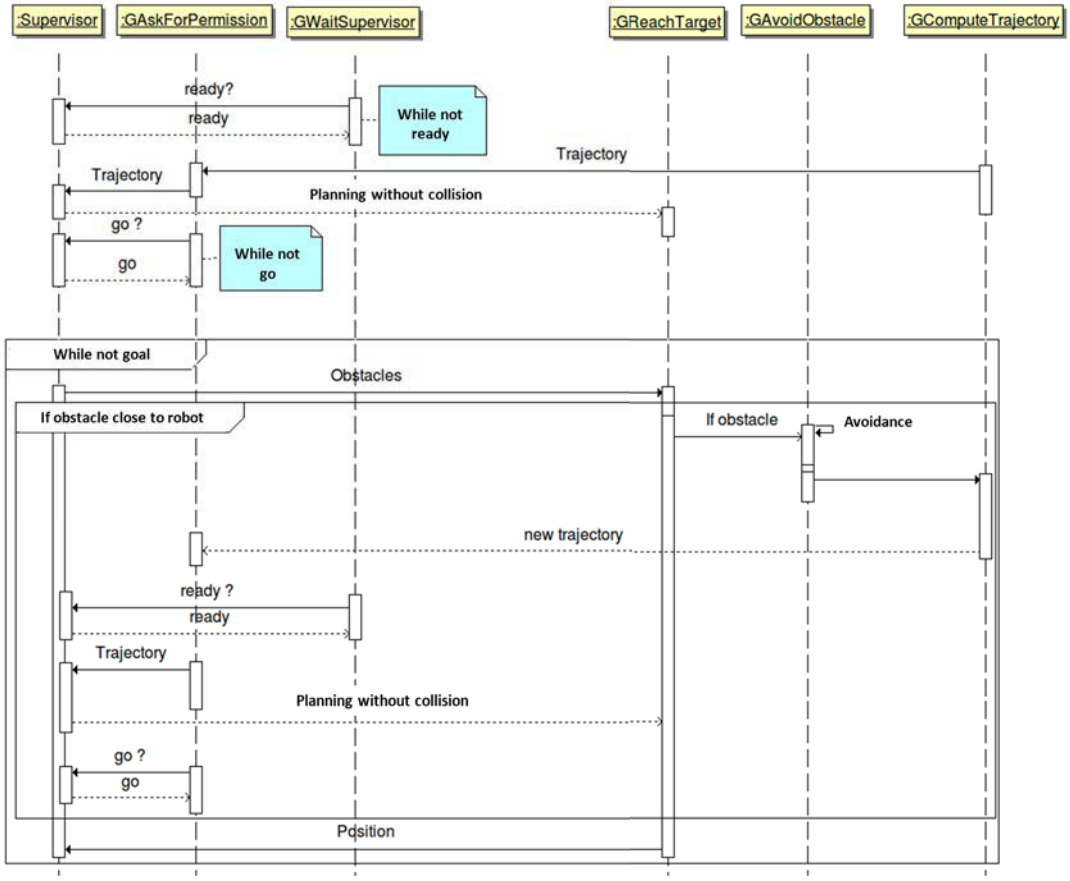


FIGURE 7.8 – Diagramme de séquence de la navigation Centralisée/Distribuée PRP.

Pour illustrer cette navigation Centralisée/Distribuée, nous nous proposons d'étudier le cas de trois robots mobiles navigants en coordination dans un environnement contenant des obstacles **statiques** et **dynamiques**, pour atteindre trois destinations différentes.

Dans ce scénario nous allons procéder à une simulation similaire à celle de la section 7.1.1, où la configuration initiale de l'environnement (robots, obstacles) est identique, mais on remarque que les résultats sont différents.

La figure 7.9(1) représente le début de la navigation des trois robots dans leur environnement, avec un placement manuel d'obstacles inattendus.

Les trois robots continuent sur leurs trajectoires respectives (cf. figure 7.9(2)) jusqu'à la détection des obstacles inattendus par les robots concernés (bleu et vert). La figure 7.9(3) et (4) montre la détection de l'obstacle par les robots bleu et vert, à ce moment ils se mettent en mode réactif le temps d'éviter l'obstacle inattendu.

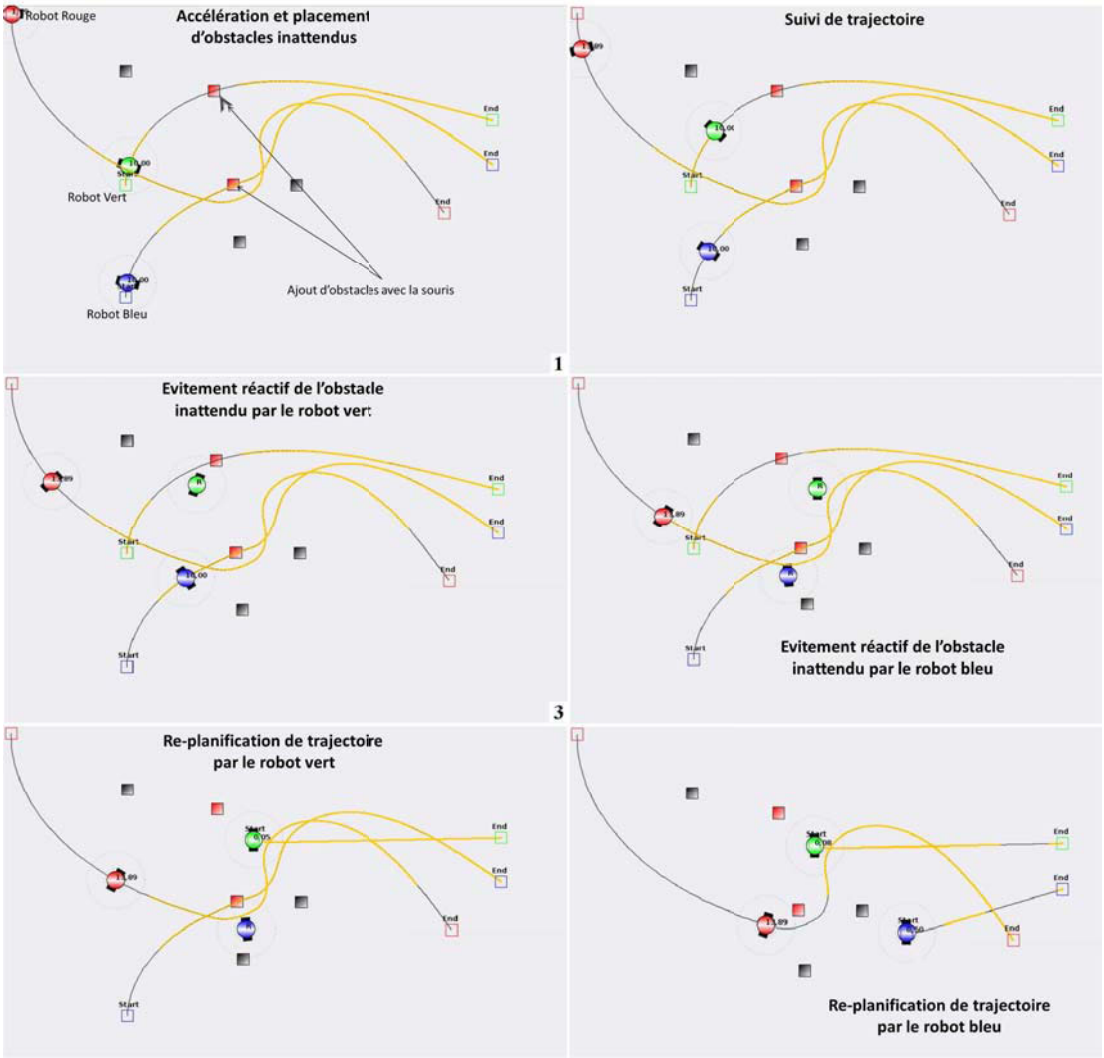


FIGURE 7.9 – Simulation du scénario Navigation Centralisée/distribuée.

Une fois l'évitement réactif terminé, les robots concernés re-planifient une nouvelle trajectoire pour leurs permettre d'atteindre leurs destinations respectives, comme illustré sur la figure 7.9(5) et (6).

Après validation des trajectoires planifiées par le superviseur, les robots se mettent à suivre ces nouvelles trajectoires calculées, jusqu'à atteindre leurs cibles.

7.4 Navigation en formation

La navigation en formation d'un groupe de robots mobile, représente un des défis que nous tenons à réussir à travers l'architecture de contrôle/commande proposée. En effet, ce genre de navigation est utile dans de nombreuses tâches de coopération telles que la tonte d'un terrain de sport, le transport ou manipulation d'objets impliquant conjointement plusieurs robots mobiles dans l'exécution des tâches requises.

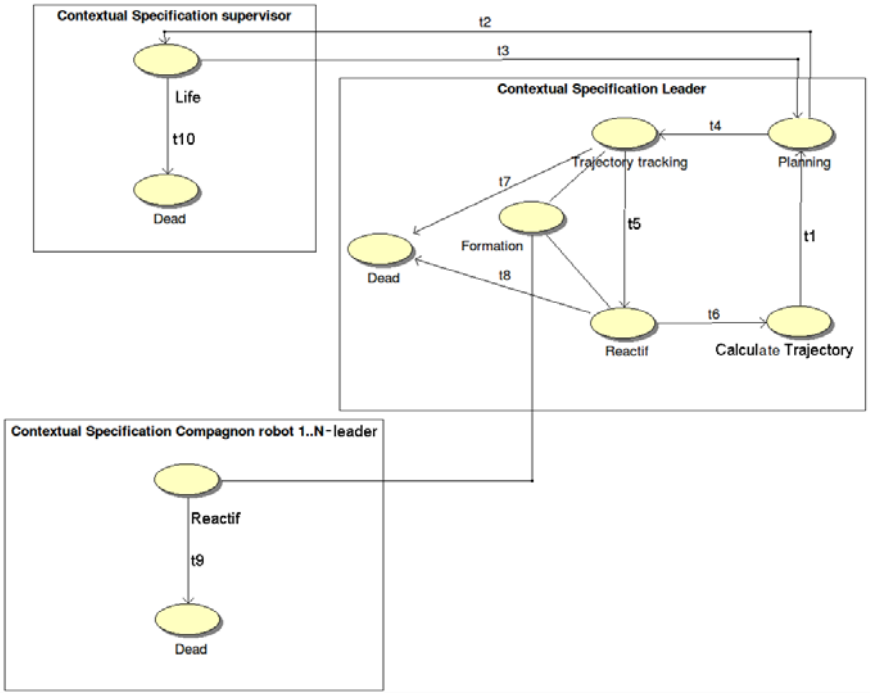


FIGURE 7.10 – Représentation simplifiée du fonctionnement de la navigation en formation Centralisée/Distribuée.

Un groupe de robots mobiles naviguent en maintenant des formes géométriques prédéterminées (ligne, colonne, triangle, etc.), doit contrôler l'emplacement de chaque robot par rapport aux autres, et doit en permettant de déplacer le groupe dans son ensemble. Cette navigation en formation doit aussi garantir l'évitement des obstacles présents dans l'environnement.

Pour aboutir à cela, nous avons référencé ce type de navigation dans la catégorie de la navigation Centralisée/Distribuée, car on a besoin de l'aspect centralisé aussi bien que l'aspect distribué. Dans notre architecture, le principe de fonctionnement de ce type de navigation repose sur l'activation de quatre normes du modèle organisationnel utilisé : $NSupervision$, $NFormationMode$, $NPlanningMode$, et $NReactiveMode$ (cf. chapitre 5). Les robots sont en mode formation où le leader évolue en mode Planification/Re-Planification supervisée tandis que les robots suiveurs sont en mode réactif suivant une cible dynamique leurs permettant de maintenir la formation. La figure 7.10 représente une illustration simplifiée du fonctionnement de ce genre de navigation en formation Centralisée/Distribuée.

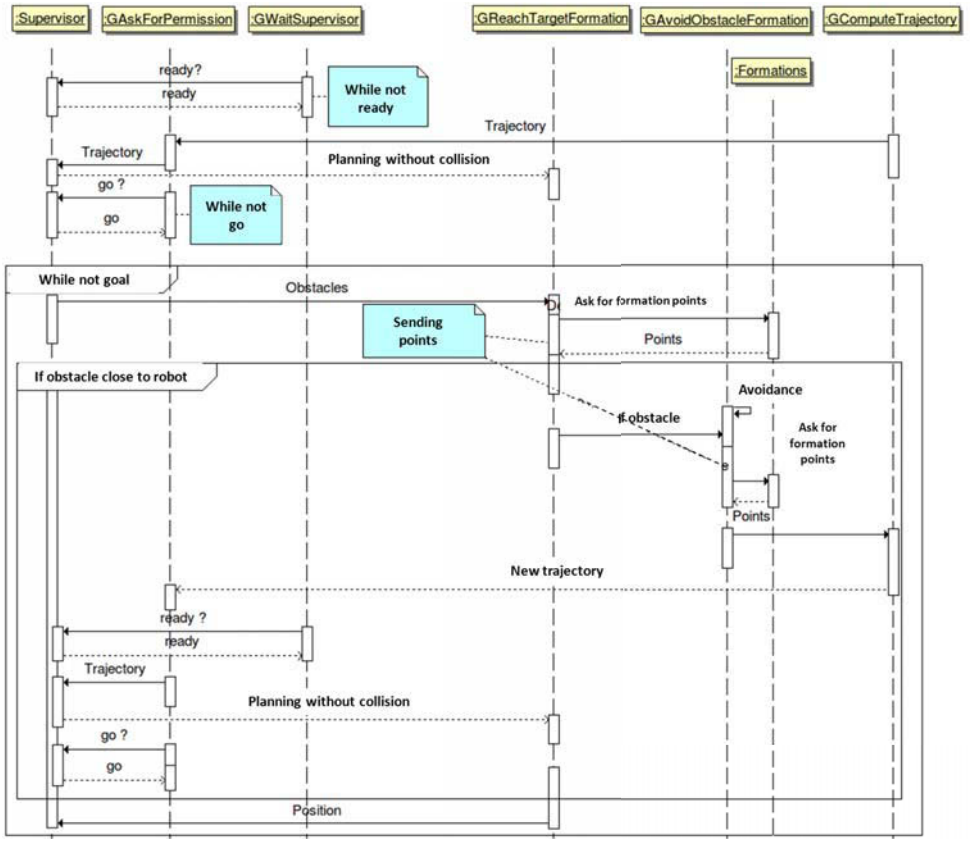


FIGURE 7.11 – Schéma de fonctionnement temporel de la navigation en formation Centralisée/Distribuée.

Dans la figure on remarque que pendant le suivi de la trajectoire par le leader il fait appel au module de génération de formation qui envoi en temps réel les positions des cibles dynamiques aux robots suiveurs pour maintenir leurs positions

dans la formation.

Un schéma temporel illustrant étape par étape le fonctionnement des robots se trouvant en mode navigation en formation centralisée/distribuée dans ROBOTOPIA est présenté sur la figure 7.11.

Nous avons dédié deux scénarios pour illustrer le fonctionnement de la navigation en formation centralisée/distribuée :

— formation avec événements inattendus : trois robots mobiles navigant en formation *triangle* dans un environnement contenant des obstacles statiques et dynamiques (inconnus par le leader), pour permettre au leader de la formation d'atteindre sa destination.
— formation sans événements inattendus : trois robots mobiles navigant en formation *triangle* dans un environnement contenant des obstacles statiques, connus à l'avance par le leader pour planifier sa trajectoire sans aucune difficulté.

Dans ce qui suit nous allons présenter les résultats de simulation de ces deux scénarios.

7.4.1 Formation avec événements inattendus par le leader

Dans ce scénario nous allons positionner les robots avec les mêmes conditions initiales que dans le scénario précédent, en rajoutant des obstacles inattendus pendant l'évolution de la formation dans l'environnement.

La figure 7.12(1) représente le début de la navigation en formation, où on voit que le leader commence à suivre sa trajectoire et les robots suiveurs entrain de rejoindre leurs positions respectives dans la formation (en mode réactif). On procède aussi au placement d'un obstacle inattendu dans la trajectoire du leader.

Le robot leader en détectant cet obstacle inattendu procède à son évitement en se mettant en mode réactif le temps de l'évitement (cf. figure 7.12(2)), puis re-planifie une nouvelle trajectoire qui lui permettra de rejoindre sa destination finale.

Comme précédemment expliqué, le leader en mode formation génère les positions de formation quel que soit son mode de fonctionnement, nous remarquerons alors sur les figures 7.12(3) et (4) que le leader en mode réactif continue de générer les positions de la formation triangle et les robots suiveurs maintiennent cette formation pendant cet évitement.

Une fois cet obstacle évité avec succès par le robot leader, celui-ci re-planifie une nouvelle trajectoire lui permettant d'atteindre sa destination finale suivi par les robots suiveurs (cf. figure 7.12(5), (6)).

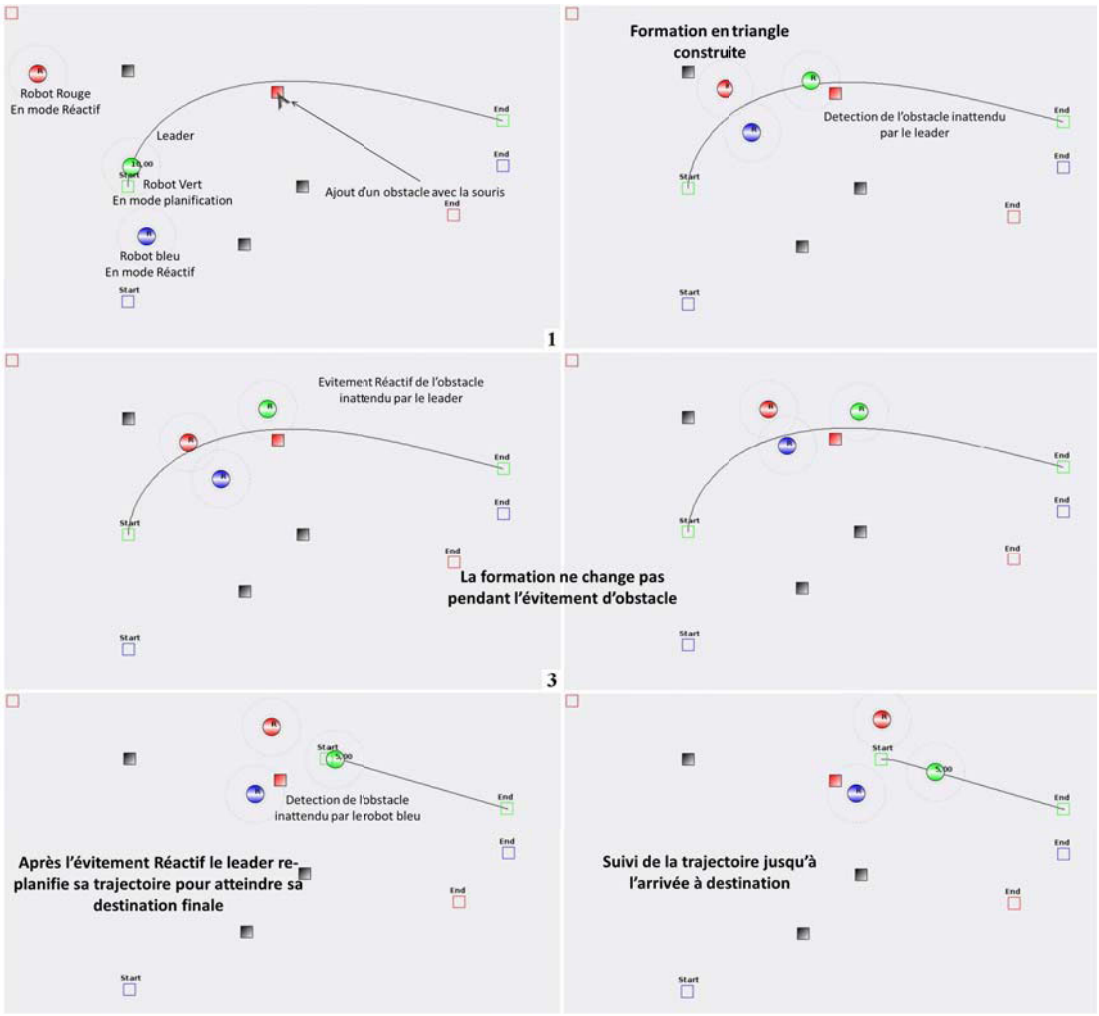


FIGURE 7.12 – Simulation du scénario formation Centralisée/Distribuée avec événements inattendus.

7.4.2 Formation sans événements inattendus par le leader

La figure 7.13(1) représente le début de la navigation des trois robots, où le robot vert est le leader de la formation en mode planification supervisée accélérant pour suivre sa trajectoire. Les robots suiveurs (le bleu et le rouge) sont en mode réactif tendent à rejoindre leurs positions dans la formation. La figure 7.13(2) représente les trois robots naviguant en formation triangle (où les robots bleu et rouge suivent leur leader).

Pendant cette navigation en formation le robot rouge (en mode réactif) ren-

contre un obstacle statique présent dans l'environnement (cf. figure 7.13(3) et (4)), il quitte alors sa position dans la formation le temps d'éviter l'obstacle en mode réactif puis rejoint à nouveau sa position dans la formation.

Une fois cet obstacle évité avec succès les robots en formation continuent leurs avancée jusqu'à ce que le leader atteigne sa destination finale souhaitée (cf. figures 7.13(5), et (6)).

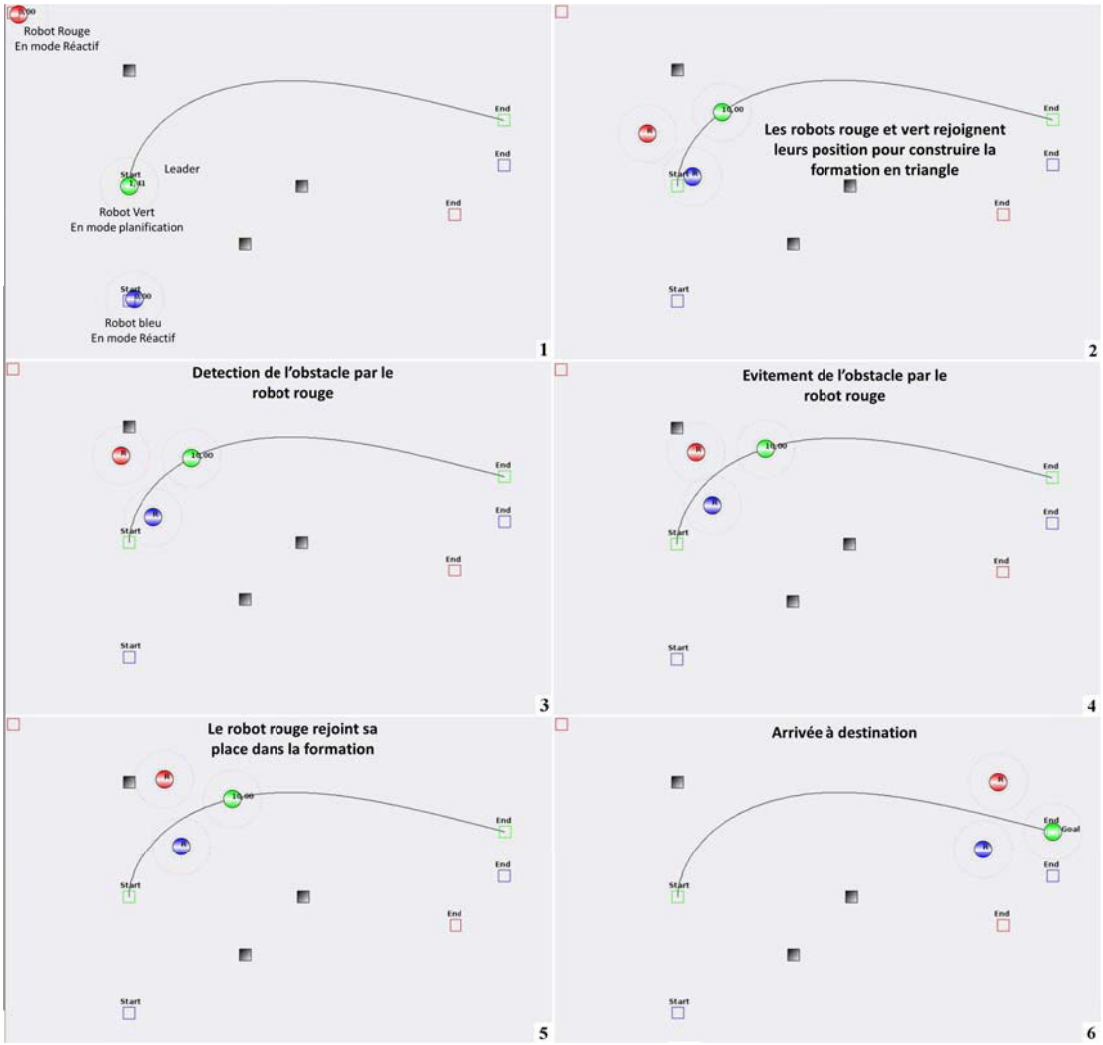


FIGURE 7.13 – Simulation du scénario formation Centralisée/Distribuée sans événements inattendus.

7.5 Conclusion

Dans ce chapitre nous avons procédé à la simulation de l'architecture de contrôle/commande proposée via le simulateur ROBOTOPIA. On a pu alors simuler les différents modes de fonctionnement de celle-ci soit la navigation Centralisée, la navigation Distribuée et la navigation Centralisée/Distribuée, à travers différents scénarios d'exécutions, ce qui nous a permis de valider le bon fonctionnement de l'architecture proposée.

En effet, une fois toutes les composantes de cette architecture développées et codés au sein de ROBOTOPIA, il nous a fallu procéder à des petites modifications de la structure XML du simulateur pour arriver à reproduire une multitude de scénarios de navigation de systèmes multi-robots dans des environnements connus ou peu connus. On a aussi réussi à valider deux scénarios de navigation en formation d'un groupe de robots mobiles.

L'organisation de notre architecture la rend hétérogène, elle permet de combiner tous les aspects de contrôle/commande précédemment décrits dans les Chapitres 3 et 4 en une seule exécution. Cela permet par exemple de gérer à la fois plusieurs SMR évoluant dans le même environnement avec différentes missions et cibles. Une illustration en simulation de cela est présentée dans l'annexe C. On a ainsi obtenu une architecture de contrôle/commande ouverte et hétérogène destinée à gérer des systèmes multi-robots autonomes dans des environnements statiques ou dynamiques.

Conclusion générale et perspectives

Conclusion générale

Les travaux réalisés dans cette thèse répondent à la volonté d'investiguer les architectures de contrôle pour les systèmes multi-robots mobiles. L'intérêt est porté plus particulièrement sur la coordination et la coopération des entités robotiques autonomes dans le but de réaliser leurs tâches respectives.

Dans le premier chapitre de ce manuscrit, nous avons d'abord abordé la robotique mobile d'une manière générale et les enjeux liés à la coopération d'un groupe d'entités autonomes. Au vu du contexte des travaux de thèse relatifs au contrôle/-commande d'un Système Multi-Robots (SMR), l'état de l'art réalisé s'est focalisé particulièrement sur les architectures de contrôle/commande dédiées à ces systèmes ainsi que les diverses méthodologies de coopération existantes. L'objectif étant de s'inspirer des approches traitées dans la littérature afin de proposer une architecture la plus appropriée pour le contrôle du SMR.

L'aspect hybride, notamment pour l'autonomie décisionnelle qu'il offre aux entités, se repose sur la combinaison des différents aspects de contrôle/commande existants, à savoir les aspects centralisés, distribués, cognitifs et réactifs permettant une meilleure gestion au sein de l'architecture de contrôle. Se plaçant dans ce cadre, dans le troisième chapitre nous nous sommes focalisés sur les approches multi-agents pour apporter de la coopération et de la coordination à notre système multi-robots mobiles. Nous avons opté spécialement pour les modèles organisationnels multi-agents, qui grâce à leur organisation sous forme de spécifications, nous permettent de décomposer le niveau de coordination en plusieurs spécifications selon les contextes, les fonctionnalités, les structures et les normes liées à notre groupe d'agents/robots mobiles.

De cette façon nous avons atteint l'objectif d'ouverture et de flexibilité désirées de notre architecture de contrôle en choisissant une approche hybride combinée avec un modèle organisationnel multi-agents pour contrôler les entités robotiques. Cette architecture nous offre la possibilité de mettre à jour le contrôle des robots selon la tâche qu'ils seront amenés à exécuter. Cette approche consiste à traduire la tâche globale en quatre spécifications du niveau de coordination de l'architecture proposée, à savoir : la spécification structurelle, la spécification fonctionnelle, la spécification contextuelle et la spécification normative. L'architecture de contrôle dispose aussi dans le niveau de contrôle d'un ensemble de comportements/contrôleurs, permettant à chacun d'accomplir une tâche élémentaire donnée. L'architecture proposée : ouverte, hétérogène et flexible permet ainsi la prise en compte de différents contextes de fonctionnement comme la planification et la re-planification de trajectoires, ainsi que la navigation en formation en présence d'obstacles statiques et dynamiques.

Dans le troisième et le quatrième chapitre, nous avons présenté notre architec-

ture de contrôle proposée. Elle comprend principalement deux niveaux : un niveau de coordination et un autre dédié au contrôle. Le niveau de coordination comprend les quatre spécifications du modèle organisationnel $\mathcal{M}$OISE^{Inst} permettant de coordonner notre SMR. Le niveau de contrôle quant à lui est composé de plusieurs modules et contrôleurs de planification et re-planification de trajectoire, de suivi de trajectoires, d'évitement réactif d'obstacles et de maintien de formation, auxquels on fait appel chaque fois que c'est nécessaire. Par ailleurs, nous présentons dans cette thèse, une nouvelle approche de planification et re-planification de trajectoires grâce à la supervision des systèmes multi-agents et la réactivité des contrôleurs, assurant au SMR une navigation sécurisée dans un environnement en présence d'obstacles statiques et/ou dynamiques. Nous proposons également une approche de navigation en formation de plusieurs groupes de robots mobiles dans un même environnement en présence d'obstacles, une illustration du fonctionnement de notre architecture de contrôle/commande dans différents modes de fonctionnement est présentée dans le chapitre cinq.

Le simulateur ROBOTOPIA présenté dans le sixième chapitre de ce manuscrit, a été développé et mis en place comme exemple de l'implémentation et d'instanciation du modèle de l'architecture de contrôle proposée. ROBOTOPIA a servi de base afin de valider notre approche dans un environnement de simulation, ce qui nous a permis de tester notre modèle d'architecture proposée. Grâce à ROBOTOPIA nous avons mis en place, dans le septième chapitre, une multitude scénarios de fonctionnements SMR et leur résultat de simulation afin d'être le plus exhaustif possible dans l'évaluation de la flexibilité, l'ouverture et l'hétérogénéité de l'architecture de contrôle proposée.

Perspectives

Ces travaux de thèse ont permis d'ouvrir le champ à des perspectives variées et étendues à l'image du thème de recherche abordé. Ces perspectives peuvent être classées en deux catégories principales : perspectives relatives aux contributions théoriques et celles liées aux expérimentations.

Perspectives théoriques : Dans l'architecture de contrôle proposée nous avons employé un modèle organisationnel basé sur les systèmes multi-agents, ce qui nous a permis d'avoir une bonne répartition du contrôle, ainsi nous avons pu avoir un système hétérogène qui nous a permis la coordination de la navigation de plusieurs robots mobiles (en formation ou pour atteindre une cible). Les perspectives théoriques qui s'inscrivent dans la continuité de nos travaux de recherche sont :

— l'amélioration de notre système de coopération en y ajoutant d'autres types de coordination et coopération de robots comme la cartographie de terrains, les travaux de manutention coopérative à l'aide de bras manipulateurs, etc. Cela permettra d'apporter les performances des modèles organisationnels à ces différents domaines.
— l'investigation du domaine de la communication inter-robots pour avoir une meilleure transmission de l'information dans la flottille.
— étendre les travaux de recherche à d'autres types de modèles d'organisation et les adapter à d'autres types de robots : marins, sous-marin et aériens.

En améliorant ainsi notre architecture, nous obtiendrons une architecture de contrôle hybride multidisciplinaires permettant de passer facilement d'un mode de fonctionnement à un autre.

Perspectives expérimentales : Dans les travaux présentés dans cette thèse nous avons évalué l'architecture de contrôle proposée seulement en simulation. En effet, la perspective de l'implémentation de cette architecture dans un milieu physique, et sur de vrais robots, spécialement dans des milieux de transport et déplacement d'objets ou de marchandises comme les entrepôts peuplés de robots mobiles, nous permettra de mieux évaluer ses performances, et aussi de soulever et de résoudre d'autres problématiques théoriques et expérimentales.

L'expérimentation permettra ainsi d'éprouver notre proposition d'architecture de contrôle/commande sur plusieurs points de vue dont :

— au point de vue algorithmique on pourra évaluer l'adéquation entre les besoins en calculs et les possibilités de traitements limitées imposées par le contexte embarqué de l'étude. La flottille de drones aériens sera l'épreuve idéale, du fait de la complexité des calculs nécessaires à la résolution des équations de trajectoires (robot évoluant dans un volume tridimensionnel), la diminution du temps moyen de réaction, ressource énergétique limitées influant sur les capacités de traitement embarquable.
— au point de vue de la robustesse, le test en milieu sous-marin nous permettra de mettre en évidence la capacité de l'architecture à maintenir ses performances dans un environnement non précisément connu et en présence d'erreurs de mesures. En effet le milieu marin est en constante évolution et induisant mêmes des déplacements aléatoires sur les véhicules immergés, même en l'absence de mouvements de ces derniers. De plus sa non-perméabilité a l'onde électromagnétique nous prive de la précision des capteurs laser, infrarouge et des caméras au profit de sondeurs acoustiques dont l'onde, lente, interférente et n'évoluant pas de façon rectiligne (diffraction par les couches d'eau aux températures différentes) nous apporte que de maigres informations sur l'environnement. La communication inter-véhicule devra aussi se limiter au strict minimum au vu des capacités

restreintes de transport de l'information dans l'eau (communication sonore à bas débit).

Annexes

Annexe A

Algorithme du cycle limite réactif pour l'évitement d'obstacles

Cette annexe reprend l'algorithme d'évitement d'obstacles basé sur la méthode du cycle-limite [Adouane 09]. Pour accomplir la tâche d'évitement, il est impératif de définir trois étapes essentielles :

— détecter l'obstacle à éviter,
— choisir une direction d'évitement (trigonométrique ou anti-trigonométrique),
— définir un critère pour évaluer si l'obstacle est considéré comme étant évité.

Naturellement, ces étapes doivent prendre en considération les situations indésirables comme les minima locaux, d'inter blocage entre plusieurs obstacles, les oscillations dues aux multiples transitions entre le contrôleur d'attraction vers la cible et celui de l'évitement d'obstacles, etc.

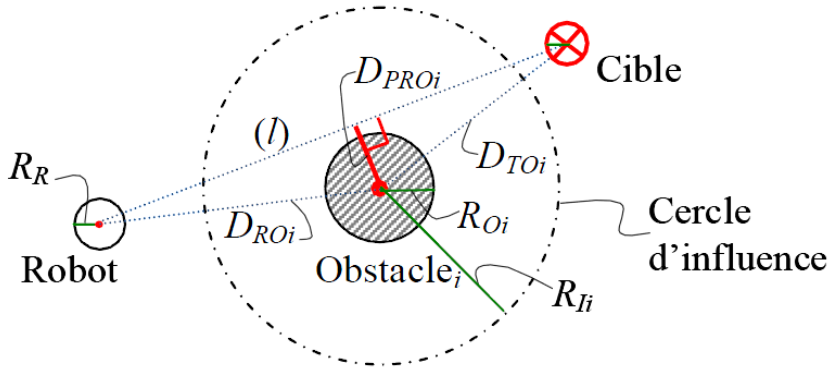


FIGURE A.1 – Grandeurs nécessaires à l'évitement de l'obstacle i.

Avant d'expliquer ces situations indésirables et les solutions apportées, on propose d'énumérer les points essentiels avant d'entamer l'algorithme d'évitement :

1. à chaque instant d'échantillonnage, obtenir les distances D_{RO_i} et D_{lRO_i} (cf. figure A.1) pour tout obstacle gênant i (un obstacle est considéré gênant si $D_{lRO_i} \leq R_{I_i}$),
2. parmi les obstacles gênants, choisir l'obstacle le plus proche (ayant la plus petite valeur D_{RO_i}). Il possède un rayon R_{O_i} et la position (x_{obst}, y_{obst}),
3. définir quatre zones déterminant le comportement du robot (sens d'évitement, phase d'attraction vers l'obstacle, phase de répulsion) (cf. Figure A.2), (cf. Algorithme 3). Ces zones sont distinguées grâce à un repère orthonormé direct (O_i, X_{O_i}, Y_{O_i}) dont l'axe $(O_i X_{O_i})$ a le centre de l'obstacle O_i pour origine et passe par la cible du robot.
4. passer de la position du robot $(x, y)_A$ donné initialement dans le repère absolu à la position $(x, y)_O$ dans le repère (O_i, X_{O_i}, Y_{O_i}) grâce à une matrice de transformation comme suit :

$$\begin{pmatrix} x \\ y \\ 0 \\ 1 \end{pmatrix}_O = \begin{bmatrix} \cos\alpha_i & -\sin\alpha_i & 0 & x_{obst} \\ \sin\alpha_i & \cos\alpha_i & 0 & y_{obst} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{pmatrix} x \\ y \\ 0 \\ 1 \end{pmatrix}_A \tag{A.1}$$

où α_i est l'angle que fait l'axe $(O_i X_{O_i})$ avec $(O_m X_m)$ (cf. Figure A.2).

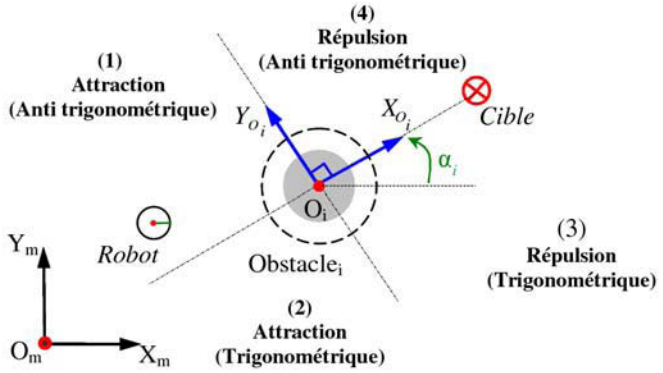


FIGURE A.2 – Zones entourant l'obstacle et déterminant la manière d'évitement de l'obstacle.

Le repère relatif à l'obstacle donne de nombreux outils d'aide à la décision au robot : direction d'évitement, phase d'attraction ou de répulsion, etc.

L'utilité de ces deux dernières phases est d'éviter des oscillations indésirables que l'on illustre ci-après (cf. Section A). Le signe de l'ordonnée du robot dans ce repère y_O permettra de savoir s'il faut éviter le robot dans le sens trigonométrique ou anti-trigonométrique :

— si $y_O \geq 0$ alors l'obstacle est évité dans le sens anti-trigonométrique,
— si $y_O < 0$, il est évité dans le sens trigonométrique.

Ceci permet alors d'optimiser la trajectoire du robot pour éviter l'obstacle et rejoindre plus rapidement sa cible. De même, grâce au signe de l'abscisse du robot x_O, il est possible de savoir s'il est en phase de répulsion ou d'attraction :

— $x_O \leq 0$ correspond à la phase d'attraction,
— $x_O > 0$ correspond à la phase de répulsion.

Ces étapes de prise de décision sont résumées dans l'algorithme 3.

Entrées: Caractéristiques de l'obstacle le plus proche : position (x_{obst}, y_{obst}), rayon R_{I_i}
Sorties: l'angle consigne $\theta_{S_{oa}}$

1: **Si** $x_O \leq 0$ **alors**
2: $R_c = R_{I_i} - \xi$ (phase d'attraction) ; {ξ est une constante telle que $\xi \ll Marge$.} Cette phase permet au robot de ne pas se trouver au voisinage de R_{I_i} ce qui risque de provoquer des oscillations
3: **Sinon**
4: $R_c = R_c + \xi$ (phase de répulsion) ;{Critère de sortie de la zone de l'obstacle en gardant une trajecoitre lisse}
5: **finSi**
6: **Si** le contrôleur d'évitement d'obstacles est actif à l'instant $(t-1)$ **alors**
7: Appliquer le même sens d'évitement qu'à l'instant t (anti-trigonométrique ou trigonométrique) ;{voir le paragraphe A}
8: **Sinon**
9: Le système d'équation du cycle-limite :
$\dot{x}_s = signe(y_O)y_s + x_s(R_c^2 - x_s^2 - y_s^2)$
$\dot{y}_s = -signe(y_O)x_s + y_s(R_c^2 - x_s^2 - y_s^2)$
{avec $signe(y_O) = 1$ si $y_O \geq 0$ et $signe(y_O) = -1$ sinon.}
10: **finSi**
11: $\theta_{S_{oa}} = \arctan(\frac{\dot{y}_s}{\dot{x}_s})$

Algorithm 3: Algorithme d'évitement d'obstacles [Adouane 09].

Gestion des situations de conflit

La performance de l'algorithme 3 réside dans son aptitude à gérer certaines situations de conflits ou de comportements indésirables.

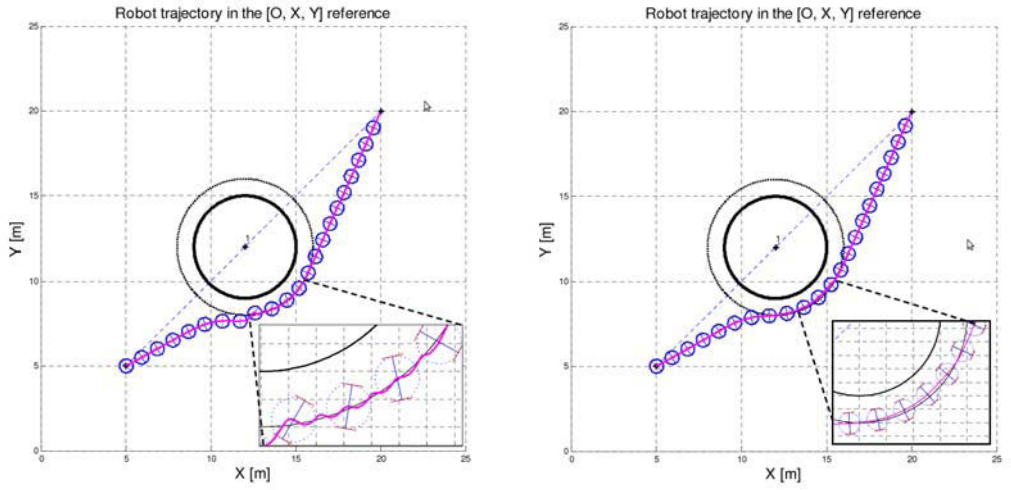
Risque d'oscillation au voisinage de R_{I_i} Les phases d'attraction et de répulsion décrites précédemment, dont les zones sont illustrées dans la figure A.2, et exprimées dans l'algorithme 3 par les lignes 2 et 4 servent à éviter ces oscillations entre $D_{ROi} \leq R_{Ii}$ (activation du contrôleur d'évitement d'obstacles) et $D_{ROi} \geq R_{Ii}$ (activation du contrôleur d'attraction vers la cible). En effet, la zone d'attraction permet d'entrer progressivement dans la zone de l'obstacle en décrémentant le rayon du cycle-limite de ξ à chaque itération, ce qui limite le risque de se trouver au voisinage de R_{Ii}. La zone de répulsion permet de préparer le robot à sortir de la zone de l'obstacle en gardant une trajectoire lisse et d'atteindre sa cible. A noter qu'afin d'éviter que le rayon du cycle-limite ne soit trop décrémentée, ce qui peut provoquer la collision, une limite peut être imposée telle que la ligne 4 de l'algorithme 3 n'est plus exécutée si

$$R_c \leq R_l$$

où R_l est un rayon limite à ne pas franchir.

Les oscillations causées par l'utilisation directe du rayon R_{I_i} comme rayon du cycle limite est illustrée dans la figure A.3(a). La figure A.3 montre que ces oscillations sont éliminées grâce aux phases d'attraction et de répulsion de l'algorithme 3.

Choix de l'obstacle à éviter S'il arrive qu'il y ait deux obstacles ou plus qui ont la même distance par apport au robot, alors il y a un conflit et le robot risque



(a) Avec oscillations (b) Sans oscillations

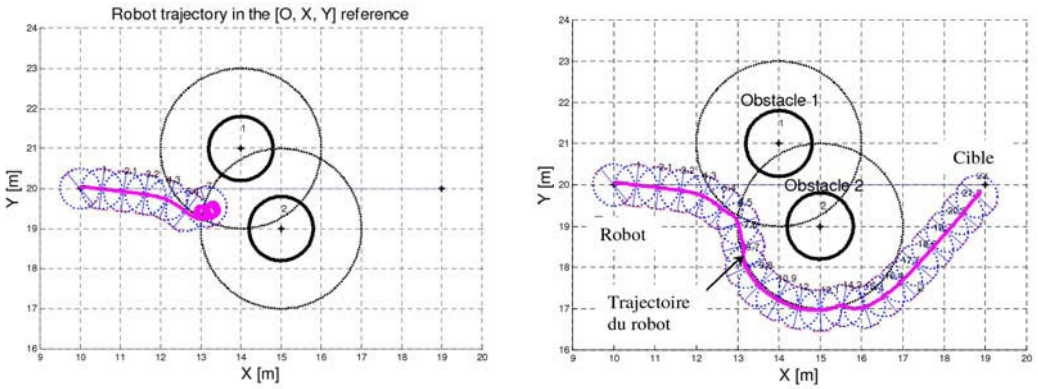
FIGURE A.3 – Evitement des oscillations de la trajectoire du robot grâce à l'algorithme 3 [Adouane 09].

de ne pouvoir choisir l'obstacle à éviter, l'algorithme 4 permet de régler ce conflit.

1: **Si** deux ou plusieurs obstacles ont la même distance D_{RO_i} **alors**
2: le robot choisira celui avec le plus petit D_{lRO_i} ;
3: **finSi**
4: **Si** ils ont le même D_{lRO_i} **alors**
5: le robot choisira celui avec le plus petit D_{CO_i} ;
6: **finSi**
7: **Si** ils ont le même D_{CO_i} **alors**
8: choisir l'un des obstacles arbitrairement ;
9: **finSi**

Algorithm 4: Choix de l'obstacle à éviter [Adouane 09].

Problèmes des minima locaux et des obstacles de type U (impasses) La ligne 7 de l'algorithme 3 permet de forcer la direction du cycle-limite afin d'éviter les minima locaux. En effet, un minimum local peut apparaître si le robot doit éviter deux obstacles qui se chevauchent. S'il évite le premier dans le sens trigonométrique, et le deuxième dans le sens inverse en suivant uniquement les quatre zones définies par le repère relatif à l'obstacle (cf. Figure A.2), il se trouvera bloqué entre les deux comme illustré sur la figure A.4(a). Dans [Kim 03], ce problème est géré en imaginant un seul obstacle virtuel englobant tous les obstacles chevauchés. Cependant, ceci suppose que lorsque le robot est face au premier obstacle, il connaît déjà la totalité des obstacles chevauchés. De plus, plusieurs obstacles peuvent générer un obstacle virtuel avec un rayon important qui n'est pas toujours justifié. Vu toutes ces informations nécessaire au préalable, la méthode risque de devenir moins réactive que l'on le souhaite. Quant à l'algorithme 3, il se base sur une information mémoire minimale gardant uniquement le sens d'évitement à l'instant d'échantillonnage précédent. Grâce à cette information minimale, le robot peut aussi sortir des impasses imposées par les obstacles de type U (cf. Figure A.5).



(a) Sans imposer la direction, le robot tombe dans un minimum local

(b) En imposant la direction

FIGURE A.4 – Gestion du minimum local [Adouane 09].

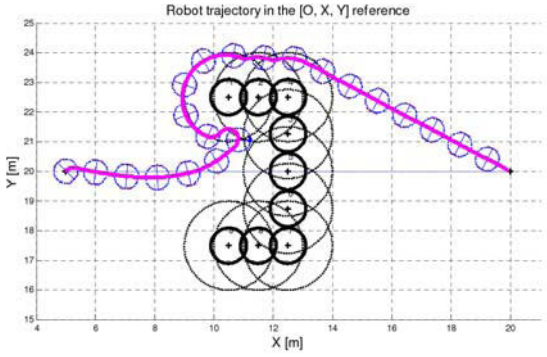


FIGURE A.5 – Gestion des impasses provoquées par les obstacles de forme U [Adouane 09].

Annexe B

Les graphes récursifs

On peut définir un graphe G de la manière suivante :

$G(V, E) = (\{v_1, v_2, .., v_n\}, \{\{v_i, v_j\}/v_i \in V, v_j \in V\})$

V est l'ensemble des sommets du graphes et E l'ensemble de ses arêtes qui sont des ensembles de deux sommets.

Les hypergraphes introduits par Claude Berge [Berge 87] généralisent la notion d'arête d'un graphe :

$H(V, E) = (\{v_1, v_2, .., v_n\}, \{E_1, E_2, ..., E_i, ..., E_m\})$ où E est une famille de parties E_i non vides de V.

Pour formaliser la notion de graphe récursif nous utilisons des hypergraphes particuliers dont les arêtes sont toutes des triplets. Cependant, ce ne sont pas trois sommets du graphe mais un ensemble de deux sommets et un **type** t. Ce n'est donc pas formellement un hypergraphe :

$H(V, E, T) = (\{v_1, v_2, .., v_n\}, \{(t, \{v_i, v_j\})/t \in T, v_i \in V, v_j \in V\}, \{t_1, t_2, ..., t_p\})$

Un tel graphe nous permet, avec un même ensemble de sommets, de gérer plusieurs types d'arêtes, ce qui nous évitera de gérer un graphe pour chaque type de relation.

Finalement un graphe récursif $R(V, E, T)$ est composé d'un ensemble d'autres graphes récursifs $V = \{R_1, R_2, ..., R_N\}$, d'un ensemble d'arêtes entre ces graphes récursifs $E = \{(t, \{R_i, R_j\})/t \in T, R_i \in V, R_j \in V\}$ et d'un ensemble de p types $T = \{t_1, t_2, ..., t_p\}$ spécifiant les arêtes.

On désigne par sommet atomique le graphe récursif $R_a(\{\emptyset\}, \{\emptyset\}, \{\emptyset\})$.

Un graphe récursif ne comportant que des sommets atomiques permet de représenter tout graphe non récursif.

Cette structure de données extrêmement générale est particulièrement adaptée pour gérer les informations de $\mathcal{M}$OISE^{Inst} dont le caractère récursif est évident : un groupe est composé de rôles ou d'autres groupes, les missions sont composées de buts ou d'autres missions, etc. En outre, cela nous permet de gérer un seul objet dont on pourrait extraire les sous-graphes selon les besoins ce qui est très utile dans un contexte d'application distribuée.

Finalement, notre structure étant très proche d'un graphe « classique » cela nous facilitera la formalisation de problèmes décisionnels et d'optimisation.

A. Gestion des sommets

Nous devons pouvoir disposer d'un ensemble de sommets pouvant être soit atomiques soit composés d'autres éléments atomiques. Pour ce faire, nous mettons à profit la notion d'héritage de la programmation orientée objet afin de réaliser une abstraction de sommet via le concepte de polymorphisme [1] [Eckel 04].

La classe abstraite $RecursiveVertex$ est le type de base dont hérite à la fois les classes $SimpleVertex$ et $RecursiveGraph$. Dès lors, le polymorphisme permet de gérer une collection de sommets "$RecursiveVertex$" pouvant être soit des sommets dits atomiques ($SimpleVertex$) soit d'autres graphes récursifs ($RecursiveGraph$).

Tout sommet de l'ensemble V des sommets d'un graphe récursif se doit d'être unique, nous allons alors les représenter par un identifiant unique, ce qui nous permet de gérer V avec une table de hachage gérant les couples (ID, $RecursiveVertex$) ce qui optimise grandement l'accès à un sommet donné avec un temps d'accès moyen en $O(1)$ et $O(n)$ au pire des cas [Cormen 01].

Bien qu'un sommet soit unique au sein d'un même graphe, rien n'empêche qu'il soit utilisé par plusieurs graphes comme l'illustre la figure suivante :

Concrètement, si une spécification nous indique qu'il faut ajouter un sommet dont l'identifiant est « ID » nous commençons par rechercher ID dans le graphe récursif. Si nous le trouvons nous ajoutons la référence obtenue lors de la recherche, sinon nous créons vraiment l'objet. Dans l'exemple ci dessus, le sommet « a » du graphe racine est le seul a avoir été créé (instancié) les sommets « a » dans les deux sous-graphes y font seulement référence.

Cette façon de procéder garantit un stockage mémoire minimal et pour optimiser la recherche, nous proposons de gérer un catalogue des sommets, maintenu au niveau du graphe racine, et qui contient tous les sommets de tous les graphes.

1. Traiter l'objet non pas d'après son type spécifique mais d'après son type de base.

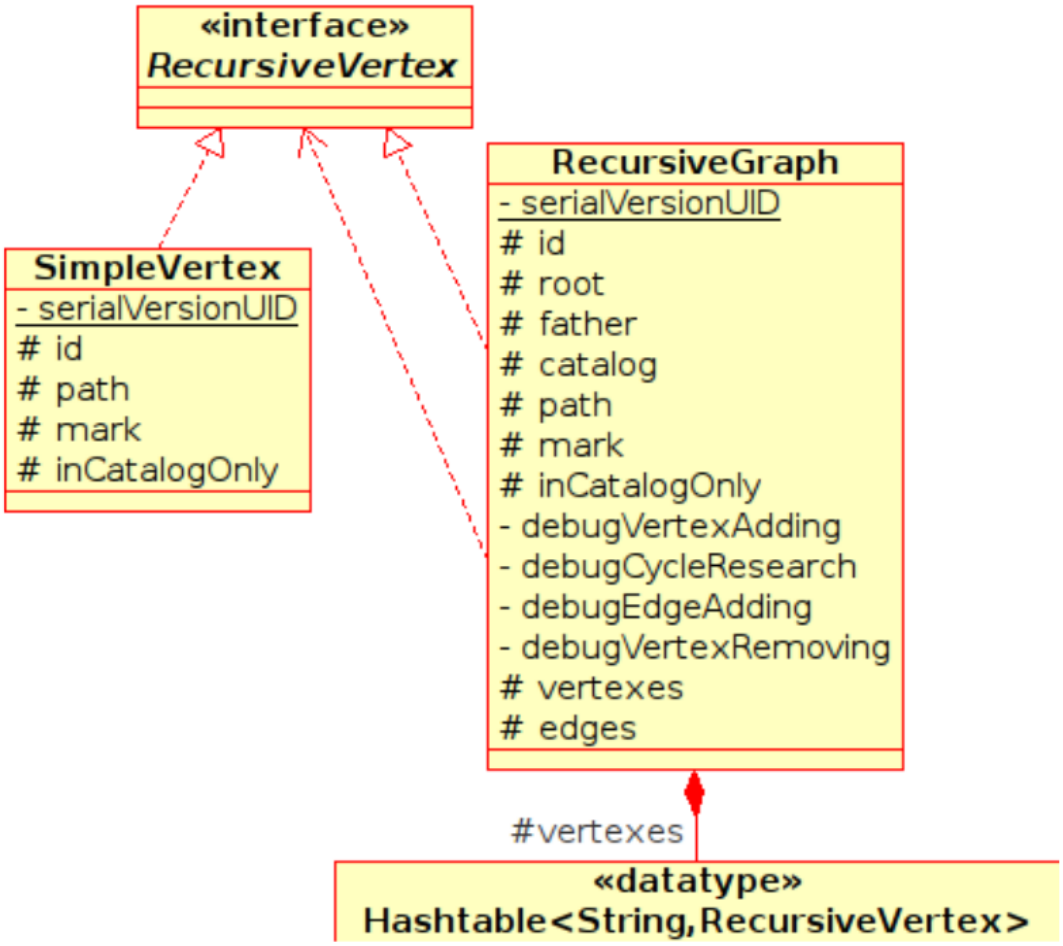


FIGURE B.1 – UML : Abstraction de sommet.

Réalisé avec une table de hachage, le catalogue assure donc une recherche en un temps moyen constant.

B. Gestion des arêtes

Pour un graphe récursif donné, nous avons plusieurs types d'arêtes ayant toutes une signification différente. La solution mathématique évidente est de gérer une collection de graphe à savoir un par type d'arête. Et pour palier au problème de redondance au niveau des sommets, nous gérons au sein de *RecursiveGraph* une table de hachage (ID, $NeighbourhoodSet$) qui au sommet identifié par ID associe un ensemble de voisinages c'est à dire l'ensemble de tous les voisinages de

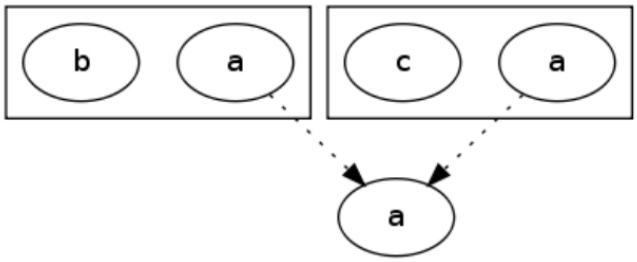


FIGURE B.2 – Unicité mémoire des sommets d'un graphe récursif.

ID pour tous les types d'arêtes.

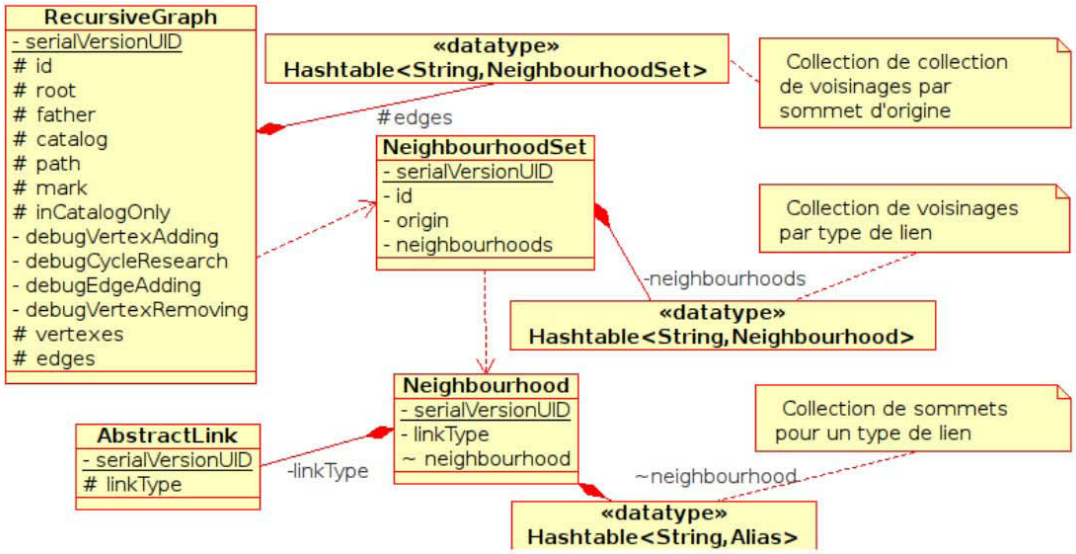


FIGURE B.3 – UML : Stockage des arêtes.

La classe *NeighbourhoodSet* est elle encore basée sur une table de hachage (*linkType*, *Neighbourhood*) où *linkType* est l'identifiant du type de lien et *Neighbourhood* est le voisinage de ID pour le lien « *linkType* » à savoir un vecteur de références vers des *RecursiveVertex* et un objet *AbstractLink* qui a pour vocation de pouvoir ajouter des informations à l'arête (des contraintes par exemple).

Pour l'exemple ci dessus notre structure de données stocke les arêtes de la manière suivante :

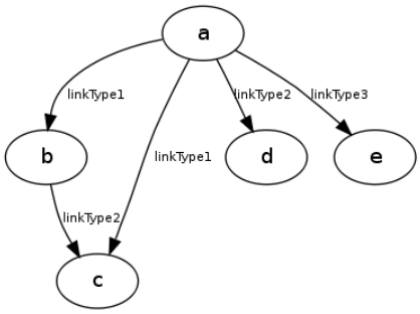


FIGURE B.4 – Différents types d'arêtes dans un même graphe.

$$E = \{(a, linkType1, \{b, c\}), (a, linkType2, \{d\}), (a, linkType3, \{e\}), (b, linkType2, \{c\})\}$$

Et nous permet de savoir rapidement que le sommet « a », pour le type de lien « $linkType1$ » a deux voisins « b » et « c ».

A ce niveau, il se pose un problème lié au fait qu'un sommet puisse se trouver dans plusieurs graphes récursifs, se pose. Pour une arête, la source est nécessairement un sommet de V mais la destination a pour seule contrainte de devoir exister. En d'autres termes la destination peut être un autre sommet « atomique » ou un autre graphe récursif, dans ce graphe comme dans un autre.

Nous devons avoir un moyen de faire référence à un sommet à un emplacement spécifique, c'est à dire à une occurrence particulière d'un sommet redondant. À cette fin, la classe $Neighbourhood$ ne doit pas être un ensemble de références vers des sommets mais une ensemble d'Alias à savoir de couples (Chemin, Sommet) afin de spécifier à quelle référence spécifique sur le sommet nous faisons allusion. Le chemin est donné d'une manière analogue aux systèmes de fichiers POSIX [2].

Nous proposons le comportement suivant lors de la création d'une arête. Si un chemin est spécifié nous faisons le lien vers cette occurrence, sinon, si la destination est un sommet du graphe nous faisons le lien vers cette occurrence spécifique (même s'il y en d'autres) et dans tous les autres cas, nous faisons le lien vers toutes les occurrences.

2. POSIX est le nom d'une famille de standards définie depuis 1988 par l'IEEE De ces standards ont émergé d'un projet de standardisation des API des logiciels destinés à fonctionner sur des variantes du système d'exploitation UNIX. C'est un acronyme de Portable Operating System Interface, dont le X exprime l'héritage UNIX de l'interface de programmation.

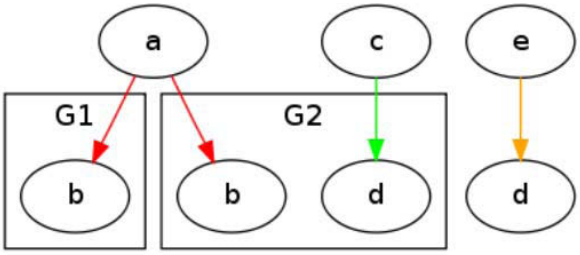


FIGURE B.5 – Gestion de la redondance lors de la création d'arêtes.

— En vert le cas (« c », « $/G2/d$ ») ou l'arête spécifique est créée.
— En orange le cas (« e », « d ») ou aucun chemin n'est donné mais ou d existe dans le même graphe que l'origine (ici la racine), seule cette arête est créée alors que d existe aussi dans $G2$.
— En rouge le cas (« a », « b ») ou des arêtes sont créées vers toutes les occurrences de b.

Pour retrouver facilement toutes les occurrences d'un sommet et se dispenser, une fois encore, d'un coûteux parcours récursif, $SimpleVertex$ et $RecursiveGraph$ maintiennent tous deux un vecteur de chemins donnant directement tous leurs emplacements. Le chemin d'un sommet est calculé dynamiquement à l'ajout dans son graphe père, en fonction de ses propres chemins. Tout cela nous mène au diagramme simplifié suivant :

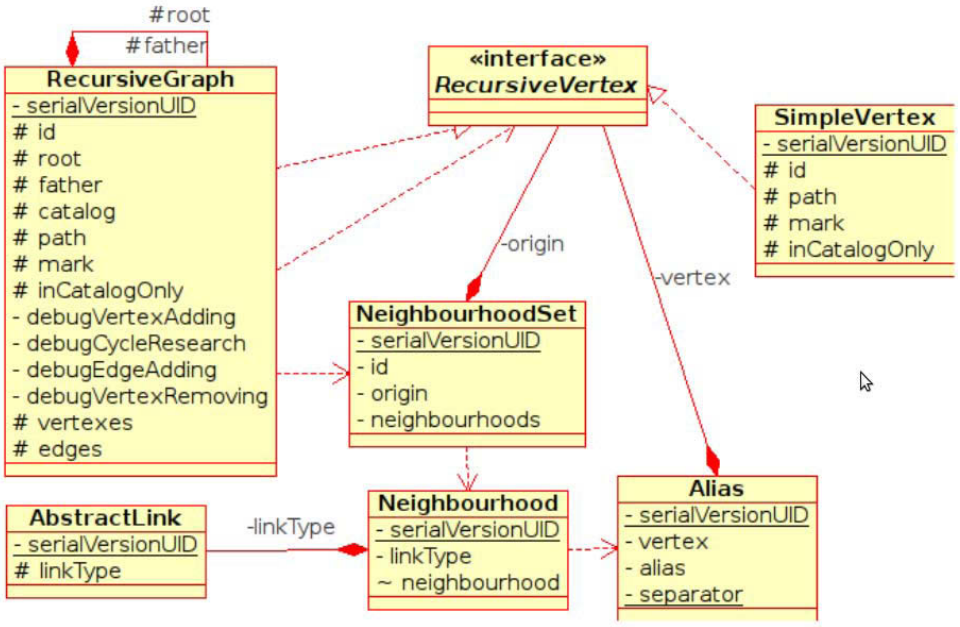


FIGURE B.6 – Vue d'ensemble d'un graphe récursif.

Annexe C

Illustration en simulation de l'aspect hétérogène de l'architecture de contrôle/commande proposée

L'architecture de contrôle proposée peut être aussi hétérogène, en d'autres termes, dans notre architecture nous pouvons combiner tous les aspects de contrôle précédemment cités en une seule exécution, cela permet par exemple de gérer à la fois plusieurs SMR évoluant dans le même environnement avec différentes missions et cibles.

Pour illustrer cela, nous avons choisi de simuler deux scénarios de fonctionnement :

— deux formations différentes dans le même environnement : deux SMR se composant chacun de trois robots, évoluant en deux formations différentes, pour atteindre deux cibles différentes.
— deux formations différentes et un robot seul dans le même environnement : deux SMR se composant chacun de trois robots, évoluant en deux formations différentes, plus un robot seul en mode planification supervisée, pour atteindre trois cibles différentes.

Deux formations différentes dans le même environnement

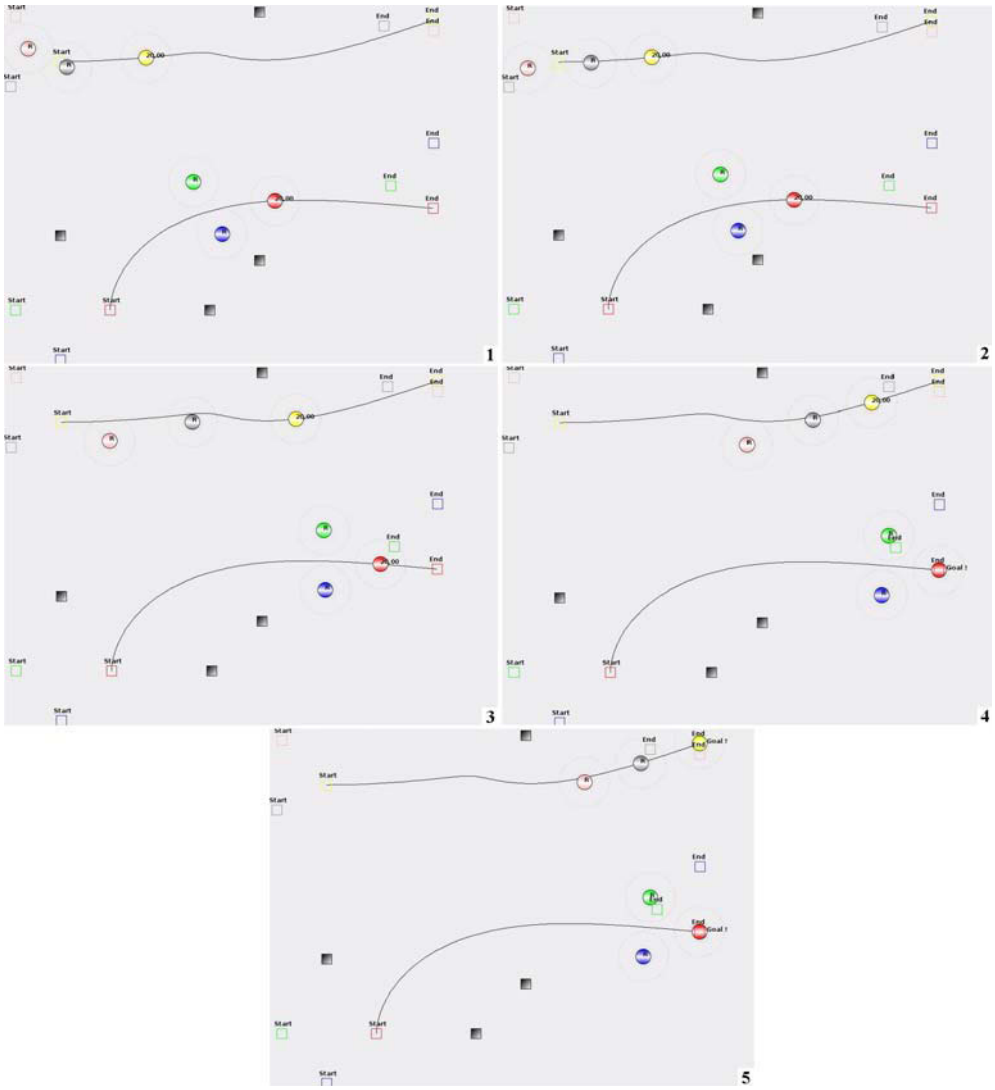


FIGURE C.1 – Simulation du scénario représentant deux formations différentes dans le même environnement.

Dans ce scénario nous disposons de deux groupes de robots, contenant chacun trois robots, évoluant en deux formations différentes **triangle** et **ligne**, et ayant deux destinations différentes. Le résultat obtenu est représenté sur la figure C.1.

Les deux leaders des deux formations sont en mode navigation centralisée,

et les suiveurs en mode navigation distribuée. On remarque bien que les leaders des deux systèmes multi-robots partent en même temps, les suiveurs respectifs se mettent en place suivant la formation assignée tout au long de leurs trajectoires respectives jusqu'à l'arrivée à destination. Cette simulation montre le degré d'hétérogénéité de l'architecture de contrôle proposée où on a pu exécuter en même temps et avec les mêmes codes sources, deux mêmes systèmes au même moment.

Cet algorithme permet aussi la gestion d'événements inattendus pour les deux systèmes multi-robots, ce qui représente une architecture ouverte permettant de gérer plusieurs SMR à la fois.

Deux formations différentes et un robot seul dans le même environnement

Dans ce scénario, nous avons ajouté la simulation d'un robot seul en mode planification supervisée, à la simulation du scénario précédent contenant uniquement deux SMR avec deux formations différentes. Les résultats obtenues sont représentés sur la figure C.2.

On remarque bien que trois systèmes se mettent en place facilement et suivent leurs chemins respectifs. Comme l'algorithme précédent on a aussi la possibilité de gérer les événements inattendus pour les trois systèmes robotiques.

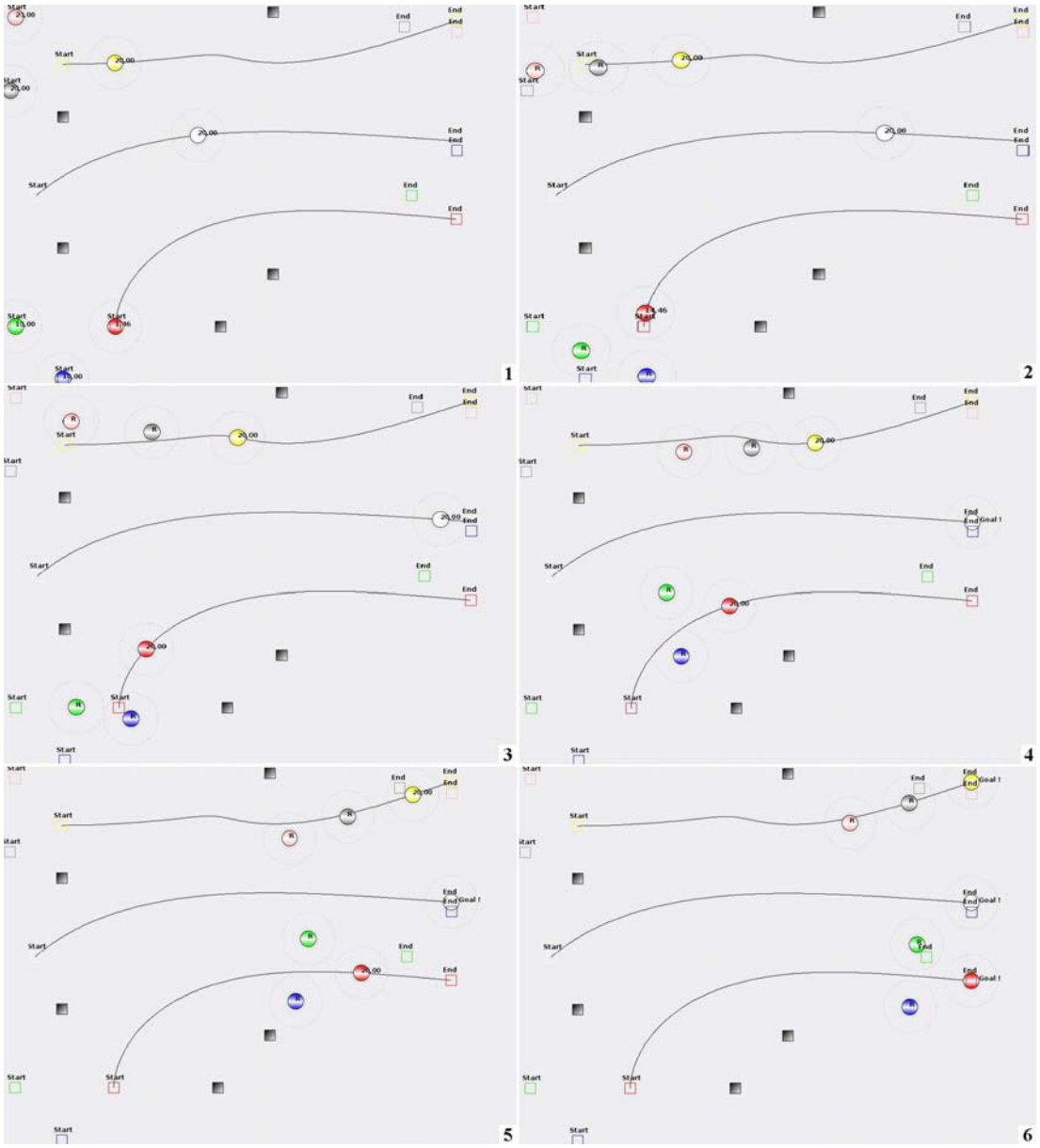


FIGURE C.2 – Simulation du scénario représentant deux formations différentes et un robot seul dans le même environnement.

Annexe D

Présentation des outils : JADE, les langages UML et XML

D.1 Présentation de JADE

JADE [Bellifemine 07] est un middleware facilitant la mise en place d'un système multi-agent, il permet de développer des agents assez simplement, en héritant d'une classe de base Agent qui permet d'obtenir des primitive de communication notamment. Les agents sont identifiés de manière unique par un AID (Agent ID), et un système de page jaunes permet de retrouver les $AIDs$ des agents en fonction des services qu'ils proposent.

Tous les agents rentrant dans la composition de $ROBOTOPIA$ utilisent ce mécanisme de page jaune. Dès lors ils peuvent tous contacter selon les besoins :

- Le Superviseur.
- Les managers : $RoleManager$, $GoalManager$ ou $ContextManager$.
- Tous les agents $ROBOTOPIAAgent$ de manière globale y compris les acteurs.
- Et de manière dynamique un acteur en particulier ou l'ensemble des acteurs jouant un même rôle.

Les messages échangés sont de type $ACLMessage$ dans lesquels on peut encapsuler des objets. Dans notre cas nous encapsulons des graphes récursifs, des objets $Command$ ou des $ActorMessage$ comme nous le verrons par la suite.

Les actions pouvant être réalisées par un agent sont définies avec des objets $Behaviour$, que l'on peut ajouter au fur et à mesure à l'agent si nous voulons qu'il les exécutent.

Plusieurs types de *Behaviours* particuliers existent, notamment les *OneShotBehaviour* (exécution unique) les *CyclicBehaviour* (exécution cyclique) et les *CompositeBehaviour* (*ParallelBehaviour* ou *SequentialBehaviour*) qui se composent d'autres sous-comportement pour obtenir des comportements plus complexes.

D.2 Présentation du langage UML

UML (Unified Modeling Language, que l'on peut traduire par langage de modélisation unifié) est une notation permettant de modéliser un problème de façon standard. Ce langage est né de la fusion de plusieurs méthodes existant auparavant, et est devenu désormais la référence en terme de modélisation objet.

UML n'est pas une méthode dans la mesure où elle ne présente aucune démarche. A ce titre UML est un formalisme de modélisation objet. Le mot méthode parfois utilisé par abus de langage dans ce qui suit ne doit donc pas être entendu comme une "démarche".

La modélisation objet consiste à créer une représentation informatique des éléments du monde réel auxquels on s'intéresse, sans se préoccuper de l'implémentation, ce qui signifie indépendamment d'un langage de programmation. Il s'agit donc de déterminer les objets présents et d'isoler leurs données et les fonctions qui les utilisent. Pour cela des méthodes ont été mises au point. Entre 1970 et 1990, de nombreux analystes ont mis au point des approches orientées objets, si bien qu'en 1994 il existait plus de 50 méthodes objet. Toutefois seules 3 méthodes ont véritablement émergé :

— la méthode OMT de Rumbaugh,
— la méthode BOOCH'93 de Booch,
— la méthode OOSE de Jacobson (Object Oriented Software Engineering).

A partir de 1994, Rumbaugh et Booch (rejoints en 1995 par Jacobson) ont unis leurs efforts pour mettre au point la méthode unifiée (unified method 0.8), incorporant les avantages de chacune des méthodes précédentes.

La méthode unifiée à partir de la version 1.0 devient UML (Unified Modeling Language), une notation universelle pour la modélisation objet. UML 1.0 est soumise à l'OMG (Object Management Group) en janvier 1997, mais elle ne sera acceptée qu'en novembre 1997 dans sa version 1.1, date à partir de laquelle UML devient un standard international. Les dernières versions des spécifications peuvent sont disponibles au téléchargement.

UML représente un moyen de spécifier, représenter et construire les composantes d'un système informatique. La figure D.1 décrit la façon avec laquelle un objet est décrit en UML.

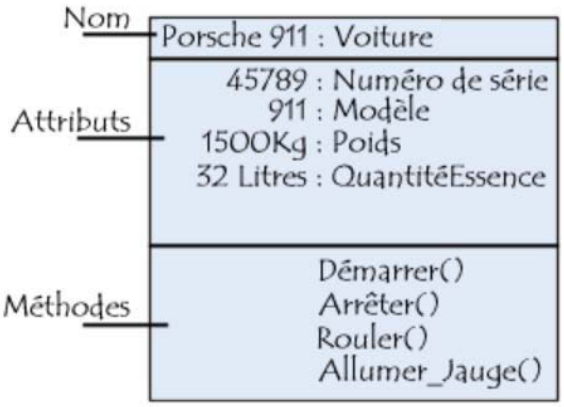


FIGURE D.1 – Représentation UML de l'objet : voiture.

A partir de 1997, UML est devenue une norme de l'OMG (Object Management Group), ce qui lui a permis de s'imposer en tant que méthode de développement objet et être reconnue et utilisée par de nombreuses entreprises.

D.3 Présentation du langage XML

XML (entendez eXtensible Markup Language et traduisez Langage à balises étendu, ou Langage à balises extensible) est en quelque sorte un langage HTML amélioré permettant de définir de nouvelles balises. Il s'agit effectivement d'un langage permettant de mettre en forme des documents grâce à des balises (markup).

La force de XML réside dans sa capacité à pouvoir décrire n'importe quel domaine de données grâce à son extensibilité. Il va permettre de structurer, poser le vocabulaire et la syntaxe des données qu'il va contenir.

Les balises XML décrivent le contenu plutôt que la présentation. Ainsi, XML permet de séparer le contenu de la présentation, etc. ce qui permet par exemple d'afficher un même document sur des applications ou des périphériques différents sans pour autant nécessiter de créer autant de versions du document que l'on nécessite de représentations !

XML fournit un moyen de vérifier la syntaxe d'un document grâce aux DTD (Document Type Definition). Il s'agit d'un fichier décrivant la structure des documents y faisant référence grâce à un langage adapté. Ainsi un document XML doit suivre scrupuleusement les conventions de notation XML et peut éventuellement faire référence à une DTD décrivant l'imbrication des éléments possibles.

XML permet donc de définir un format d'échange selon les besoins de l'utilisateur et offre des mécanismes pour vérifier la validité du document produit. Il est donc essentiel pour le receveur d'un document XML de pouvoir extraire les données du document. Cette opération est possible à l'aide d'un outil appelé analyseur (en anglais parser, parfois francisé en parseur).

Le parseur permet d'une part d'extraire les données d'un document XML (on parle d'analyse du document ou de parsing) ainsi que de vérifier éventuellement la validité du document.

Voici les principaux atouts de XML :

— la lisibilité : aucune connaissance ne doit théoriquement être nécessaire pour comprendre un contenu d'un document XML.
— autodescriptif et extensible.
— une structure arborescente : permettant de modéliser la majorité des problèmes informatiques.
— universalité et portabilité : les différents jeux de caractères sont pris en compte.
— déployable : il peut être facilement distribué par n'importe quels protocoles à même de transporter du texte, comme HTTP.
— intégrabilité : un document XML est utilisable par toute application pourvue d'un parser (c'est-à-dire un logiciel permettant d'analyser un code XML).
— extensibilité : un document XML doit pouvoir être utilisable dans tous les domaines d'applications.

Ainsi, XML est particulièrement adapté à l'échange de données et de documents.

Bibliographie

Bibliographie

[Adams 99] M. Adams. *High Speed Target Pursuit and Asymptotic stability in Mobile Robotics*. IEEE Transactions on Robotics and Automation, vol. 15, pp. 230 – 237, 1999.

[Adouane 05] L. Adouane. *Architectures de Contrôle Comportementales et Réactives pour la Coopération d'un Groupe de Robots Mobiles Architecture de contrôle hybride pour systèmes multi-robots mobiles*. Thèse de doctorat, Laboratoire d'Automatique de Besançon (UMR CNRS 5696), 2005.

[Adouane 09] L. Adouane. *Orbital Obstacle Avoidance Algorithm for Reliable and On-Line Mobile Robot Navigation*. 9th Conference on Autonomous Robot Systems and Competitions, May 2009.

[Agheli 13] M. Agheli. *Analytical Workspace, Kinematics, and Foot Force Based Stability of Hexapod Walking Robots*. Thèse de doctorat, Worcester Polytechnic Institute, 2013.

[Agmon 08] N. Agmon, V. Sadov, G. A. Kaminka & S. Kraus. *The impact of adversarial planning in perimeter patrol*. the Seventh International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-08), vol. 1, pp. 55–62, 2008.

[Agmon 11] N. Agmon, D. Urieli & P. Stone. *Multiagent patrol generalized to complex environmental conditions*. 2011.

[Alami 98a] R. Alami, R. Chatila, S. Fleury, M. Ghallab & F. Ingrand. *An architecture for autonomy*. International Journal of Robotics Research, 1998.

[Alami 98b] R. Alami, S. Fleury, M. Herrb, F. Ingrand & F. Robert. *Multi-Robot Cooperation in the MARTHA Project*. IEEE Robotics and Automation Magazine, vol. 5, pp. 36–47, 1998.

[Albus 97] J.S. Albus, R. Lumia, J. Fiala & A. Wavening. Nasrem. *The nasa/nbs standard reference model for telerobot control system architecture*. 1997.

[Albus 02] J.S. Albus. *4d/rdc : A reference model architecture for unmanned vehicle systems*. 2002.

[Ardema 91] M.D. Ardema & J.M. Skowronski. *Dynamic game applied to coordination control of two arm robotic system*. Differential Games - Developments in Modelling and Computation, pp. 118–130, 1991.

[Arkin 98] R. C. Arkin. *Behavior-Based Robotics*. The MIT Press, Cambridge, 1998.

[Asama 95] H. Asama, Sato M., Bogoni L., Kaetsu H., Matsumoto A. & Endo L. *Development of an omni-direetional mobile robot with 3 dof decoupling drive mechanism*. IEEE International Conference on Robotics and Automation, pp. 1925–1930, 1995.

[Bahr 09] A. Bahr. *Cooperative Localization for Autonomous Underwater Vehicles*. Thèse de doctorat, Massachusetts Institute of Technology/Woods Hole Oceanographic Institution, 2009.

[Balch 98] T. Balch & R.C. Arkin. *Behavior-based formation control for multirobot teams*. IEEE Transactions on Robotics and Automation, vol. 4(6), pp. 926–939, 1998.

[Balch 99] T. Balch & R.C. Arkin. *Behavior-based Formation Control for Multi-robot Teams*. IEEE Transactions on Robotics and Autmation, 1999.

[Barfoot 04] T. Barfoot & C. Clark. *Motion planning for formations of mobile robots*. Robotics and Autonomous Systems, vol. 46(2), pp. 65–78, 2004.

[Barraquand 92] J. Barraquand, B. Langois & J.C. Latombe. *Numerical potential field techniques for robot path planning*. IEEE Transactions on Robotics and Automation, Man and Cybernetics, vol. 22, pp. 224, 1992.

[Basilico 09] N. Basilico, N. Gatti & F. Amigoni. *Leader-follower strategies for robotic patrolling in environments with arbitrary topologies*. AAMAS (1), pp. 57–64, 2009.

[Beard 01] R.W. Beard, J. Lawton & F.Y. Hadaegh. *A Coordination Architecture for Spacecraft Formation Control*. IEEE Transactions on Control Systems Technology, vol. 9, pp. 777–790, 2001.

[Beigel 89] R. Beigel & W.I. Gasarch. *On the complexity of finding the chromatic number of a recursive graph*. Pure and Appl. Logic, vol. 45, pp. 227–247, 1989.

[Bellifemine 07] F. Bellifemine, G. Caire & D. Greenwood. *Developing multi-agent systems with JADE*. Wiley Series in Agent Technology, 2007.

[Bennewitz 02] M. Bennewitz, W. Burgard & S. Thrun. *Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots*. Robotics and Autonomous Systems, vol. 41, pp. 89–99, 2002.

[Benzerrouk 10] A. Benzerrouk, L. Adouane, L. Lequievre & P. Martinet. *Navigation of Multi-Robot Formation in Unstructured Environment Using Dynamical Virtual Structures*. IEEE/RSJ International Conference on Intelligent Robots and Systems, 2010.

[Benzerrouk 11] A. Benzerrouk. *Architecture de contrôle hybride pour systèmes multi-robots mobiles*. Thèse de doctorat, LAboratoire des Sciences et Matériaux pour l'Electronique, et d'Automatique, 2011.

[Berge 87] C. Berge. *Hypergraphes. Combinatoires des ensembles finis*. Gauthier-Villars, 1987.

[Bloch 96] A.M. Bloch & S.V. Drakunov. *Stabilization and tracking in the nonholonomic integrator via sliding modes*. Systems and Control Letters, vol. 29, pp. 91–99, 1996.

[Bom 05] J. Bom, B. Thuilot, F. Marmoiton & P. Martinet. *Nonlinear control for urban vehicles platooning, relying upon a unique kinematic GPS*. pp. 4149–4154, 2005.

[Boutilier 96] Craig Boutilier. *Planning, learning and coordination in multiagent decision processes*. In Proceedings of the 6th conference on Theoretical aspects of rationality and knowledge, pp. 195–210, 1996.

[Brooks 85] R.A. Brooks. *A robust layered control system for a mobile robot*. IEEE Journal of Robotics and Automation, vol. 2, pp. 14–23, 1985.

[Brooks 86] R. A. Brooks. *A robust layered control system for a mobile robot*. IEEE Journal of Robotics and Automation, RA-2(1), pp. 14–23, 1986.

[Brooks 89] R. A. Brooks & A. M. Flynn. *Fast, cheap and out of control : A robot invasion of the solar system*. pp. 478–485, 1989.

[Brooks 90a] R. Brooks, P. Maes, M. Mataric & G. More. *Lunar base construction robots*. Towards a New Frontier of Applications, Proceedings. IROS '90. IEEE International Workshop, pp. 389–392, 1990.

[Brooks 90b] Rodnes A. Brooks. *A robust layered control system for a mobile robot*. Artificial intelligence at MIT : expanding frontiers, pp. 2–27, 1990.

[Burgard 05] W. Burgard, M. Moors, C. Stachniss & F. Schneider. *Coordinated multirobot exploration*. IEEE Transactions on Robotics, vol. 21, no. 3, pp. 376–386, 2005.

[Caloud 90] P. Caloud, W. Choi, J.-C. Latombe, C. Le Pape & M. Yim. *Indoor automation with many mobile robots*. pp. 67–72, 1990.

[Camacho 06] D. Camacho, F. Fernandez & M. Rodelgo. *Roboskeleton : An architecture for coordinating robot soccer agents*. Engineering Applications of Artificial Intelligence, vol. 19, no. 2, pp. 179–188, 2006.

[Caprari 00] Gilles Caprari, Kai Oliver Arras & Roland Siegwart. *The autonomous miniature robot Alice : from prototypes to applications*. IROS, pp. 793–798, 2000.

[Chen 03] X. Chen, A. Serrani & H. Ozbay. *Control of leader-follower formations of terrestrial uav's*. 42nd IEEE Decision and Control Conference, vol. 1, pp. 498–503, 2003.

[Chiem 04] S. Chiem & E. Cervera. *Vision-based robot formations with Bézier trajectories*. Proceedings of Intelligent and Autonomous Robots, pp. 191–198, 2004.

[CogniTeam 09] CogniTeam. *CogniTAO (Think As One) software developement kit and run-time kernel*. Ltd, 2009.

[Cormen 01] T.H. Cormen, C.E. Leiserson, R.L. Rivest & C. Stein. *Introduction à l'algorithmique*. Second Edition, Dunod, 2001.

[Cruz 78] J. Cruz. *Leader-follower strategies for multilevel systems*. IEE Transactions on Automatic Control, vol. 23, pp. 244–255, 1978.

[Cruz 07] D. Cruz, J. Mcclintock, B. Perteet, O. Orqueda, Y. Cao & R. Fierro. *Decentralized cooperative control - a multivehicle platform for research in networked embedded systems*. IEE Control Systems Magazine, vol. 27, pp. 58–78, 2007.

[Das 02] A. K. Das, R. Fierro, V. Kumar, J. P. Ostrowski, J. Spletzer & C. J. Taylor. *A vision-based formation control framework*. IEEE Transactions on Robotics and Automation, vol. 18, pp. 813–825, 2002.

[Davids 02] A. Davids. *Urban search and rescue robots : from tragedy to technology*. pp. 81–83, 2002.

[de Wit 91] C. Canudas de Wit & J.E. Slotine. *Sliding observers in robot manipulators.* Automatica, vol. 27(5), pp. 859–864, 1991.

[Defoort 05] M. Defoort, T. Floquet, A. Kokosy & W. Perruquetti. *Tracking of a unicycle mobile robot using integral sliding mode control.* International Conference on Informatics in Control, Automation and Robotics (ICINCO), 2005.

[Defoort 07] M. Defoort, J. Palos, A. Kokosy, T. Floquet & W. Perruquetti. *Experimental motion planning and control for an autonomous non holonomic mobile robot.* IEEE International Conference on Robotics and Automation, 2007.

[Durfee. 01] E. H. Durfee. *Scaling up agent coordination strategies.* Computer, 34(7), pp. 39–46, 2001.

[Eckel 04] B. Eckel. *Penser en Java.* http ://penserenjava.free.fr, 2004.

[Edwards 04] D. Edwards, T. Bean, D. Odell & M. Anderson. *A leader-follower algorithm for multiple auv formations.* IEEE/OES in Autonomous Underwater Vehicles, pp. 40–46, 2004.

[Elmaliach 08] Y. Elmaliach & G. A. Kaminka. *Robust multi-robot formations under human supervision and control.* Journal of Physical Agents, vol. 2, no. 1, pp. 31–52, 2008.

[Erdmann 87] M. Erdmann & T. Lozano-Perez. *On multiple moving objects.* Algorithmica, pp. 477–521, 1987.

[Farinelli 04] A. Farinelli, L. Iocchi & D. Nardi. *Multirobot systems : a classification focused on coordination.* pp. 2015–2028, 2004.

[Fave 09] D. Fave, S. Canu, L. Iocchi, D. Nardi & V. Ziparo. *Multi-objective multi-robot surveillance.* ICRA'09, pp. 68–73, 2009.

[Feddema 02] J. T. Feddema, C. Lewis & D. A. Schoenwald. *Decentralized control of cooperative robotic vehicles : theory and application.* IEE Transactions on Robotics and Automation, vol. 18, pp. 852–864, 2002.

[Fei 92] Z. Fei & X. Liu. *f-COCOMO : fuzzy constructive cost model in software engineering.* IEEE International Conference on Fuzzy Systems, 1992., pp. 331–337, 1992.

[Ferber 95] J. Ferber. *Les systèmes multi-agents, vers une intelligence collective.* Inter Editions, 1995.

[Ferber 98] J. Ferber & O. Gutknecht. *A meta-model for the analysis and design of organizations in multi-agent systems.* The Third International Conference on Multi-Agent Systems (ICMAS98), pp. 128–135, 1998.

[Ferber 03] J. Ferber, O. Gutknecht & F. Michel. *From agents to organizations : An organizational view of multi-agent systems.* In Paolo Giorgini, Jorg P. Muller, and James Odell, editors, AOSE, pp. 214–230, 2003.

[Ferrari 98] C. Ferrari, E. Pagello, J. Ota & T. Arai. *Multirobot motion coordination in space and time.* Robotics and Autonomous Systems, vol. 25, pp. 219–229, 1998.

[Fierro 01] R. Fierro, A.K. Das, V. Kumar & J.P. Ostrowski. *Hybrid control of formations of robots.* Proceedings of IEEE International Conference on Robotics and Automation, vol. 1, pp. 157–162, 2001.

[Fischer 05] F. Fischer, M. Rovatsos & G. Weiss. *Acquiring and adapting probabilistic models of agent conversation.* Proceedings of the 4th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), pp. 106–113, 2005.

[Fitoussi 00] D. Fitoussi & M. Tennenholtz. *Choosing social laws for multi-agent systems : Minimality and simplicity.* Artificial Intelligence, 119(1-2), pp. 61–101, 2000.

[Fliess 95] M. Fliess, J. Levine, P. Martin & P. Rouchon. *Flatness and defect of nonlinear systems : introductory theory and examples.* International Journal of Control, vol. 61(6), pp. 1327–1361, 1995.

[Floquet 03] T. Floquet, J-P. Barbot & W. Perruquetti. *Higher-order sliding mode stabilization for a class of nonholonomic perturbed systems.* Automatica, vol. 39(6), pp. 1077–1083, 2003.

[Fox 97] D. Fox, W. Burgard & S. Thrun. *The Dynamic Window Approach to Collision Avoidance*, 1997.

[Fredslund 02] J. Fredslund & M.J. Mataric. *A general algorithm for robot formations using local sensing and minimal communication.* IEEE Transactions on Robotics and Automation, vol. 18, pp. 837–846, 2002.

[Fukuda 89] T. Fukuda, S. Nakagawa, Y. Kawauchi & M. Buss. *Structure decision method for self organising robots based on cell structures-cebot.* pp. 695–700, 1989.

[Gat 98] E. Gat & D. Korten Kamp. *On three-layer architecture.* 1998.

[Gateau 04] B. Gateau. *A model for Organizational Interaction Based on Agents, Founded in Logic.* Thèse de doctorat, PhD thesis, Universiteit Utrecht, 2004.

[Gateau 07] B. Gateau. *Modélisation et Supervision d'Institutions Milti-Agents.* Thèse de doctorat, In collaboration between CRP Henri Tudor and Ecole des Mines de Saint-Etienne, 2007.

[Ge 02] S.S. Ge & Y.J. Cui. *Dynamic motion planning for mobile robots using potential field method.* Autonomous Robots, vol. 13(3), pp. 207–222, 2002.

[Gilles 11] Mourioux Gilles, Novales Cyril & Josserand Laurence. *Framework for modular robot control architecture.* ICRA 2011 Workshop on Robotics Modular Architecture Design and Standardization, 2011.

[Guemkam 13] G. Guemkam, D. Khadraoui, B. Gâteau & Z. Guessoum. *ARMAN : Agent-based Reputation for Mobile Ad hoc Networks.* PAAMAS, pp. 122–132, 2013.

[Harel 98] D. Harel. *Towards a theory of recursive structures.* 23rd International Symposium on Mathematical Foundations of Computer Science, vol. 1450 of LNCS, pp. 36–53, 1998.

[Healey 96] A. J. Healey, D. B. Marco & R. B. McGhee. *Autonomous underwater vehicles Hybrid control of mission and motion.* Autonomous Robots 3, pp. 169–186, 1996.

[Higashimori 11] Laboratory Kaneko Higashimori. *Omni-ball.* `http://www.pobot.org/Kaneko-Higashimori-Laboratory.html`, 2011.

[Hopcroft 84] J. Hopcroft, J.T. Schwartz & M. Sharir. *Complexity of motion planning for multiple independent objects-PSPACE-hardness of the warehouseman's problem.* International Journal of Robotics Research, vol. 4, pp. 76–88, 1984.

[Huang 08] L. Huang. *Velocity planning for a mobile robot to track a moving target - a potential field approach.* Robotics and Autonomous Systems, vol. 57(1), pp. 55–63, 2008.

[Huntsberger 03] T. Huntsberger, P. Pirjanian, A. Trebi-ollennu, H.A. Nayar, A.J. Ganino, M. Garrett, S.S. Joshi & S.P. Schenker. *CAMPOUT : A Control Architecture for Tightly Coupled Coordination of Multi-Robot Systems for Planetary Surface Exploration.* IEEE Trans. Systems, Man and Cybernetics, Part A : Systems and Humans, vol. 33, pp. 550–559, 2003.

[Hübner 02] J.F. Hübner, J.S. Sichman & O. Boissier. *A model for the structural, functional, and deontic specification of organizations in multi-agent systems.* The 16th Brazilian Symposium on Artificial Intelligence (SBIA'02), no. 2507, pp. 118–128, 2002.

[Hübner 03] J.F. Hübner. *Um Modelo de Reorganização de Sistemas Multiagentes*. Thèse de doctorat, Universidade de São Paulo, 2003.

[Jalaoui 05] A. El Jalaoui, D. Andreu & B Jouvencel. . *A Control Architecture for Contextual Tasks Management : Application to the AUV Taipan*. . IEEE-OES Oceans'05 Europe Conference, 2005.

[Jarras 02] I. Jarras & B. Chaib-Draa. *Aperçu sur les systèmes multiagents*. Rapport technique, CIRANO, 2002.

[Jensen 11] E. Jensen, M. Franklin, S. Lahr & M. L. Gini. *Sustainable multi-robot patrol of an open polyline*. ICRA, pp. 4792–4797, 2011.

[Jeong 99] I. Jeong & J. Lee. *Evolving Fuzzy Logic Controllers for Multiple Mobile Robots Solving a Continuous Pursuit Problem*. IEEE International Conference on Fuzzy Systems, pp. 685–690, 1999.

[Jones 01] H. L. Jones & M. Snyder. *Supervisory Control of Multiple Robots Based on Real-Time Strategy Game Interaction Paradigm*. International Conference on Systems, Man and Cybernetics, pp. 383–388, 2001.

[K-Team 03] Khepera K-Team. *Khepera mobile robot*. `http://en.wikipedia.org/wiki/Khepera_mobile_robot`, 2003.

[Kaminka 08] G. A. Kaminka, R. G. Schechter & V. Sadov. *Using sensor morphology for multi-robot formations*. IEEE Transactions on Robotics, pp. 271–282, 2008.

[Kaminka 10] G. A. Kaminka, D. Erusalimchik & S. Kraus. *Adaptive multi-robot coordination : A game-theoretic perspective*. ICRA, pp. 328–334, 2010.

[Kanayama 91] Y. Kanayama, Y. Kimura, F. Miyazaki & T. Noguchi. *A stable tracking control method for a non-holonomic mobile robot*. IEEE/RSJ International Workshop on Intelligent Robots and systems IROS, pp. 1236–1241, 1991.

[Kazakov 04] D. Kazakov & M. Bartlett. *Cooperative navigation and the faculty of language*. Applied Artificial Intelligence, 18(9-10), pp. 885–901, 2004.

[Khatib 86] O. Khatib. *Real time obstacle avoidance for manipulators ans mobile robots*. International Journal of Robotics Research, vol. 5, pp. 90–99, 1986.

[Khennouf 95] H. Khennouf & C. Canudas de Wit. *On the construction of stabilizing discontinuous controllers for nonholonomic systems.* IFAC Nonlinear Control Systems Design Symposium, 1995.

[Khoshnevis 98] B. Khoshnevis & G. Bekey. *Centralized sensing and control of multiple mobile robots.* Computers & Industrial Engineering, vol. 35, pp. 503–506, 1998.

[Kim 03] D. Kim & J. Kim. *A real-time limit-cycle navigation method for fast mobile robots and its application to robot soccer.* Robotics and Autonomous Systems, vol. 42, pp. 17–30, 2003.

[Kim 04] J.-H. Kim, D.-H. Kim, Y.-J. Kim & K. T. Seow. *Soccer Robotics.* Springer Verlag, 2004.

[Ko 98] N. Y. Ko, R. Simmons, R. Ko & G. Simmons. *The Lane-Curvature Method for Local Obstacle Avoidance.* IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 3, pp. 1615–1621, 1998.

[Koren 91] Y. Koren & J. Borenstein. *Potential field methods and their inherent limitations for mobile robot navigation.* Confrence on Robotics and Automation, 1991.

[Lechevin 06] N. Lechevin, C. Rabbath & P. Sicard. *Trajectory tracking of leader-follower formations characterized by constant line-of-sight angles.* Automatica, vol. 42, pp. 2131–2141, 2006.

[Lee 00] S-O. Lee, Y-J. Cho, M. Hwang-Bo, B-J. You & S-R. Oh. *A Stable Target-Tracking Control for Unicycle Mobile Robot.* IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1822–1827, 2000.

[Leonard 01] N. Leonard & E. Fiorelli. *Virtual leaders, artificial potentials and coordinated control of groups.* 40th IEEE Conference on Decision and Control, vol. 3, pp. 2968–2973, 2001.

[Lewis 97] M.A. Lewis & K-H. Tan. *High precision formation control of mobile robots using virtual structures.* Autonomous Robots, vol. 4(4), pp. 387–403, 1997.

[Maalouf 06] E. Maalouf, M. Saad & H. Saliah. *A higher level path tracking controller for a four-wheel differentially steered mobile robot.* Journal of Robotics and Autonomous Systems, vol. 54, pp. 23–33, 2006.

[Marc 02] E. Marc, D.L.C. David & S. Carles. *Islander : an electronic institutions editor.* 1rst international joint conference on autonomous agents and multi-agent systems (AAMAS), vol. 3, pp. 1045–1052, 2002.

[Mataric 92] M. Mataric. *Minimizing complexity in controlling a mobile robot population.* pp. 830–835, 1992.

[Mataric 95] M.J. Mataric. *Issues and Approaches in Design of Collective Autonomous Agents.* Robotics and Autonomous Systems, vol. 16, pp. 321–331, 1995.

[Maxwell 13] P. Maxwell, J. Rykowski & G. Hurlock. *Proposal for the initiation of general and military specific benchmarking of robotic convoys.* 2013 IEEE International Conference on Technologies for Practical Robot Applications (TePRA), pp. 1–6, 2013.

[Michael 08] N. Michael, M. M. Zavlanos, V. Kumar & G. J. Pappas. *Distributed multi-robot task assignment and formation control.* ICRA, pp. 128–133, 2008.

[Michaud 02] F. Michaud, D. Letourneau, M. Guilbert & J. Valin. *Dynamic robot formations using directional visual perception.* IEEE/RSJ International Conference on Intelligent Robots and System, vol. 3, pp. 2740–2745, 2002.

[Minguez 00] J. Minguez & L. Montano. *Nearness diagram navigation (nd) : A new real time collision avoidance approach.* RSJ Int. Conference on Intelligent Robots and Systems, 2000.

[Morette 11] N. Morette, L. Josserand & P. Vieyres. *Direct Model Navigation issue shifted in the continuous domain by a predictive control approach for mobile robots.* ICRA'11, pp. 2566–2573, 2011.

[Morin 01] P. Morin & C. Samson. *A characterization of the Lie algebra rank condition by transverse periodic function.* SIAM Journal on Control and Optimization, vol. 40(4), pp. 1227–1249, 2001.

[Morin 03] P. Morin & C. Samson. *Practical stabilization of driftless systems on Lie groups : the transverse function approach.* IEEE Transactions on Automatic Control, vol. 49(9), pp. 1496–1508, 2003.

[Murphy 08] R. Murphy, Tadokoro S., Nardi D., Jacoff A., Fiorini P., Choset H. & Erkmen A. M. *Search and Rescue Robotics.* Springer Handbook of Robotics, pp. 1151–1173. 2008.

[NASA 12] NASA. *Sojourner et Opportunity.* `www.marsrovers.jpl.nasa.gov`, 2012.

[Noreils 90] F. Noreils. *Integrating multirobot coordination in a mobile-robot control system.* pp. 43–49, 1990.

[Noreils 93] F. R. Noreils. *Toward a robot architecture integrating cooperation between mobile robots : application to indoor environment.* International Journal of Robotics Research, vol. 12, pp. 79–98, 1993.

[Ogren 05] P. Ogren & N. E. Leonard. *A convergent Dynamic Window Approach to obstacle avoidance.* IEEE Transactions on Robotics, vol. 21, pp. 188–195, 2005.

[Ortiz 11] M. Ortiz, J. Pereda, J. Lope & F. Paz. *Inspection method based on multi-agent auction for graph-like maps.* 2011 Third World Congress on Nature and Biologically Inspired Computing (NaBIC), pp. 317–322, 2011.

[Ozdemir 07] V. Ozdemir & H. Temeltas. *Decentralized formation control using artificial potentials and virtual leaders.* 6th IFAC symposium on Intelligent Autonomous Vehicles, 2007.

[Py 10] F. Py, K. Rajan & C. McGann. *A systematic agent framework for situated autonomous systems.* AAMAS, pp. 583–590, 2010.

[Ranganathan 03] A. Ranganathan & S. Koenig. *A Reactive Robot Architecture with Planning On Demand.* IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1462–1468, 2003.

[Ren 08] W. Ren & Beard R. W. *Distributed consensus in multi-vehicle cooperative control, theory and application.* springer, 2008.

[Robots 13] Planète Robots. *Les robots sont dans le pré.* `http://www.planeterobots.com/2013/10/29/les-robots-sont-dans-le-pre/`, 2013.

[Rooker 07] M. N. Rooker & A. Birk. *Multi-robot exploration under the constraints of wireless networking.* Control Engineering Practice, vol. 15, no. 4, pp. 435–445, 2007.

[Rosenblatt 95] J. K. Rosenblatt. *DAMN : A distributed architecture for mobile navigation.* In Proc. Of the AAAI Spring Symp. on Lessons Learned from Implememted Software Architectures for Physical Agents, Stanford, CA, 1995.

[Rosenfeld 08] A. Rosenfeld, G. A. Kaminka, S. Kraus & O. Shehory. *A study of mechanisms for improving robotic group performance.* Artificial Intelligence, vol. 172, no. 6-7, pp. 633–655, 2008.

[Sardiña 11] Sebastian Sardiña & Lin Padgham. *A BDI agent programming language with failure handling, declarative goals, and planning.* Autonomous Agents and Multi-Agent Systems, vol. 23, no. 1, pp. 18–70, 2011.

[Saska 11] M. Saska, Vonásek V., Kulich V., Fišser D. & Krajník T. *Bringing Reality to Evolution of Modular Robots : Bio-Inspired Techniques for Building a Simulation Environment in the SYMBRION Project. In "Reconfigurable Modular Robotics : Challenges of Mechatronic and Bio-Chemo-Hybrid Systems".* Piscataway : IEEE, pp. 1–6, 2011.

[Schmitt 10] P. Schmitt, C. Bonhomme & B. Gateau. *Easy programming of Agent based Electronic Institution with UTOPIA.* NOuvelles TEchnologies de la REpartition (NOTERE), pp. 211–217, 2010.

[Schulz 01] D. Schulz, W. Burgard, D. Fox & A. B. Cremers. *Tracking Multiple Moving Objects with a Mobile Robot.* IEEE Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 371–377, 2001.

[Seiler 01] P. J. Seiler. *Coordinated control of unmanned aerial vehicles.* Ph.D. dissertation, University of California, 2001.

[Simmons 96] R. Simmons. *The Curvature-Velocity Method for Local Obstacle Avoidance.* IEEE International Conference on Robotics and Automation, pp. 3375–3382, 1996.

[Slotine 83] J. Slotine & S. Sastry. *Tracking control of nonlinear system using sliding mode surface, with application to robotics manipulators.* International Journal of Control, vol. 38, pp. 465–492, 1983.

[Smith 11] Stephen L. Smith, Mac Schwager & Daniela Rus. *Persistent monitoring of changing environments using a robot with limited range sensing.* ICRA, pp. 5448–5455, 2011.

[Smithers 94] T. Smithers, D. Corne & P. Ross. *On computing exploration and solving design problems.* Formal Design Methods for CAD 1994, pp. 293–313, 1994.

[SRI 67] SRI. *Shakey.* `www.sri.com/work/timeline/shakey-robot`, 1967.

[Stilwell 05] D. J. Stilwell, B. E. Bishop & C. A. Sylvester. *Redundant manipulator techniques for partially decentralized path planning and control of a platoon of autonomous vehicles.* IEEE Transactions on Systems, Man, and Cybernetics, vol. 35, pp. 842–848, 2005.

[Stonier 98] R. Stonier & M. Mohammadian. *Knowledge Acquisition for Target Capture.* IEEE International Conference on Evolutionary Computation, pp. 721–726, 1998.

[Stuart 96] A. Stuart & A. Humphries. *Dynamical Systems and Numerical Analysis*. Cambridge University Press, 1996.

[Svestka 98] P. Svestka & M.H. Overmars. *Coordinated path planning for multiple robots*. Robotics and Autonomous Systems, vol. 23, pp. 125–152, 1998.

[Systems 13] Kiva Systems. *Overview*. `http://en.wikipedia.org/wiki/Kiva_Systems`, 2013.

[Tang 07] Fang Tang & Lynne E. Parker. *A Complete Methodology for Generating Multi-Robot Task Solutions using ASyMTRe-D and Market-Based Task Allocation*. ICRA, pp. 3351–3358, 2007.

[Tate 94] A. Tate, B. Brabble & R. Kirby. O-plan2 : an open architecture for command, planning and control. Technical report, Artificial Intelligence Application Institute, University of Edimburg, 1994.

[Toledo 05] C. B. E. Toledo & N. R. Jennings. *Using reinforcement learning to coordinate better*. Computational Intelligence, 21(3), pp. 217–245, 2005.

[Tomsguide 12] Tomsguide. *Une moto à roues sphériques*. `http://www.tomsguide.fr/actualite/transport-moto,18297.html`, 2012.

[Tournassoud 86] P. Tournassoud. *A strategy for obstacle avoidance and its application to multi- robot systems*. IEEE International Conference on Robotics and Automation, pp. 1224–1229, 1986.

[Tousignant 11] S. Tousignant, E. V. Wyk & M. Gini. *An overview of xrobots : A hierarchical state machine based language*. ICRA-2011 Workshop on Software development and Integration in Robotics, 2011.

[Traub 11] M. Traub. Topics in multi-robot teamwork. Master's thesis, Master's thesis, Bar Ilan University. Available at http ://www.cs.biu.ac.il/ galk/Publications/b2hd-meytal-msc.html, 2011.

[Utkin 96] V.I. Utkin & J. Shi. *Integral sliding mode in systems operating under uncertainty conditions*. IEEE International Conference on Decision and Control, 1996.

[VanDenBerg 05] J. VanDenBerg & M. Overmars. *Roadmap-based motion planning in dynamic Environments*. IEEE Transactions on Robotics, vol. 21, pp. 885–897, 2005.

[Vanek 05] B. Vanek, T. Peni, J. Bokor & G. Balas. *Practical approach to realtime trajectory tracking of uav formations.* in Proceeding of the American Control Conference, vol. 1, pp. 122–127, 2005.

[Vidal 03] R. Vidal, O. Shakernia & S. Sastry. *Formation control of nonholonomic mobile robots with omnidirectional visual servoing and motion segmentation.* IEEE International Conference on Robotics and Automation, vol. 1, pp. 584–589, 2003.

[Virginia 04] D. Virginia, V. Javier & D. Frank. *A model of almost everything : Norms, structure and ontologies in agent organizations.* 3rd international joint conference on Autonomous Agent and Multi-Agent Systems (AAMAS), vol. 3, pp. 1496–1497, 2004.

[Virginia 05] D. Virginia, V. Javier & D. Frank. *OMNI : Introduction social structure, norms and ontologies into agent organizations.* Programming Multi-Agent Systems, Second International Workshop ProMas, vol. 3346, pp. 181–198, 2005.

[Volpe 97] R. Volpe, J. Balaram, T. Ohm & R. Ivlev. *Rocky 7 : A next generation mars rover prototype.* Advanced Robotics, vol. 11(4), pp. 341–358, 1997.

[Webster 81] Webster. Webster's third New International Dictionary. Encyclopaedia Britannica Inc., Chicago, 1981.

[Weisbin 00] C. Weisbin & G. Rodriguez. *Nasa robotics research for planetary surface exploration.* IEEE Robotics and Automation Magazine, vol. 7, no. 4, pp. 25–34, 2000.

[Wooldridge 00] M. Wooldridge, N.R. Jennings & D. Kinny. *The gaia methodology for agent-oriented analysis and design.* Autonomous Agents and Multi-Agent Systems, vol. 3(3), pp. 285–312, 2000.

[Xin 06] M. Xin, Fang M., Yibin L., Weidong C. & Yugeng X. *Multi-agent-based Auctions for Multi-robot Exploration.* The Sixth World Congress on Intelligent Control and Automation, 2006. WCICA 2006., vol. 2, pp. 9262–9266, 2006.

[Yamaguchi 01] H. Yamaguchi, T. Arai & G. Beni. *A Distributed Control Scheme for Multiple Robotic Vehicles to Make Group Formations.* Robotics and Autonomous Systems, vol. 36, pp. 125–147, 2001.

[Yamauchi 99] B. Yamauchi. *Decentralized coordination for multirobot exploration.* pp. 111–118, 1999.

[Yang 07] E. Yang & D. Gu. *Nonlinear formation-keeping and mooring control of multiple autonomous underwater vehicles.* Tran-

sactions on Mechatronics, IEEE/ASME, vol. 12, pp. 164–178, 2007.

[Young 88] K. Young. *A variable structure model following control design for robotics applications.* IEEE Transactions on Automatic Control, vol. 33, pp. 556–561, 1988.

[Zadeh 75] L. A. Zadeh. *The concept of a linguistic variable and its application to approximate reasoning-1.* Information sciences, vol. 8, pp. 199–249, 1975.

[Zambonelli 01] F. Zambonelli, N.R. Jennings, A. Omicini & M. Wooldridge. *Agent-Oriented Software Engineering for Internet Applications.* pp. 326–346, 2001.

[Zambonelli 03] F. Zambonelli, N.R. Jennings & M. Wooldridge. *Developing multiagent systems : The gaia methodology.* ACM Transactions on Software Engineering and Methodology (TOSEM), vol. 12(3), pp. 317–370, 2003.

[Zhiqiang 06] C. Zhiqiang, M. Tan, L. Li, N. Gu & S. Wang. *Cooperative hunting by distributed mobile robots based on local interaction.* IEEE transactions on robotics, vol. 22, pp. 403–407, 2006.

[Zhu 09] S. Zhu, D. Wang & Q. Chen. *Standoff Tracking Control of Moving Target in Unknown Wind.* Proceedings of the 48th IEEE Conference on Decision and Control, pp. 776–781, 2009.

[Ziparo 10] V. A. Ziparo, L. Iocchi, P. U. Lima, D. Nardi & P. F. Palamara. *Petri net plans.* Autonomous Agents and Multi-Agent Systems, vol. 23, no. 3, pp. 344–383, 2010.

Résumé : La complexité associée à la coordination d'un groupe de robots mobiles est traitée dans cette thèse en investiguant plus avant les potentialités des architectures de commande multi-contrôleurs dont le but est de briser la complexité des tâches à exécuter. En effet, les robots mobiles peuvent évoluer dans des environnements très complexes et nécessitent de surcroît une coopération précise et sécurisée pouvant rapidement devenir inextricable. Ainsi, pour maîtriser cette complexité, le contrôleur dédié à la réalisation d'une tâche est décomposé en un ensemble de comportements/contrôleurs élémentaires (évitement d'obstacles et de collision entre les robots, attraction vers une cible, planification, etc.) qui lient les informations capteurs (provenant des capteurs locaux du robot, etc.) aux actionneurs des différentes entités robotiques. La tâche considérée dans cette thèse correspond à la navigation d'un groupe de robots mobiles dans des environnements peu ou pas connus en présence d'obstacles (statiques et dynamiques). La spécificité de l'approche théorique consiste à allier les avantages des architectures multi-contrôleurs à ceux des systèmes multi-agents et spécialement les modèles organisationnels afin d'apporter un haut niveau de coordination entre les agents/robots mobiles. Le groupe de robots mobiles est alors coordonné suivant les différentes normes et spécifications du modèle organisationnel. Ainsi, l'activation d'un comportement élémentaire en faveur d'un autre se fait en respectant les contraintes structurelles des robots en vue d'assurer le maximum de précision et de sécurité des mouvements coordonnés entre les différentes entités mobiles. La coopération se fait à travers un agent superviseur (centralisé) de façon à atteindre plus rapidement la destination désirée, les événements inattendus sont gérés quant à eux individuellement par les agents/robots mobiles de façon distribuée. L'élaboration du simulateur ROBOTOPIA nous a permis d'illustrer chacune des contributions de la thèse par un nombre important de simulations.

Mots-clés : Systèmes multi-robots mobiles, Systèmes multi-agents, Évitement d'obstacles, Architecture de contrôle/commande, Modèles organisationnels multi-agents, Simulateur ROBOTOPIA.

Abstract : The difficulty of coordinating a group of mobile robots is adressed in this thesis by investigating control architectures which aim to break task complexity. In fact, multi-robot navigation may become rapidly inextricable, specifically if it is made in hazardous and dynamical environment requiring precise and secure cooperation. The considered task is the navigation of a group of mobile robots in unknown environments in presence of (static and dynamic) obstacles. To overcome its complexity, it is proposed to divide the overall task into a set of basic behaviors/controllers (obstacle avoidance, attraction to a dynamical target, planning, etc.). Applied control is chosen among these controllers according to sensors information (camera, local sensors, etc.). The specificity of the theoretical approach is to combine the benefits of multi-controller control architectures to those of multi-agent organizational models to provide a high level of coordination between mobile agents-robots systems. The group of mobile robots is then coordinated according to different norms and specifications of the organizational model. Thus, activating a basic behavior in favor of another is done in accordance with the structural constraints of the robots in order to ensure maximum safety and precision of the coordinated movements between robots. Cooperation takes place through a supervisor agent (centralized) to reach the desired destination faster ; unexpected events are individually managed by the mobile agents/robots in a distributed way. To guarantee performance criteria of the control architecture, hybrid systems tolerating the control of continuous systems in presence of discrete events are explored. In fact, this control allows coordinating (by discrete part) the different behaviors (continuous part) of the architecture. The development of ROBOTOPIA simulator allowed us to illustrate each contribution by many results of simulations.

Key-words : Multi-mobile robots systems, Multi-agent systems, Obstacle avoidance, Control and Management Architecture, Multi-agent organizationnal models, ROBOTOPIA simulator.