REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE SAAD DAHLEB DE BLIDA
FACULTE DES SCIENCES
DEPARTEMENT D'INFORMATIQUE

C.D.T.A

# MEMOIRE DE FIN D'ETUDES

**Pour l'obtention**

**D'un diplôme de master en informatique.**

Option : Génie système informatique

## THÈME :

# Contribution au développement d'une architecture multi-agents de contrôle d'une équipe de robots hétérogènes dans un système cyber-physique.

**Promoteur :**

Mme N.REZOUG

**Réalisé par :**

Gasmi Abdennour

Ouahal Mohamed

**Encadreur :**

Dr A.HENTOUT

Soutenue le: date soutenance, devant le jury composé de :

| | |
|---|---|
| Nom. Président du jury, grade, organisme | Président |
| Nom examinateur 1, grade, organisme | Examinateur |
| Nom examinateur 2, grade, organisme | Examinateur |

2015/2016

# Remerciments

*Nous remercions tout d'abord, Allah qui nous a donné la force et le courage afin de parvenir à élaborer ce modeste travail.*

*Nous tenons à remercier de tout cœur notre encadreur Monsieur Dr A.HENTOUT et témoigner toute notre reconnaissance pour ses conseils judicieux, de ses remarques objectives et surtout de ses gentillesses permanentes.*

*Nous tenons à exprimer nos sincères remerciements à notre promoteur Mm N.REZOUG enseignante au niveau de l'université USDB d'avoir dirigé notre travaux, pour leur constante disponibilité, pour les conseils qu'ils n'ont cessé de nos prodiguer et enfin pour leur encouragements.*

*Nos vifs remerciements à tous les enseignants qui ont contribués à notre formation, et ainsi tous les gens de prés et de loin qui ont aidé à l'élaboration de ce sujet.et à tous nos professeurs et travailleurs de l'université SAAD DAHLEB –BLIDA.*

*Et enfin, que nos chers parents et familles, et bien avant tout, trouvent ici l'expression de nos remerciements les plus sincères et les plus profonds en reconnaissance de leurs sacrifices, aides, soutien et encouragement afin de nous assurer cette formation de Master 2 dans les meilleures conditions.*

# Dédicaces

*Avant tout, je remercie le grand Dieu, qui nous a aidés à élaborer ce modeste travail.*
*J'ai l'immense honneur de dédier ce modeste travail :*

*A mes très chers parents qui étaient présents pour moi durant toute ma vie.*

*A mes frères et mes sœurs.*
*FAIZ, Fatima, Sadjia, Naoual, Fatiha, Hayat , Adel et Younes.*
*A toute la famille OUAHAL sans exception.*

*A mes amis Krimo, Riad, Abasse, Imene Zeta, Feriel.*
*A mon binome abdennour et sa famille.*

*A tous ceux que j'aime, tous ceux qui m'aiment et tous ceux qui me sont chers.*

*A tous les professeurs et enseignants qui ont collaboré à ma formation depuis mon premier cycle d'étude jusqu'à la fin de mes études universitaires.*

*A tous ceux qui m'ont aidé durant ma vie universitaire*
*A toute la promotion GSI 2016.*

*M.OUAHAL*

# Dédicaces

*Avant tout, je remercie le grand Dieu, qui nous a aidés à élaborer ce modeste travail.*
*J'ai l'immense honneur de dédier ce modeste travail :*

*A mes très chers parents qui étaient présents pour moi durant toute ma vie.*

*A mes frères et mes sœurs.*
*Khaled, Ayoub, Amel, Houda, Asmaa.*
*A toute la famille GASMI sans exception.*

*A mes amis Yahia , Omar, Aymen, Amine.*
*A mon binôme Mohammed et toute sa famille.*

*A tous ceux que j'aime, tous ceux qui m'aiment et tous ceux qui me sont chers.*

*A tous les professeurs et enseignants qui ont collaboré à ma formation depuis mon premier cycle d'étude jusqu'à la fin de mes études universitaires.*

*A tous ceux qui m'ont aidé durant ma vie universitaire*
*A toute la promotion GSI 2016.*

*A.GASMI*

# Tables des matières

# Table des figures

# Liste des tableaux

# Abstract

The aim of this work is to develop a software cyber-physical architecture in which human operators, robots and other objects could register and exchange information with all the registered entities.

The Proposed control architecture is composed of two types of agents organized into two-levels communicating via a local network (Wi-Fi…). We distend wish at the top level, a supervisory agent. This latter processes collected information; then, sends them to the physical-level agents. At the low-level of the control architecture, we distinguish robots agents; each robot is assigned a robot agent dedicated to decision-making in order to select the most suitable robots to carry out operations. The implementation of the control architecture is realized using JADE (Java Agent DEvelopment Framework) platform.

In order to validate the proposed control architecture and analyze its performances, we realized various simulation scenarios of the leader/followers task.

**Keywords:**

Cyber-physics systems, control architecture, multi-agent systems, heterogeneous multi-robot systems.

# Résumé

Notre travail a pour objectif le développement d'une architecture logicielle cyber-physique dans laquelle des opérateurs humains, des robots ou des objets quelconques peuvent s'enregistrer au niveau du contrôleur et échanger des informations avec toutes les entités enregistrées.

L'architecture proposée est composée de deux types d'agents répartis sur deux niveaux connectés par un réseau local (sans fil ou autre). Nous distinguons au sommet, un agent Superviseur. Ce dernier traite les informations collectées de l'extérieur ; ensuite, il les envoie aux agents de niveau physique. Dans le deuxième niveau de l'architecture, nous distinguons des agents robots ; à chaque robot est assigné un agent robot dédié à la prise de décision, afin de sélectionner les robots les plus appropriés à exécuter les opérations. La mise en œuvre de cette architecture de contrôle a été réalisée en utilisant la plateforme JADE (Java Agent DEvelopment Framework).

Afin de valider l'architecture de contrôle proposée et analyser ses performances, nous avons réalisé quelques scénarios en simulation de la tâche de Leader/Followers.

**Mots clés :** Systèmes cyber-physiques, Architectures de contrôle, Systèmes multi-agents, Systèmes multi-robots hétérogènes.

# Introduction générale

# Introduction générale

Généralement, lors de la réalisation de tâches robotiques, les robots se basent uniquement sur les informations récoltées à travers leurs capteurs proprioceptifs et extéroceptifs. Ces derniers récoltent et fournissent des informations liées à l'état interne du robot lui-même et à l'état de l'environnement où évoluent ces robots. Afin de faciliter l'exécution des tâches aux robots et garantir une interaction efficace avec leurs environnements, il faut implémenter un mécanisme d'échange d'information entre eux robots et environnements).

Les évolutions scientifiques survenues cette dernière décennie ont attribué plus d'intelligence aux objets de l'environnement, et ont défini de nouvelles architectures de contrôle des systèmes robotisés basées, principalement, sur l'intégration d'objets distribués, communicants et autonomes dans l'environnement. Cela a été rendu possible grâce à une intégration des tags (ou capteurs), de plus en plus poussée, au sein de multiples objets d'intérêt de l'environnement.

Ces derniers peuvent être utilisés comme support pour aider à localiser exactement les robots dans leur environnement, détecter les obstacles et, éventuellement, connaître la nature de toutes les entités évoluant dans cet environnement (humains/robots, fixes/dynamiques, objets, etc.).

Les systèmes capables d'intégrer toutes ces innovations sont appelés « Systèmes Cyber-Physiques (SCP) ». Développer des systèmes qui interagissent continuellement et de manière dynamique avec leur environnement grâce à l'association d'éléments du monde physique et d'éléments distribués du monde de l'informatique, nécessite des technologies de haut niveau. Pour cela, les SCP utilisent plusieurs technologies et protocoles de communication tels que le NFC, le Zigbee, la RFID, etc.

Les Systèmes Robotiques Cyber-Physiques (SRCP) peuvent être appliqués dans le domaine de la robotique où un ensemble de robots autonomes ayant la capacité de coopérer et de communiquer avec les différentes entités de l'environnement afin d'accomplir des tâches par une gestion plus efficace de l'environnement et la multiplication de services (domiciles, lieux de travail, espaces publics, etc.). À titre d'exemples, on peut citer la surveillance et l'assistance aux personnes âgées ou handicapées, les robots chirurgicaux, etc.

Bien que les SRCP permettent l'interaction entre les différents composants, ils ne disposent pas d'une architecture générique et globale qui permet de les contrôler, c.-à-d. de configurer, d'ordonnancer, de déclencher et de suivre l'exécution de ces différents composants. Une architecture de contrôle a donc pour vocation de coordonner les différents éléments de l'environnement afin d'effectuer différents types de tâches, tout en ayant la capacité de réagir efficacement à différents types d'événements.

Il existe plusieurs approches pour la mise en œuvre d'architectures de contrôle telle que les architectures de contrôle centralisées issues du domaine de l'Intelligence Artificielle (IA), les architectures distribuées issues du domaine de l'Intelligence Artificielle Distribuée (IAD) qui permet de contrôler un système en distribuant les connaissances et les tâches sur plusieurs entités et, les Systèmes Multi-Agents (SMA) qui permettent l'interaction entre les entités du système lors du traitement. Nous avons choisi les SMA vu ces divers avantages.

Un SMA est composé d'un certain nombre d'entités appelées agents, qui exécutent des opérations simples et dont la coopération permettra l'émergence d'un comportement complexe, qui offre un grand avantage pour contrôler un SRCP. Dans ce contexte, nous nous fixons comme objectif le développement d'une architecture logicielle cyber-physique multi-agents dans laquelle des opérateurs humains, des robots ou des objets quelconques (salles, murs, obstacles, objets étiquetés, etc.) peuvent s'enregistrer au niveau du contrôleur et échanger des informations avec toutes les entités enregistrées.

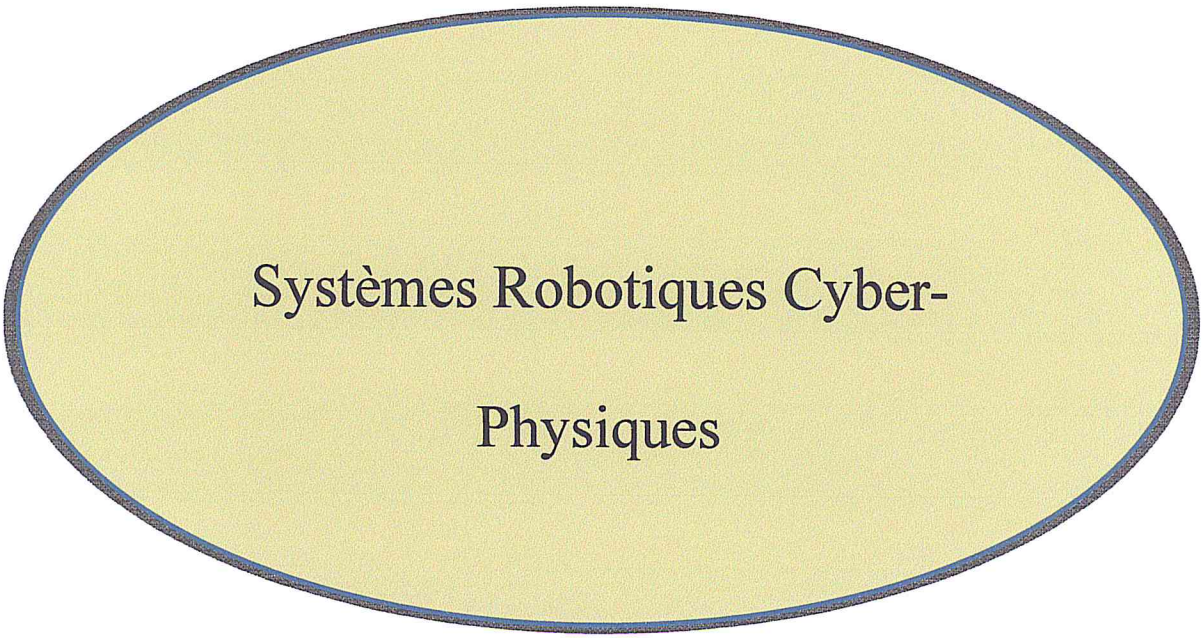
Outre cette introduction générale, ce mémoire est organisé en cinq chapitres, une conclusion générale et une annexe donnés comme suit :

- Le premier chapitre est dédié à la description des SRCP. Nous présentons les SCP, les technologies utilisées par ces systèmes et les différents domaines d'applications. Nous présentons aussi, les SRCP en citant quelques exemples de travaux récents dans ce domaine aussi que leur différents architectures de contrôles.
- Le second chapitre est consacré à la description des architectures multi-robots hétérogènes. Dans ce chapitre nous définissons le Système Multi-Robots (SMR), ainsi que la communication et la coopération dans les SMR et les différents aspects liés à la coopération. À la fin de ce chapitre, nous présentons les différentes classes d'architectures de contrôle des SMR hétérogènes développées et nous citons quelques exemples de la littérature.
- Le troisième chapitre est dédié à la présentation de l'architecture multi-agents de contrôle de SRCP que nous avons proposée. Nous présentons les différentes composantes du

système, les comportements des agents qui y évoluent ainsi que les différentes interactions entre ces derniers grâce à différents diagrammes.

- Le quatrième chapitre décrit l'implémentation de notre système réalisée à l'aide de la plateforme JADE. Nous retrouvons la présentation des outils et des langages utilisés dans la programmation ainsi que les différentes classes réalisées.
- Le cinquième chapitre valide l'architecture de contrôle proposée. Nous présentons les résultats de simulation obtenus via différents scénarios de la tâche Leader/Followers.
- Nous terminons ce mémoire par une conclusion générale ainsi que quelques perspectives de travaux futurs.
- L'annexe a décrit les agents et les systèmes multi-agents (SMA).

# Chapitre 1:



# Chapitre 1
# Systèmes Robotiques Cyber-physiques

## 1. Introduction

Les systèmes informatiques sont de plus en plus utilisés pour recueillir des données à partir de l'infrastructure physique liée à des individus, des entreprises et des communautés. Pour faire d'une telle infrastructure intelligente, une grande capacité de calcul et de communication est intégrée à des nouvelles technologies (capteurs, réseau sans fil, caméras, ...). Cette intégration qui combine le cyber-monde de l'informatique et de la communication avec le monde physique est appelée le système cyber-physiques (SCP).

Nous retrouvons les SCP dans les réseaux électriques intelligents (smart-grid), les réseaux de circulation de véhicule, les télécommunications, les systèmes automobiles et avioniques, les robots coopératifs, etc.

Dans ce chapitre nous allons définir les SCP et montrer les différents domaines d'applications de ces systèmes. En outre, nous allons définir quelques technologies utilisées par ces systèmes. Ainsi, nous présenterons les Systèmes Robotiques Cyber-Physiques en citant quelques travaux récents dans ce domaine. À la fin de ce chapitre, nous présenterons les architectures de contrôle systèmes robotiques cyber-physiques existants.

## 2. Systèmes cyber-physique (SCP)

### 2.1. Définition

Helen Gill [1] a été la première à donner naissance à ce terme à la National Science Fondation aux États-Unis en 2006. Elle a défini les SCP comme "des systèmes physiques, biologiques et résultant de l'ingénierie dont le fonctionnement est intégré, surveillé ou contrôlé par un noyau informatique". D'après elle [1], "l'informatique est profondément intégrée dans tout composant physique. Le noyau informatique est un système intégré, exige généralement une réaction en temps réel et est le plus souvent réparti". Par la suite plusieurs définitions ont été données par les experts de ce domaine. Nous citons entre autres :

- "Les SCP sont l'intégration de calcul informatique avec les processus physiques, c.-à-d. intégrer les ordinateurs et les réseaux pour surveiller et contrôler les processus physiques avec des boucles de rétroaction où les processus physiques influent sur le traitement informatique et vice-versa" [2].
- "Les SCP sont les systèmes qui combinent le cyber-monde de l'informatique et de la communication avec le monde physique réel. Ils sont des systèmes physiques dont les opérations sont surveillées, coordonnées, contrôlées et intégrées par un ensemble d'éléments informatiques et un système de communication" [3].
- "Les SCP sont une nouvelle génération des systèmes dans lesquels trois éléments clés (les 3C) sont étroitement intégrés : Calcul, Communication et Contrôle" [4].

Les composants clés des SCP peuvent être identifiées comme suit (voir figure 1 pour plus de détail) [5] :

- Les entités physiques : incluent des capteurs, des actionneurs, des procédés biologiques ou chimiques, ou des opérateurs humains.
- Le système informatique : c'est la plateforme de traitement et de contrôle, elle consiste en un ou plusieurs ordinateurs et, éventuellement, un ou plusieurs systèmes d'exploitation.
- La communication : représente les mécanismes de communication entre les différentes entités.

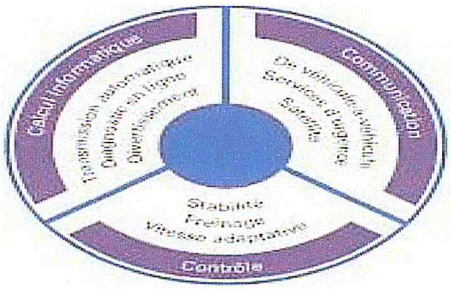



Figure 1:Les trois éléments clés d'un SCP.

La figure suivante (figure 2) montre un schéma générique de SCP, où les composants clés identifiés ci-dessus sont répartis sur deux mondes : le monde physique et le cyber-monde de l'informatique [5].

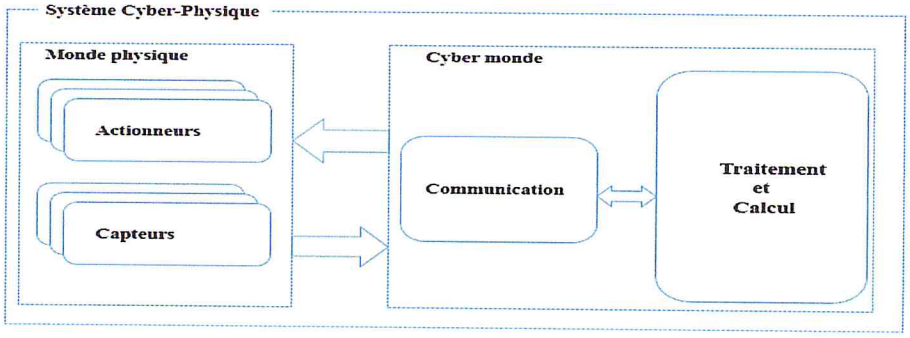



Figure 2: schéma générique d'un SCP [5]

Le SCP lit l'état de son environnement physique grâce à des capteurs. Par la suite, un processus de calcul est déclenché grâce au résultat de détection précédent ; cela aide à élaborer comment la partie physique du système doit réagir ; le résultat de ce processus est envoyé vers l'actionneur par le biais d'un réseau de communication. Généralement, un SCP est caractérisé par [6] :

- L'enregistrement direct des données physiques à l'aide de capteurs et l'influence sur les processus physiques à l'aide des actionneurs.
- L'évaluation et l'enregistrement des données enregistrées et interaction de façon active ou réactive avec le monde physique et numérique.
- La capacité de se connecter avec d'autres SCP et dans les réseaux mondiaux via les moyens de communication numériques (sans fil et/ou câblés, locaux et/ou mondiaux).
- L'utilisation des données et services disponibles à l'échelle mondiale.
- Le comportement d'une série d'interfaces homme/machine dédiées.
- Le contrôle distribué.
- L'incertitude aux lectures.
- Les performances temps-réel.
- Une large distribution géographique.

### 2.2. Technologies de système cyber-physique (SCP)

Elle consiste à développer des systèmes qui interagissent continuellement et de manière dynamique avec leur environnement grâce à l'association d'élément du monde physique et d'élément du monde de l'information distribuée. Elle nécessite des technologies de haut niveau. Pour cela, les SCP utilisent plusieurs technologies comme la RFID, la NFC et le protocole de communication zigbee, Bluetooth et wifi. D'autre solution sont en cours de développement tels que les systèmes d'indentification acoustique, les micro-ondes, les systèmes optiques, etc.

### 2.2.1. Système RFID

Le système RFID(RadioFrequencyIDentification). Ou identification par radio fréquence, est une technologie qui permet de mémoriser et de récupérer des informations à distance grâce à une étiquette qui émet des ondes radio. Il est utilisé pour détecter la présence d'objets physiques. Un système RFID est composé de deux entités qui communiquent entre elles (voir figures 3) [7] :

- Des tags ou étiquettes intelligentes.
- Une station de base ou lecteur RFID.

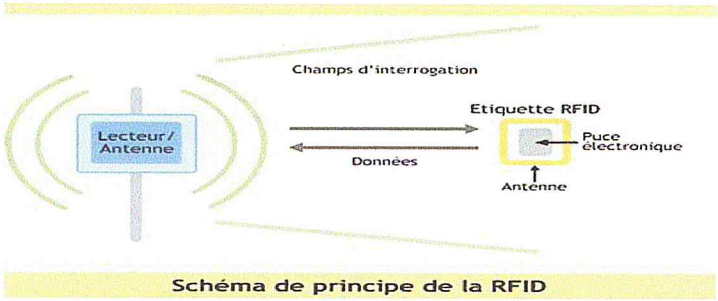



Figure 3: Fonctionnement de système RFID [1].

Comme il est montré dans la figure 3, l'étiquette RFID, aussi appelé transpondeur, est elle-même équipée d'une puce reliée à une antenne. L'antenne permet à la puce de transmettre les informations (numéro de série, poids...) qui peuvent être lues grâce à un lecteur émetteur-récepteur. Une fois les informations transmises au lecteur RFID équipé d'une antenne intégrée ou externe, celui-ci n'a plus qu'à convertir les ondes-radios en données et celles-ci pourront être lues par un logiciel RFID. La distance entre l'étiquette RFID et le lecteur peut aller de quelques centimètres jusqu'à plusieurs centaines de mètres [8].

### 2.2.2. Technologie NFC

La NFC (Near Field Communication), ou communication en champ proche, est une technologie de communication à très courte distance (environ 10 cm) basée sur la RFID. Une technologie NFC combine l'interface d'une carte à puce et un lecteur au sein d'un seul périphérique. Pour faire communiquer deux périphériques, le principe est de les faire rapprocher l'un de l'autre. Ceci déclenche instantanément l'interface sans fil qui se charge

automatiquement de configurer une connexion réseau. Le principe de la communication NFC est l'alimentation, l'émission, puis la réception (voir figure 4).

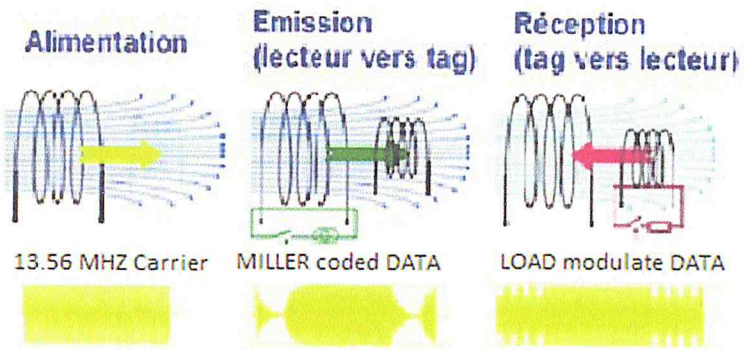



Figure 4: Principe de la communicationNFC [9].

La NFC fonctionne avec l'utilisation d'une induction magnétique : Un lecteur émet un faible courant électrique qui crée un champ magnétique entre les deux appareils. L'acteur démarrant la connexion est appelé initiateur. Un client reçoit le champ et le transforme en impulsion électrique qu'il peut traduire en bits de données. Ce tracteur est appelé la cible. La manière dont la réponse est envoyée dépend du mode de fonctionnement [10] (mode émulation de carte, mode lecteur, mode pair-à-pair).

### 2.2.3. Protocole de communication Zigbee

ZigBee est un protocole de haut niveau permettant la communication de petites radios, à consommation réduite, basée sur la norme IEEE802.15.4 pour les réseaux à dimension personnelle (Wireless Personal Area Networks : WPAN).

La technologie ZigBee a pour but la communication de courte distance telle que le propose déjà la technologie Bluetooth, tout en étant moins chère et plus simple. À titre d'exemple, les nœuds ZigBee classiques nécessitent environ 10% du code nécessaire à la mise en œuvre de nœuds Bluetooth ou de réseaux sans fil, et les nœuds ZigBee les plus élémentaires peuvent ainsi descendre jusqu'à 2% [11].


Figure 5: Équipements de protocole Zigbee [11].

Parmi les avantages que procure ce protocole de communication, nous pouvons citer la faible consommation d'énergie, l'utilisation optimale de la bande passante, et son faible coût de mise en œuvre. Par contre son débit est bien inférieur à celui du wifi : 250kb/s.

### 2.3. Domaines d'application des SCP

La science de SCP n'est apparue que récemment mais a déjà fourni des résultats spectaculaires dans divers domaine:

#### 2.3.1. Domaine de transport

Le développement et le déploiement à grande échelle des applications des SCP représentent une véritable révolution dans le domaine des transports. En effet, les SCP liés à ce domaine sont connues par ITS (Intelligent Transport Systems) [12].

- **Véhicules autonomes** : des véhicules intelligents et autonomes équipés de capteurs et de systèmes de commande innovants sont capables de rouler automatiquement dans le trafic réel et sur une infrastructure non spécifique sans l'intervention d'un être humain. Ces véhicules contribuent à renforcer la sécurité routière et à améliorer le quotidien des malvoyants, des personnes souffrant d'un handicap ou encore des personnes âgées. Bien que la notion de voiture autonome soit nouvelle, on peut voir aujourd'hui des voitures équipées de système d'aide au stationnement, de freinage d'urgence automatique, etc. [13].De nombreux constructeurs automobiles travaillent actuellement sur des projets de véhicule autonome. Par exemple, Google travaille sur une Google-Car en équipant huit véhicules par un système de pilotage automatique qui utilise un lidar, une caméra, des radars, un récepteur GPS (figure 6) et les rendant ainsi totalement autonomes. Ils auraient déjà parcouru plus de 800000 km en Californie sans avoir provoqué le moindre accident [12].

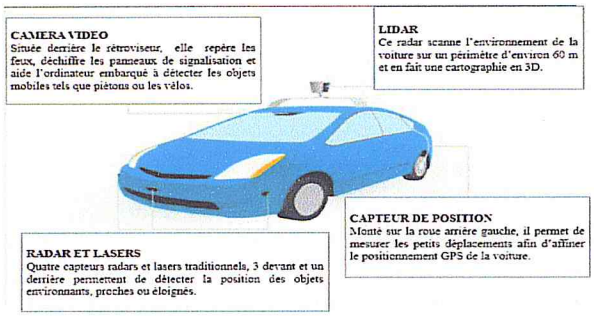



Figure 6: Véhicule autonome de Google [12].

- **Parkings intelligents** : Le principe est de mettre sur chaque place du parking un système à base de capteur qui détecte les masses métalliques quand une voiture se gare. Citons par exemple les parkings qui se basent sur des Panneaux à Messages Variables(PMV), un système très répandu dans les grandes métropoles qui indique le nombre de places disponibles [14], le parking de port de San Francisco [15], où le conducteur sera guidé, par GPS, jusqu'à la place libre la plus proche, en utilisant son Smartphone.

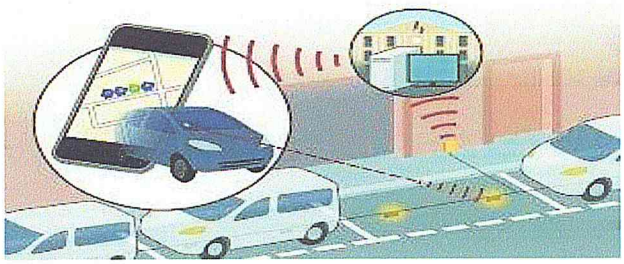


Figure 7: Système de parking intelligent [15].

### 2.3.2. Domaine médical

On utilise les SCP médicaux (SCPM) dans les hôpitaux pour fournir des soins continue hautement qualifie aux patients. Ces systèmes englobent différent équipements de diagnostic médical et thérapeutique et les appareils chirurgicaux.

Les SCPM utilisent des micro-capteurs autonomes avales ou implantés sous le peu du patient afin de collecter et stocker les données physiologiques (tension artérielle, battements de cœur, etc.) et de détecter des anomalies et assurer une surveillance permanente des oranges vitaux de l'être humain. Ces capteurs permettront aux médecins de surveiller les données biométriques clés en dehors de l'environnement hospitalier.

Citons comme exemple le système BodyGuardian [12] qui utilise des algorithmes cliniques développés par des médecins pour assurer la surveillance à distance des patients atteints d'arythmies cardiaques (figure 8). La surveillance est rendue possible grâce à un petit

capteur fixé sur le corps du patient. Ce capteur collecte des données importantes (électrocardiogramme, rythme cardiaque, fréquence respiratoire, etc.) et les transmet aux médecins par l'intermédiaire de la connexion du téléphone mobile du patient, de sorte que ces derniers peuvent surveiller leurs patients avec leur tablette ou leur ordinateur de bureau.

Figure 8: Système BodyGuardian [12].

### 2.3.3 Domaine militaire

Le SCP est surtout utile pour le contrôle, la communication, le calcul, la tolérance aux pannes. Le déploiement rapide et intelligence rendent le SCP une solution prometteuse dans le domaine militaire ; c'est pour cela que plusieurs projets ont été lance pour aider les unités militaires Par exemple, le projet JHPSSL(Joint High Power Solid-State Laser) [16] consiste à équiper des navires, des véhicules et des avions militaires de dispositif laser anti-missiles. Ce dernier est doté d'un système complexe permettant de détecter, de suivre et de détruire les missiles et rockets avant qu'ils n'atteignent leur cible.

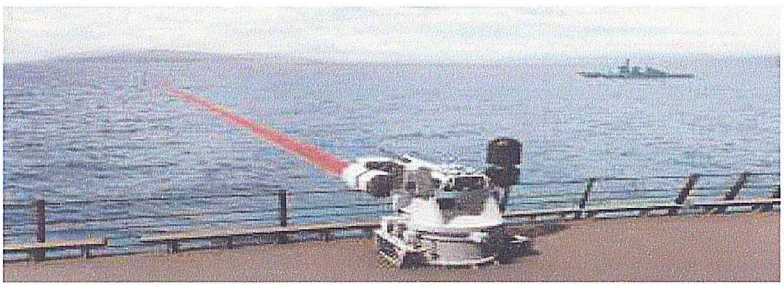


Figure 9: Laser anti-missiles de projet JHPSSL [16].

### 2.3.4. Domaine environnemental

Le champ d'applications des SCP est de plus en plus élargi. En effet, afin de bénéficier de robustesse des SCP, des travaux ont été élaborés dans le domaine de protection de l'environnement surtout dans le nettoyage industriel, réduire l'émission de gaz, etc. Comme exemple de ces applications, nous citons le nettoyage d'algues et surveillance des bateaux (figure 10). En effet, le travail présenté dans [17] répond à un problème qui se produit pour environ 1 million de personnes chinoises. Lak Tai est l'un des endroits qui combattent les algues. Les scientifiques ont mis au point un SCP utilisant [17] :

- Un dispositif de capteurs qui utilise des algorithmes pour déterminer ce qui doit être envoyé à travers un réseau 3G afin de détecter les algues et les suivre sur l'eau.
- Centres d'informations qui analyser ont les caractéristiques des informations envoyées par les capteurs afin de rediriger les bateaux de traitement où aller après que la menace est déterminée pour assurer une intervention rapide et efficace.

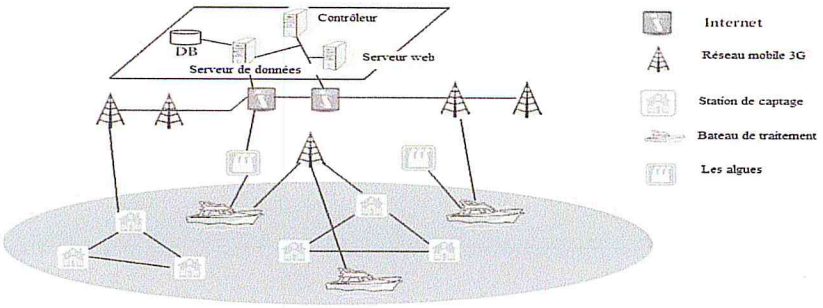



Figure 10: Architecture de système de nettoyage d'algues et de surveillance des bateaux [17].

## 3. Systèmes Robotiques Cyber-Physique (SRCP)

### 3.1. Définition

Un Système Robotique Cyber-Physique(SRCP) peut être défini comme étant un ensemble de robots autonomes fonctionnant dans un environnement intelligent où les robots ont la capacité de coopérer et communiquer avec les différentes entités de cet environnement afin d'accomplir des tâches. Un SRCP comporte essentiellement trois grands éléments de base [5] :

- Les entités physiques : incluent les opérateurs humains, les objets et les robots. Ces entités sont équipées de capteurs, d'actionneurs, des tags et des lecteurs pour leurs permettre la perception de l'état de l'environnement, l'identification et la localisation des objets, etc.
- Le système informatique : c'est le processus de traitement et de contrôle intégré dans les entités physiques pour garantir la réaction en temps réel de ces dernières.
- Le système de communication : englobe les mécanismes de communication entre les différentes entités physiques de l'environnement (Wifi, Bluetooth,etc.).

### 3.2. Applications des SRCP

#### 3.2.1. Projet ADREAM

Le matériel, le logiciel, les réseaux et la robotique pour suivant leur forte évolution, le projet ADREAM a pour but, à la fois, de développer ces derniers, d'anticiper les futures synergies entre eux, de préparer les outils nécessaires à cette conception, et de proposer les premières expérimentations s'y rapportant.

ADREAM se situe ainsi dans la problématique émergente des systèmes ubiquistes et des agents mobiles autonomes, situés dans des environnements ayant des infrastructures de communication à la fois fixes et mobiles, contraintes en ressources (calcul, énergie), et nécessitant de fortes propriétés de performances, de robustesse et de résilience vis-à-vis d'entraves externes, même non prévisibles (obstacles, défaillances, niveaux de confiance et de coopération, etc.).

Dans ce contexte, le double objectif du projet est d'abord, de bâtir les méthodologies et les solutions système nécessaires à la mise en réseau massive d'objets et d'agents intelligents, intégrés dans des environnements munis de multiples capteurs et actionneurs, et, ensuite, de déployer et d'évaluer les méthodes proposées et les résultats obtenus dans un contexte d'application réel et de complexité significative [18] .

Figure 11: Robot en cours de l'exécution d'une tâche [18].

### 3.2.2. Systèmes robotique d'assistance aux personnes âgées et handicapées

Le principe est de construire des robots intelligents, qui facilitent le déplacement des personnes âgées ou handicapées et les aident dans leurs vies, en communiquant avec l'environnement externes. Dans ce cadre, plusieurs travaux ont été réalisés, citons en particulier. Le travail présenté dans [19] qui décrit une application de robot d'assistance qui combine une interface sans fil portable basée sur l'électro-oculographie [1] (EOG) et un système RFID. Cette application d'assistance est destinée aux utilisateurs handicapés qui souffrent d'un handicap moteur sévère. Il se compose d'un environnement dans lequel les utilisateurs peuvent apporter un verre et une bouteille d'eau de plus près avec seulement l'aide de leur mouvement de l'œil en utilisant un bras manipulateur (voir figure 12).

1. Examen de l'oeil destiné à enregistrer le potentiel de repos (activité électrique de base, en l'absence de stimulation) de cet organe lors des mouvements oculaires.

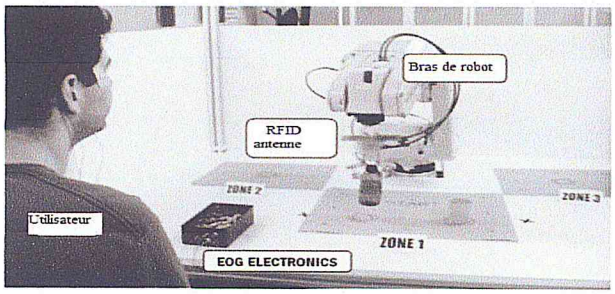



Figure 12: Robot d'assistance basé sur l'électro-oculographie et le système RFID [19].

- Le fauteuil roulant fabriqué par Dupont Médical (figure 13). Ce fauteuil est equipé de dispositifs qui aident à éviter les collisions et garantir la sécurité de la navigation [20]. Les dispositifs comprennent des capteurs à ultrasons à faibles coûts et à infrarouge pour détecter les obstacles autour du fauteuil roulant. La vitesse du fauteuil est réduite en fonction de la distance entre le fauteuil roulant et les obstacles, même si l'utilisateur tente d'aller plus vite. Un arrêt de sécurité sur le fauteuil roulant est également prévu en cas de danger. Aussi, un feedback visuel est fourni pour l'utilisateur et l'aide à comprendre le comportement du fauteuil roulant.

Figure 13: Fauteuil roulant fabriqué par DupontMedical [7].

### 3.2.3. Robots chirurgicaux

Un robot médical est un système robotique utilisé dans le cadre d'une application thérapeutique, par exemple lors d'une chirurgie ou au cours d'un programme de réhabilitation neuro-motrice. Du fait des contraintes importantes en termes de sécurité, ce type de robot est en général doté d'un faible niveau d'autonomie [21].

Le robot le plus utilisé aujourd'hui dans ce secteur est le robot Da Vinci. Il est utilisé, principalement aux États-Unis et en Europe, pour diverses opérations chirurgicales dites mini-invasives dans les cavités abdominales et thoraciques des patients. Son application principale

est la chirurgie de la prostate : 60 % des opérations de la prostate aux États-Unis ont eu recours à un Da Vinci [22]

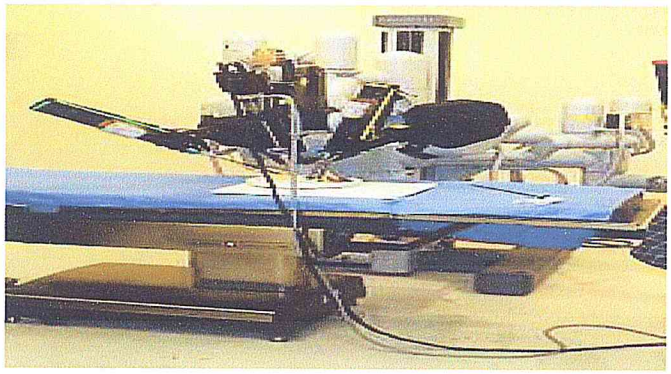


Figure 14: Robot chirurgical da Vinci [22]

### 3.2.4. SRCP pour l'automatisation des laboratoires scientifiques

Pour garantir la fiabilité et l'extensibilité du système dans les grands laboratoires, Liu et al. [23] proposent un SRCP en se basant sur un module de localisation appelée StarGazer Localization. Ce dernier utilise des caméras infrarouges et un certain nombre de capteurs (ceiling passive landmarks ) pour capturer une série continue d'images qui seront renvoyées avec leurs identifiants au serveur central. Chaque module est connecté par le même réseau Wifi avec une adresse IP indépendante. La communication entre les différents modules est basée sur le protocole TCP/IP.

Le modèle représenté sur la (figure 15) représente le schéma du système qui comprend :

- Un système de gestion des processus de laboratoire, Process Management System (PMS).
- Un contrôleur central de serveur distant, Remote Server Control Center (RRC).
- Des robots mobiles où chacun est équipé d'un contrôleur central, Robot Board Control Center (RBC).

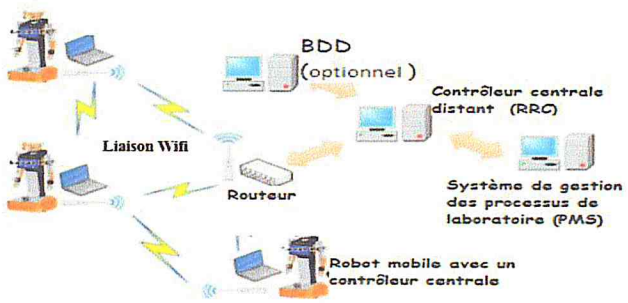



Figure 15 : Schéma synoptique du système proposé dans [24].

La stratégie de contrôle du système proposé peut être expliquée comme suit : Le plus haut niveau est le système de la gestion de processus (PMS), qui s'occupe de la gestion des tâches de différents modules de laboratoire. Lorsqu'une tâche se présente, le PMS envoie une commande (requête) au système de commande de robot(RRC). Quand ce dernier reçoit la tâche, il sélectionne un robot mobile approprié et lui donne un chemin. Lorsque le RBC d'un robot obtient le chemin, il contrôle le matériel du robot pour exécuter le chemin reçu. Quand les robots arrivent à la position attendue, le RBC active les bras du robot pour exécuter la tâche.

## 4. Contrôle d'un Système Robotique Cyber-Physique

Un SRCP est un système complexe qui est constitué d'un nombre important d'entités de natures hétérogènes ; chaque entité interagit avec les entités voisines selon des règles.

Le domaine de l'Intelligence Artificielle (I.A) avait pour objectif d'écrire et à résoudre des problèmes complexes. Dans ce domaine, il est possible de construire des programmes informatiques, capables de contrôler un ensemble d'entités existant dans le système en centralisant l'intelligence (contrôle centralisé) au sein d'un système unique [25].

Il est cependant difficile d'entrer dans une même base de données, les connaissances, les compétences d'individus totalement différents qui communiquent entre eux. De ce fait, nous pouvons dire que l'intelligence artificielle est jugée comme inadéquate pour contrôler un SRCP.

Cela a incité les chercheurs à trouver une autre méthode permettant de remédier à cet embarras en décomposant les problèmes sur des entités de résolveurs, d'où vient le terme Intelligence Artificielle Distribue (IAD). L'IAD permet de contrôler un système en distribuant les connaissances et les tâches sur plusieurs entités, ce qui permet d'augmenter la vitesse de calcul et de raisonnement. Malheureusement, l'absence d'interaction entre les différentes entités lors de traitement rend l'IAD inadaptée pour contrôler un SRCP.

Les chercheurs ont, ensuite, réfléchit sur la possibilité de faire interagir ces entités lors du traitement pour améliorer les résultats. De cette idée, que sont apparus les Systèmes Multi-Agents (SMA). Un SMA, permet d'introduire dans un système, un ensemble d'individus (ou agents) dotés de connaissances, d'intentions et de capacités d'évolution différentes. Ces agents sont capables d'interagir entre eux.

L'approche multi-agents offre un grand avantage pour contrôler un SRCP car le paradigme agent dispose de tous les concepts (objet est une entité, comportement réactifs,

interaction, adaptation, auto organisation) nécessaires pour prendre en charge cette nouvelle classe de systèmes.

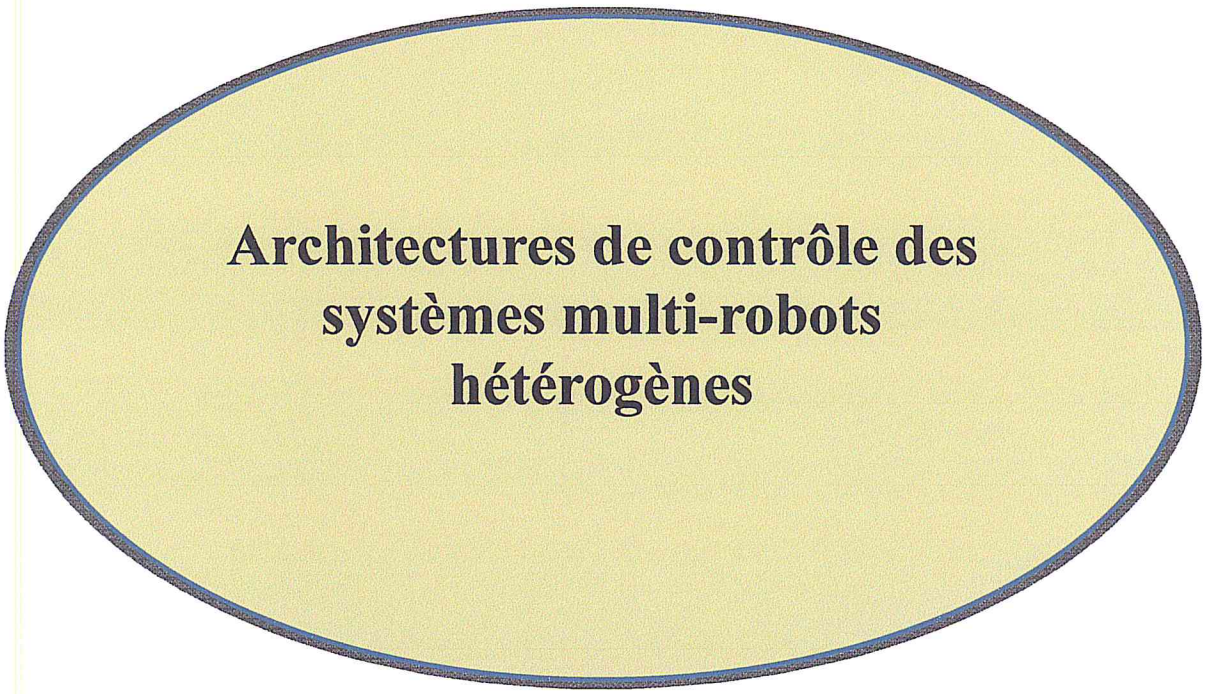
## 5. Conclusion

Dans ce chapitre nous avons présenté les SCP, leur technologie et leurs domaines d'applications. Nous avons aussi défini les SRCP en décrivant quelques exemples d'applications et de recherche, dans ce domaine. Nous avons de plus vu la grande diversité qu'offrent ces systèmes et qui englobent quasiment tous les domaines.

De nos jours, les SCP et les SRCP peuvent changer la façon dont les individus et les organisations interagissent et contrôlent le monde physique ; dans le futur leur importance ne fera que croitre. Ces systèmes complexe combine le calcul, la communication, et la physique, ils sont plus difficiles à modéliser, plus difficile à concevoir et à analyser, et leur contrôle est actuellement une préoccupation majeure de la recherche.

Dans ce qui va suivre, nous allons nous intéresser 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. Notre travail aura pour but d'implémenter une architecture de contrôle d'un SRCP.

# Chapitre 2:


Architectures de contrôle des
systèmes multi-robots
hétérogènes

# Chapitre 2
# Architectures de contrôle des systèmes multi-robots hétérogènes

## 1. Introduction

Depuis une vingtaine d'années, la robotique fait face à une croissance rapide dans la complexité des besoins et des exigences pour des robots chargés de tâche multiples, capables de se coordonner, et développés de telle manière que des garanties de sûreté et de sécurité puissent être vérifiées et certifiées en vue d'une utilisation dans un environnement humain. En parallèle, une évolution similaire dans le domaine des systèmes temps-réel embarqués répartis a justifié l'émergence du domaine des systèmes cyber-physiques reflétant une montée similaire en complexité. D'autres domaines sont également apparus, comme l'informatique autonomique, partageant les mêmes objectifs scientifiques dans la conception et la mise en œuvre de leurs architectures logicielles.

Dans ce chapitre, nous allons définir les systèmes multi-robots, ainsi nous présenterons la communication et la coopération dans les systèmes multi-robots et les différents aspects liés à la coopération. À la fin de ce chapitre, nous présenterons les différentes classes d'architectures de contrôle pour les systèmes multi-robot hétérogènes développées dans la littérature. En l'occurrence, nous détaillerons les spécificités des architectures centralisées par rapport aux architectures distribuées et nous citons quelques exemples de la littérature.

## 2. Systèmes Multi-Robots

### 1.1. Définition

D'après [26] [27], la robotique est une science complexe du fait qu'elle nécessite le concours de plusieurs disciplines comme l'électronique, la mécanique et le génie logiciel. Cette complexité est accentuée avec le passage aux systèmes multi-robots (SMR).

### 1.2. De l'homogénéité à l'hétérogénéité

La majorité des premiers travaux sur les SMR s'est intéressée aux systèmes homogènes constitués de robots identiques. Cependant, ces dernières années ont vu une évolution des travaux de la communauté robotique vers les SMR hétérogènes. Il s'agit de systèmes constitués de robots dotés de capacités différentes notamment au niveau physique : caractéristiques mécaniques, mode de locomotion, capteurs, actionneurs. Les motivations fondamentales de la recherche en SMR sont [28] :

- la capacité de résoudre des problèmes qui sont intrinsèquement distribués dans l'espace, dans le temps.
- la capacité de résoudre des problèmes plus rapidement grâce au parallélisme.
- la capacité d'augmenter la robustesse des solutions par la redondance.

Dans une proportion importante de la recherche sur les SMR, les avantages du parallélisme, de la redondance des solutions distribuées dans l'espace et dans le temps est obtenu par l'utilisation de robots homogènes, qui sont complètement interchangeables. Cependant, un nombre croissant de recherches essaye de répondre aux questions liées à l'utilisation de robots hétérogènes. Ces recherches impliquent généralement un nombre relativement faible de robots peut-être de l'ordre d'une dizaine de robots ou moins. Même la recherche en robotique homogène est rarement expérimentée avec des équipes de plus de dix à vingt robots.

Les applications complexes futures qui exigent l'utilisation simultanée de grandes équipes avec plusieurs capteurs, qui ne peuvent pas être groupés sur un seul type de robot. Les robots devraient sans doute être conçus aussi avec des tailles plus petites, ce qui limiterait leur charge utile, ou bien rendrait certains capteurs nécessaires trop chers à dupliquer dans toute une équipe de plusieurs robots. Cela conduit à la nécessité de permettre à de nombreux robots hétérogènes de travailler ensemble en coopération pour résoudre des applications intéressantes.

### 1.3. Communication et coopération dans les SMR

Du point de vue général, une des questions centrales qu'on doit traiter dans un SMR, quel que soit le domaine d'application, est comment faire coopérer efficacement les robots pendant une mission, d'une manière ou d'une autre, soit automatiquement soit, éventuellement, par l'intervention d'un opérateur externe. La coopération signifie que les

robots doivent communiquer pour échanger des informations et coordonner leurs actions dans le but d'accomplir une mission commune globale [29].

La coopération est le point clé pour exploiter le potentiel des SMR [30]. La communication est une condition préalable et indispensable pour n'importe quel algorithme de coopération. La présence d'un canal de communication sûr et de débit suffisant permettrait de mettre en place un mécanisme de coopération sophistiqué et efficace.

La communication entre les robots dans une équipe peut être réalisée implicitement ou explicitement. La communication implicite, typiquement via l'environnement, est généralement accomplie par les actionneurs et les capteurs des robots. Cela limite à la fois la quantité de données transmises et le degré d'abstraction des informations ainsi échangées. Par conséquent, la communication implicite ne convient pas pour des mécanismes sophistiqués de coopération. Il faut recourir à la communication explicite. D'autre part, avec l'avancement des technologies de communication, les robots d'aujourd'hui sont équipés d'interfaces de communication sans fil haut-débit qui leur fournissent un moyen de communication sûr.

## 2. Systèmes Multi-Robots Collaboratifs

La robotique est devenue en quelques années une science importante qui ne cesse d'évoluer. Les chercheurs parviennent petit à petit à donner à des machines une intelligence artificielle. Cependant, pour certaine tâche, les robots peuvent être amenés à devoir travailler en groupes pour la réalisation d'un objectif en commun. Dans un SMR coopérant il est nécessaire de s'intéresser aux aspects suivants :

### 2.1. Tâches et Opérations

L'idée de base dans de nombreux travaux en robotique est connue comme le paradigme « diviser puis attribuer » [30] [31] [32]. Il s'agit de décomposer une tâche robotique en un ensemble d'opérations. Cette subdivision se poursuit jusqu'à ce que l'on arrive à des opérations élémentaires. Une opération est dite élémentaire si elle est directement réalisable par un robot seul.

Une fois la décomposition réalisée, il faut répartir les opérations élémentaires sur les robots. Puis, les robots vont réaliser ces opérations, et ainsi accomplir la tâche. Le degré de coordination des actions des robots dépend du degré d'interdépendance des opérations. Cette interdépendance doit donc être explicitement exprimée dans la définition des opérations.

La coordination est nécessaire pour la réalisation d'une tâche. En effet, une tâche se décompose en un ensemble d'opérations élémentaires interdépendantes. Un robot qui travaille

sur une de ces opérations élémentaires doit coopérer avec au moins un autre robot réalisant une autre opération élémentaire.

### 2.2. Mécanisme de coopération

La façon dont une tâche est décomposée en opération, l'ordre d'exécution de ces tâches et le niveau de synchronisation des actions des robots pendant l'exécution des opérations élémentaires représentent ensemble la logique du mécanisme de coopération.

Une méthode très utilisée pour répartir les opérations élémentaires sur les robots est l'appel d'offre [28]. Il emploie le protocole de réseau contractuel ou Contracta-Net Protocol (CNP).

Par le moyen du CNP, les robots négocient entre eux afin d'optimiser une fonction objectif. Le processus de négociation implique généralement des communications explicites entre des robots sur les opérations à réaliser. Étant donné une opération, chaque robot disposant de capacités requises pour la réaliser émet une offre qui dépend de ses ressources. La détermination du robot «gagnant» est faite généralement de manière gloutonne en fonction des profits estimés.

### 2.3. Performance du système et fonction objectif

La répartition des opérations élémentaires entre des robots doit être réalisée de manière à ce que le SMR soit le plus performant possible pour réaliser une tâche. La performance du système peut être représentée par des caractéristiques comme, par exemple, le temps d'exécution de la tâche, la complexité algorithmique, la robustesse et la tolérance aux pannes. Elle peut dépendre de la structure globale du système, par exemple, de la stratégie de décomposition des tâches ou des caractéristiques de la communication, etc. Idéalement, tous ces facteurs devraient être pris en compte dans l'évaluation de la performance du système. Cependant, une telle quantité est souvent difficile à mesurer lors de l'exécution du système [30].

## 3. Architectures de contrôle en robotique

Une distinction principale entre les architectures de contrôle robotique cyber-physique porte sur le choix de centraliser le contrôle du SMR ou de le distribuer sur les entités robotiques. On trouve alors des architectures de contrôle centralisées et des architectures de contrôle distribuées.

### 3.1. Architecture de contrôle centralisée

Comme son nom l'indique, le contrôle du SMR est délocalisé par rapport à la structure physique des robots ; il se trouve au niveau d'une unité centrale (le superviseur) qui gère et garantit l'exécution de l'opération. L'unité contient les parties sensorielles (capteurs) afin de collecter les informations de l'environnement. 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 [33], ce type d'architectures de contrôle a été essentiellement motivé par deux facteurs :

- L'ambition d'alléger la structure physique des robots : en effet, les robots utilisés n'ont besoin ni de capteurs embarqués ni d'une puissante unité de calcul. Ceci réduit considérablement leur coût.
- La concentration des capteurs au niveau d'une unité centrale offre une meilleure connaissance de l'environnement global, ce qui peut assurer une meilleure prise de décision par rapport à des robots se basant sur des informations locales fournies par des capteurs embarqués.

### 3.2. Architecture de contrôle distribuée

Par opposition aux architectures de contrôle centralisées, les ressources (capteurs, unités de calcul, etc.) 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.

La distribution des ressources peut être très avantageuse si la mission du SMR devient trop complexe et fastidieuse. En effet, la complication de la tâche du système SMR entraîne naturellement la complication du contrôle nécessaire si bien qu'il devient difficile, voire impossible, sa réalisation par un superviseur [34]. Par exemple, le simple ajout de nouveaux robots signifie de nouveaux agents (avec tous les calculs relatifs) que le superviseur doit gérer. Ce type d'architectures de contrôle est donc venu dans l'ambition de pallier aux limitations des architectures de contrôle centralisées.

### 3.3. Architecture de contrôle centralisée ou distribuée ?

En pratique, beaucoup de SMR n'ont pas un contrôle strictement centralisé ou strictement distribué et utilisent des architectures de contrôle hybrides pour bénéficier des avantages des deux approches.

Le tableau suivant présente les différences observées principalement au niveau des trois modules nécessaires à la commande d'un robot mobile, à savoir : la perception et la localisation, la décision, et l'action.

| Module | 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 (trajectoire, points de passages, etc.). |
| **Action** | Les entités robotiques appliquent directement la 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. |

Tableau 1: Architecture centralisée versus distribuée. [35]

La classification des architectures de contrôle peut alors être illustrée comme dans la (figure 16) suivantes.

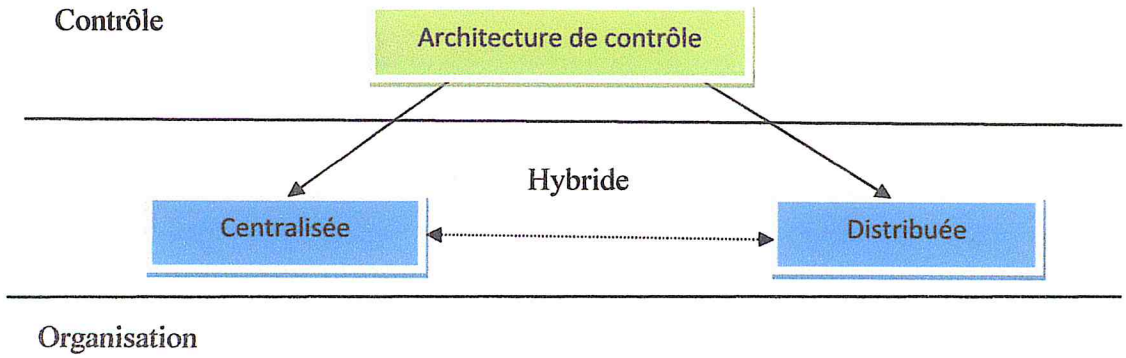



Figure 16 : Classification des architectures de contrôle selon leur Organisation centralisée ou distribuée.

### 3.4. Architectures de contrôle pour systèmes multi-robots

Après avoir cherché dans plusieurs références des informations sur les architectures de contrôle robotique cyber-physique nous avons rien trouvé sur ce sujet ; c'est pour cela qu'on a conclus que les architectures de contrôle RCP étant quasi-inexistants dans la littérature, nous *allons présenter dans ce qui suit, quelque architecture de contrôle de SMR :*

#### 3.4.1. Architecture d'ALLIANCE

L'architecture ALLIANCE [36] a été développée dans le but de permettre la coopération au sein de SMR hétérogènes. Elle se veut une architecture distribuée qui impose certaines règles d'implantation de façon à assurer une certaine tolérance aux fautes. La (figure 17) résume schématiquement le fonctionnement de cette architecture. L'architecture ALLIANCE implémente à un premier niveau les principes de la « Subsumption Architecture» [37] par l'utilisation de comportements réactifs (fonctions reliant directement les actuateurs aux capteurs). Ensuite, à un deuxième niveau, le principe de «Behavior Set» est utilisé dans le but de contrôler l'activation ou l'hibernation de groupes de comportements situés au premier niveau. Ce deuxième niveau permet donc à l'architecture de se reconfigurer dynamiquement, ce qui permet d'adresser la problématique de la coopération de systèmes multi-robots hétérogènes. Pour arriver à établir une telle coopération, l'architecture utilise un troisième niveau, basé sur le principe de «Motivational Behavior». Ce troisième niveau permet tout simplement d'activer/désactiver le deuxième niveau, celui des «Behavior Sets». Mais, en plus de considérer les informations sensorielles comme le font les autres niveaux, il considère des informations de motivation interne permettant entre autre l'inhibition latérale des motivations (cross-inhibition), et finalement des informations provenant de la communication inter-robots.

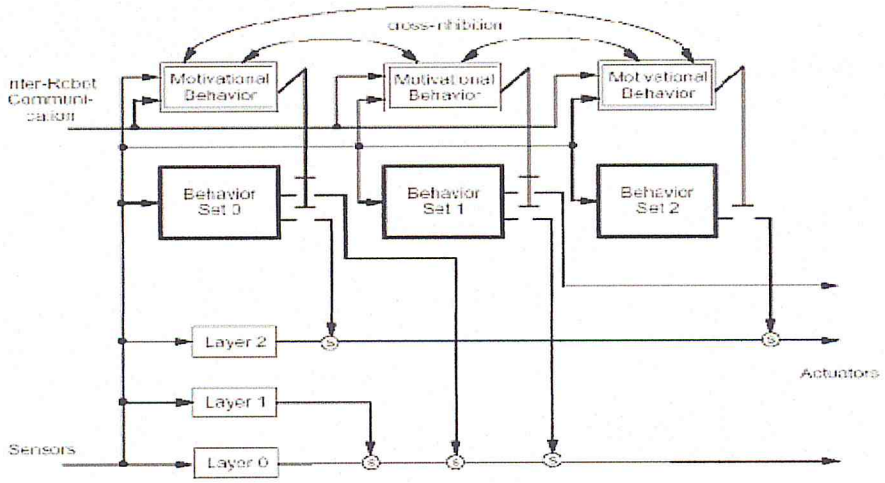



Figure 17 : Schéma résumant l'architecture ALLIANCE [36]

### 3.4.2. Architecture de CAMPOUT (Control Architecture for Multirobot Planetary OUTposts)

CAMPOUT [38] est une architecture qui a été développée par la NASA au Jet Propulsion Laboratory du California Institute of Technology. L'architecture CAMPOUT se veut une architecture hybride (Centralisé et Décentralisé). CAMPOUT est a priori destinée pour les SMR à perception et cognition distribuées. Elle part du principe que chacun des robots est une entité indépendante et que la prise de décision se fait de façon entièrement, et strictement, distribuée.

Cette architecture est composée de trois couches distinctes. La couche supérieure permet la planification, l'allocation et le monitorage hiérarchiques de tâches. La seconde couche permet la composition de différentes classes de comportements. La dernière couche de cette architecture permet de faire le lien entre la couche des comportements et le matériel du système robotisé (capteurs et actuateurs).La (figure 18) schématise le fonctionnement de l'architecture CAMPOUT.

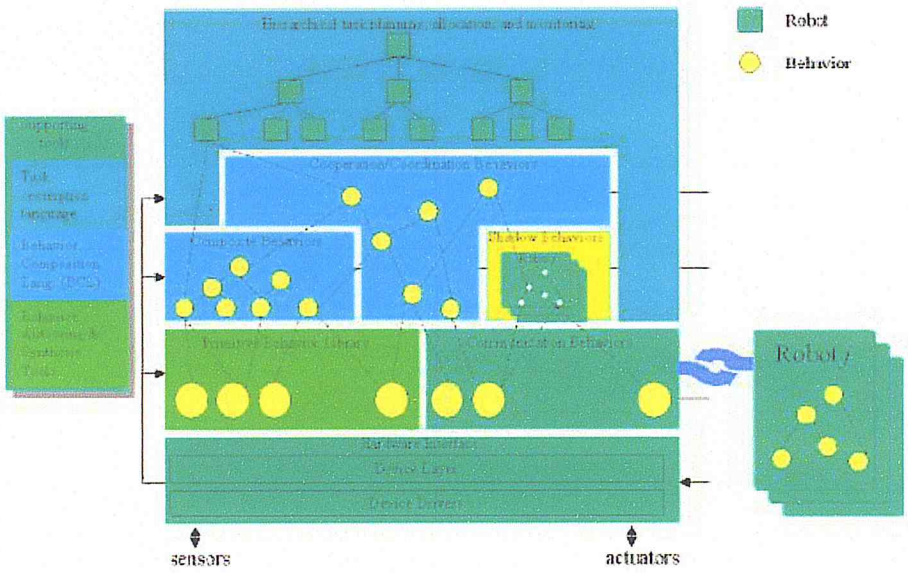



Figure 18 : Schéma résumant l'architecture CAMPOUT [39]

L'architecture étant à la base conçue pour les SMR coopératifs ; elle intègre certains éléments pertinents à ces systèmes, comme une infrastructure de communication (basée sur des sockets TCP/IP). Au niveau des comportements, trois classes sont importantes pour ces systèmes, soient les comportements de communication (Communicatio Behaviors), les comportements images (Shadow Behaviors) qui permettent la représentation des comportements des co-équipiers et finalement, les comportements de coopération/coordination (Cooperation/Coordination Behaviors). Pris individuellement, les robots d'un système peuvent utiliser des comportements primitifs et composés (Primitive Behaviors et Composite Behaviors) pour accomplir leur travail. La gestion de l'utilisation des divers comportements est assurée par la couche supérieure de l'architecture (Hierarchical task planning, allocation and monitoring).

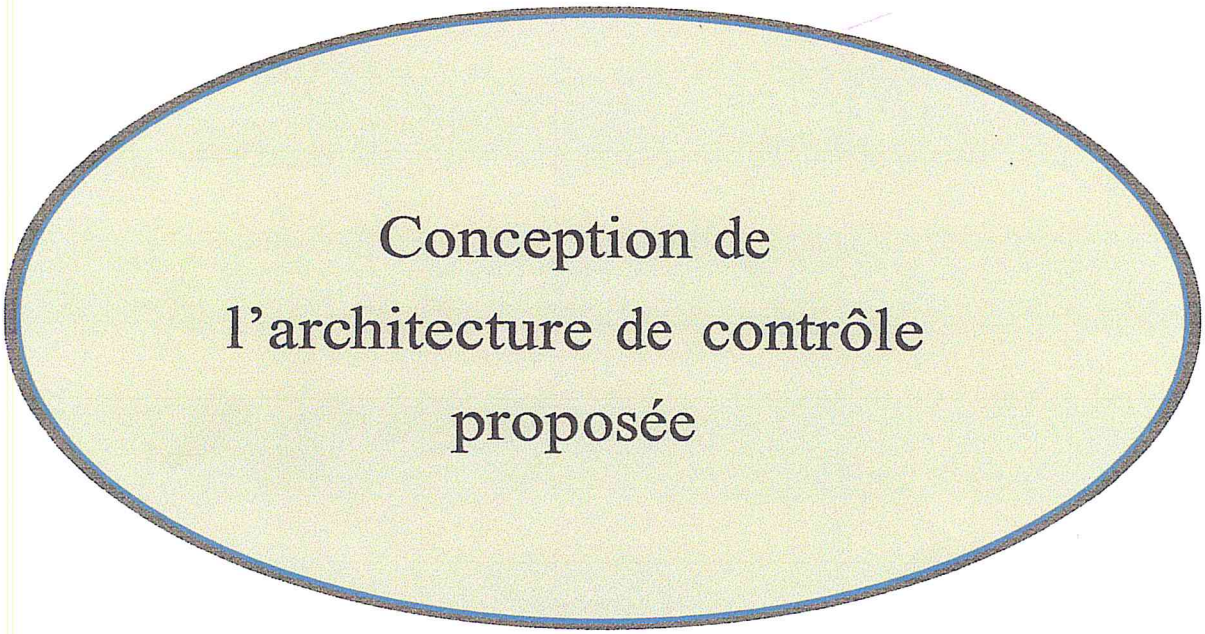
## 4. Conclusion

Les SMR collaboratifs sont des mécanismes suffisants pour pouvoir être utilisés dans le cadre d'applications réelles nécessitant le respect de contraintes opérationnelles fortes. Il existe plusieurs classes pour la mise en œuvre des architectures de contrôle telle que les architectures

Dans ce chapitre, nous nous sommes intéressés aux SMR, nous les avons d'abord définis. Ensuite, nous avons présenté la communication et la collaboration dans un SMR, ainsi que les différents aspects liés à la coopération dans un SMR. Enfin les différentes classes d'architectures de contrôle en robotique en décrivant quelques exemples d'application dans ce domaine.

Dans le prochain chapitre, nous proposerons une architecture de contrôle d'un SMR hétérogène évoluant dans un environnement cyber-physique.

# Chapitre 3:



# Chapitre 3
# Conception de l'architecture de contrôle proposée

## 1. Introduction

Le contrôle d'un SRCP dans lequel évoluent un grand nombre d'entités autonomes réactives est un challenge à la fois scientifique et technologique en plein essor. En effet, ceci exige non seulement d'utiliser des entités robotiques et les hautes technologies de détection et de communication, mais nécessite également au haut niveau du contrôle.

Dans ce chapitre, nous allons faire une description de la dynamique du système. Par la suite, nous ferons une présentation des comportements des agents dans ce système. Après les différentes interactions entre ces agents à l'intérieur du système seront décrites ainsi que la conception de l'architecture de contrôle proposée.

## 2. Objectifs et spécifications

Le rôle général d'une architecture de contrôle d'un système multi-robots est de coordonner les différents éléments du système afin d'effectuer différents types de tâches tout en ayant la capacité de réagir efficacement à différents types d'événements.

Le principal objectif de ce travail est de trouver le meilleur moyen pour décrire une architecture logicielle multi-agents cyber-physique dans laquelle des opérateurs humains, des robots ou des objets quelconques (salles, murs, obstacles, objets, etc.) peuvent s'enregistrer au niveau du contrôleur et échanger des informations avec toutes les entités enregistrées. À la suite de l'enregistrement d'un robot, par exemple, le robot aura tous les autres entités comme des accointances possibles pouvant, ainsi, partager des informations avec eux. Un exemple pourrait être la communication entre les différents robots, entre les robots et le contrôleur, entre les robots et les autres entités, etc. De cette manière, un robot peut accomplir ses tâches, fournir des informations concernant d'autres entités connus ou inconnus qu'il rencontre dans son environnement et qui peuvent être utilisées par d'autres entités du SRCP.

L'architecture proposée doit répondre aux caractéristiques des SRCPs et garantir les propriétés suivantes:

- Intégration : Dans un environnement robotique cyber-physique, plusieurs types de robots ainsi qu'un grand nombre d'entités sont connectés. Donc, il est important de proposer les bons outils et mécanismes de communication pour offrir un échange transparent des données.
- Réactivité en temps réel : Les différents composants de l'architecture doivent être capables de réagir de façon appropriée aux stimuli spécifiques qu'ils reçoivent.
- Robustesse et tolérance aux pannes : Dans un SRCP, il se peut que l'un des composants tombe en panne, soit pour des raisons logicielles (erreurs dans l'implémentation d'un algorithme par exemple) ou matérielles (échec d'un des capteurs ou des tags). Ces raisons sont liées au robot lui-même ou à son environnement. Toutefois, dans tous les cas, l'échec d'un des composants ne doit pas entraîner l'échec de la tâche courante ; l'architecture de contrôle doit donc être capable de détecter et de raisonner sur les différentes causes possibles d'échecs, et de trouver une solution alternative pour achever la tâche.
- Extensibilité : La modularité de l'architecture doit permettre d'ajouter de nouvelles fonctionnalités, sans remettre en cause l'existant.
- Gestion de la concurrence : Dans un environnement cyber-physique, de nombreuses tâches parallèles vont potentiellement accéder et/ou modifier un nombre fini de ressources. Donc, l'architecture de contrôle doit être capable de raisonner sur l'état global courant et sur les tâches en cours afin d'assurer un comportement efficace et cohérent, en particulier en évitant les conflits sur ces différentes ressources.
- Programmabilité : L'architecture de contrôle proposée doit être réalisable et facile à mettre en place.

## 3. Description de l'architecture de contrôle proposée

### 3.1 .Description de l'environnement

Notre environnement (figure 19) est un environnement dynamique où des tags RFID (ou autre) sont déployés de telle sorte que des opérateurs humains et des objets physiques (murs, salles, bureaux, etc.) étiquetés soient en mesure de communiquer avec des robots hétérogènes. Des lecteurs RFID (ou autre) sont aussi déployés sur les robots afin de récupérer les données utiles stockées dans les différents tags.

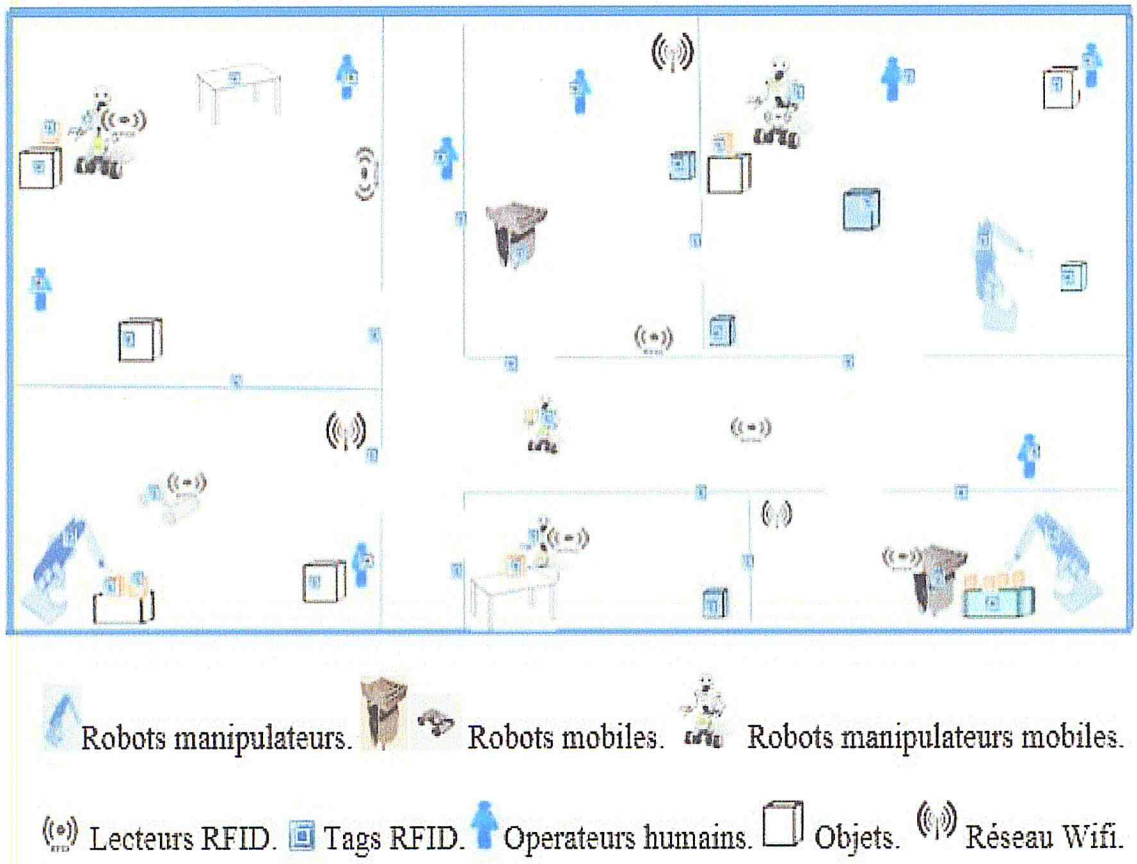



Figure 19: Vue globale de l'environnement.

Les entités physiques de l'environnement sont classées en deux types :

- Entités passives : ce sont les entités qui ne peuvent pas agir sur l'environnement. ces entités ne peuvent pas changer leurs états que par une intervention d'une autre entité active. Ce type regroupe, par exemple, les tables, les bureaux, les chaises, etc.
- Entités actives : Ces entités peuvent agir sur l'environnement. Ce type regroupe les opérateurs humains et les robots. Ces derniers, sont classés, selon les différentes tâches qu'ils peuvent réaliser, en trois catégories :
  - Robots Manipulateurs : robots fixés physiquement à leurs emplacements de travail et généralement mis en place pour réaliser des tâches précises ou répétitives.
  - Robots Mobiles : robots capables de se déplacer ; leurs rôle principal est la transportation des objets.
  - Robots Manipulateurs Mobiles : ce sont des robots mobiles surmontés d'un ou de plusieurs manipulateurs.

### 3.2 .Architecture de contrôle proposée

Notre choix c'est porté sur une architecture de contrôle qui englobe un grand nombre de types de traitements et de contrôle. En effet, l'architecture de contrôle proposée est hybride, elle représente le comportement des deux types d'architectures existants dans la littérature (centralisée et distribuée). Pour ce faire, l'architecture de contrôle proposée se base principalement sur deux stratégies :

- La hiérarchisation des couches en plusieurs niveaux de traitement de l'information.
- L'utilisation et l'adaptation des SMA dans un contexte robotique cyber-physique.

Notre architecture est composée alors de deux niveaux tel que montré par la (figure 20):

- Couche physique : elle représente toutes les composantes physiques de l'environnement (les robots, les différentes entités actives et les entités passives). Ce niveau communique avec le niveau contrôle, par envoi de données et par réception de commande.
- Couche de contrôle: elle joue le rôle d'intermédiaire entre l'extérieur (utilisateurs) et le système.

  Il est destiné à contrôler et à gérer la couche physique, grâce à la coordination et la négociation. Cette couche communique avec la couche physique, par envoi des requêtes et par réception des informations et des comptes rendus.

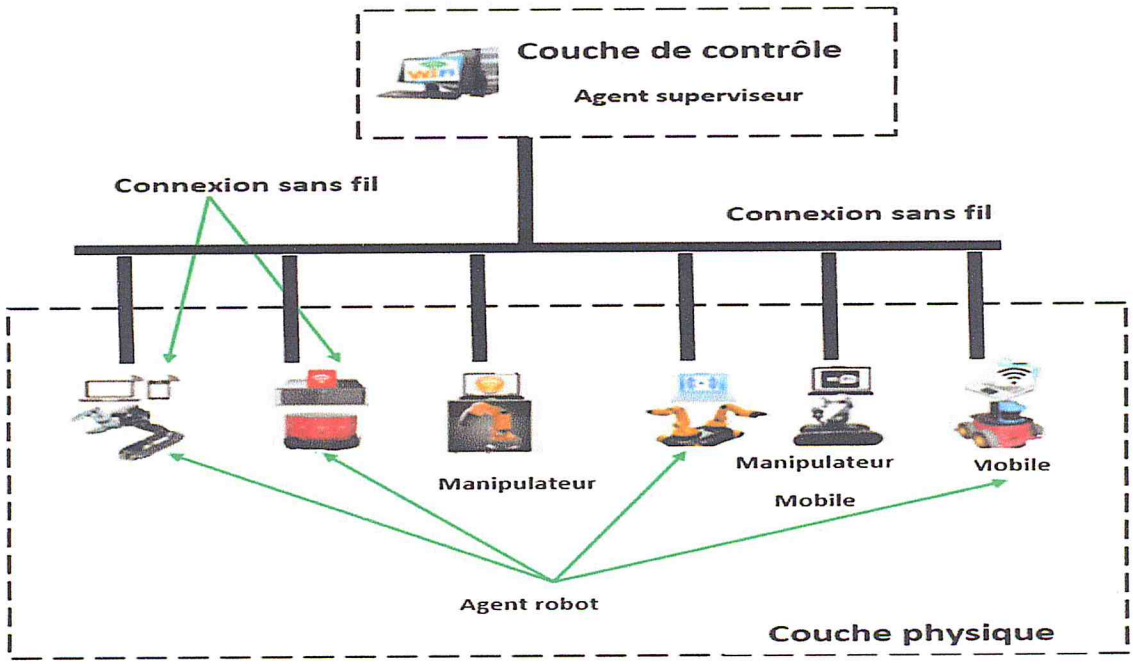



Figure 20: Architecture globale du système multi-agents de contrôle d'un SRCP.

L'architecture proposée est composée de deux types d'agents répartis sur les deux niveaux (décrits précédemment) et qui sont connectés par un réseau local (sans fil ou autre). Nous distinguons au sommet, un agent Superviseur. En outre, dans le deuxième niveau de l'architecture, nous retrouvons pour chaque robot, un agent dédié à l'exécution des consignes envoyées par l'agent Superviseur.

L'agent Superviseur traite les informations collectées de l'extérieur ; ensuite, il les envoie aux agents de niveau physique, afin de sélectionner les robots les plus appropriés à exécuter les opérations. Ces derniers communiquent et coordonnent leurs opérations afin d'accomplir la tâche assignée.

Le diagramme d'agents de la figure suivante résume la relation entre les différents agents du système de contrôle :

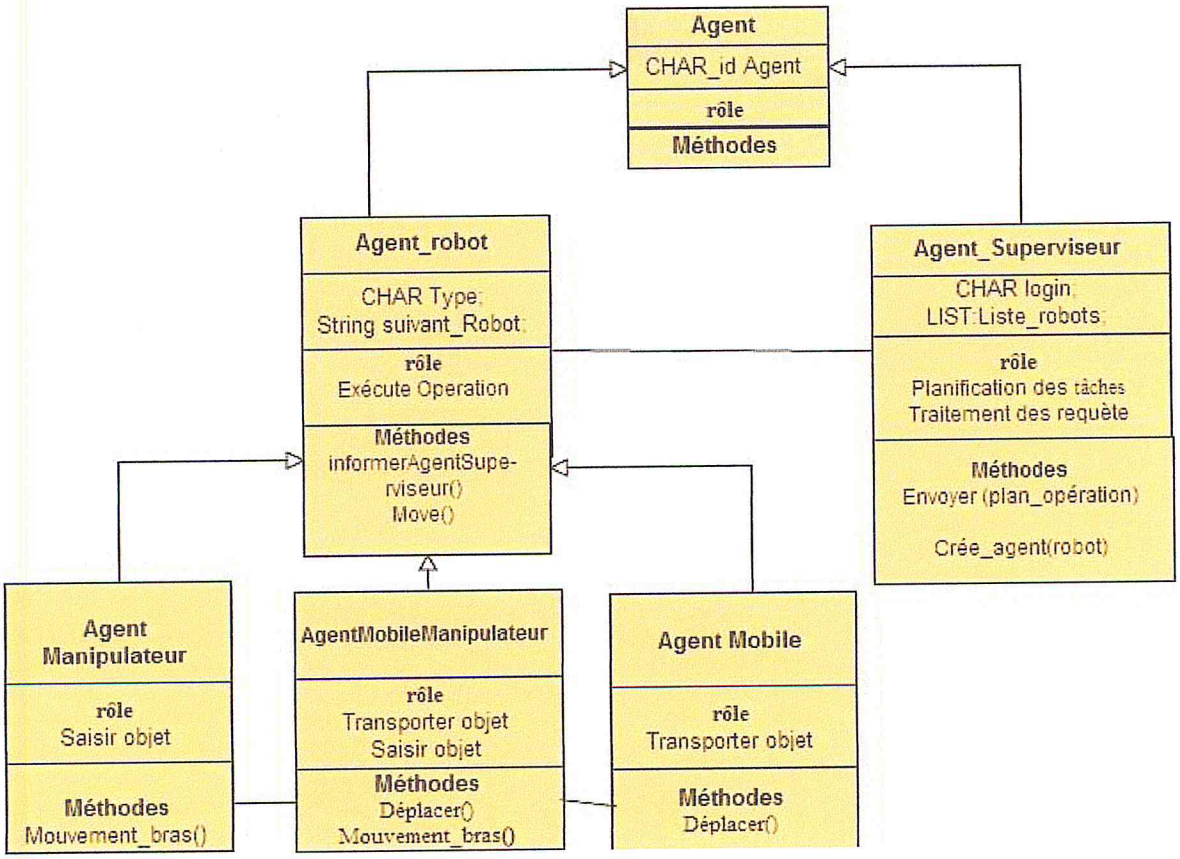



Figure 21: Diagramme d'agents du système de contrôle proposé.

Dans ce qui suit, nous allons procéder à la spécification des différents agents et leurs rôles dans l'architecture proposée. Nous allons donner aussi les différents modules et interactions qui existent entre eux.

### 3.2.1. Spécification des agents

#### 3.2.1.1. Agent Superviseur

C'est un agent hybride, son rôle est de configurer le système et planifier les tâches. Aussi, il participe dans le processus de prise de décision. Cet agent, qui est implémenté sur un PC hôte, permet aux opérateurs humains de gérer et d'interagir avec l'architecture de contrôle. En outre, l'agent Superviseur reçoit des tâches à exécuter, les décompose en opérations primitives et ordonnance ces opérations ; puis, cet agent les alloue, et les envoie aux agents robots du niveau physique. L'agent Superviseur regroupe les modules suivants (figure 22) :

- **Gestion des utilisateurs** : Ce module est chargé de la gestion des utilisateurs telle que l'attribution des droits d'accès, etc.
- **Interface utilisateur** : Ce module permet de l'interaction avec l'utilisateur. Il permet de récupérer les requêtes des utilisateurs ; puis, les envoyer au module Gestion des tâches. Ce module est, aussi, le responsable d'afficher les résultats obtenues aux utilisateurs.
- **Base de Données** : Ce module se compose deux parties : les données individuelles et les données sociales. Les données individuelles reflètent la vue qu'a l'agent de lui-même, notamment, son nom, son adresse, ses objectifs individuels, ses protocoles de décision et ses différents états possibles. Les données sociales reflètent la représentation qu'a l'agent de l'environnement dans lequel il évolue. Ces données particulières portent sur les agents qu'il peut contacter (Agent ID, Adresse IP, Ports d'envoi, Ports de réception, etc.), sur les protocoles de communication ou d'interaction à utiliser, ainsi que les informations sur tous les objets de l'environnement. En effet, cette base de données englobe toutes les bases de données des autres agents du système.
- **Gestion des tâches** : Ce module décide de l'acceptation ou du refus de la tâche reçue. Si elle est acceptée, il l'envoie au module de planification ou au module de configuration selon le type de la tâche (exécution d'une tâche, demande d'intégration, etc.).
- **Planification** : Ce module divise la tâche en opérations primitives et construit un plan d'opérations. Ce plan sera envoyé au module de Traitement.
- **Traitement** : Ce module s'occupe de l'analyse des plans d'opérations reçus, de la vérification de l'état de ses agents Robots, de leurs capacités à réaliser ces opérations afin de choisir les robots qui vont exécuter ces opérations.

- **Configuration** : Ce module est le responsable de l'ajout ou de la suppression des robots et des objets de l'environnement de tels sorte qu'il garantisse l'intégrité de système.
- **Communication interne** : Ce module est responsable de l'interaction de l'agent avec les autres agents du système pour envoyer et recevoir des informations. Il regroupe tous les sous-modules nécessaires pour coder, décoder et analyser les messages reçus.
- **Communication externe** : Ce module est responsable de l'interaction de l'agent avec les nouveaux robots qui veulent s'intégrer dans le système.
- **Diagnostic et Gestion des pannes** : il est chargé de contrôler l'état (en panne, non panne) de l'agent lui-même, de vérifier l'état du système (en panne, non panne) et de réagir en cas de panne.
- **Module Gestion des objets** : Ce module se charge de la gestion des objets, par exemple si l'objet est déjà utilisé par d'autres robots ou pas.

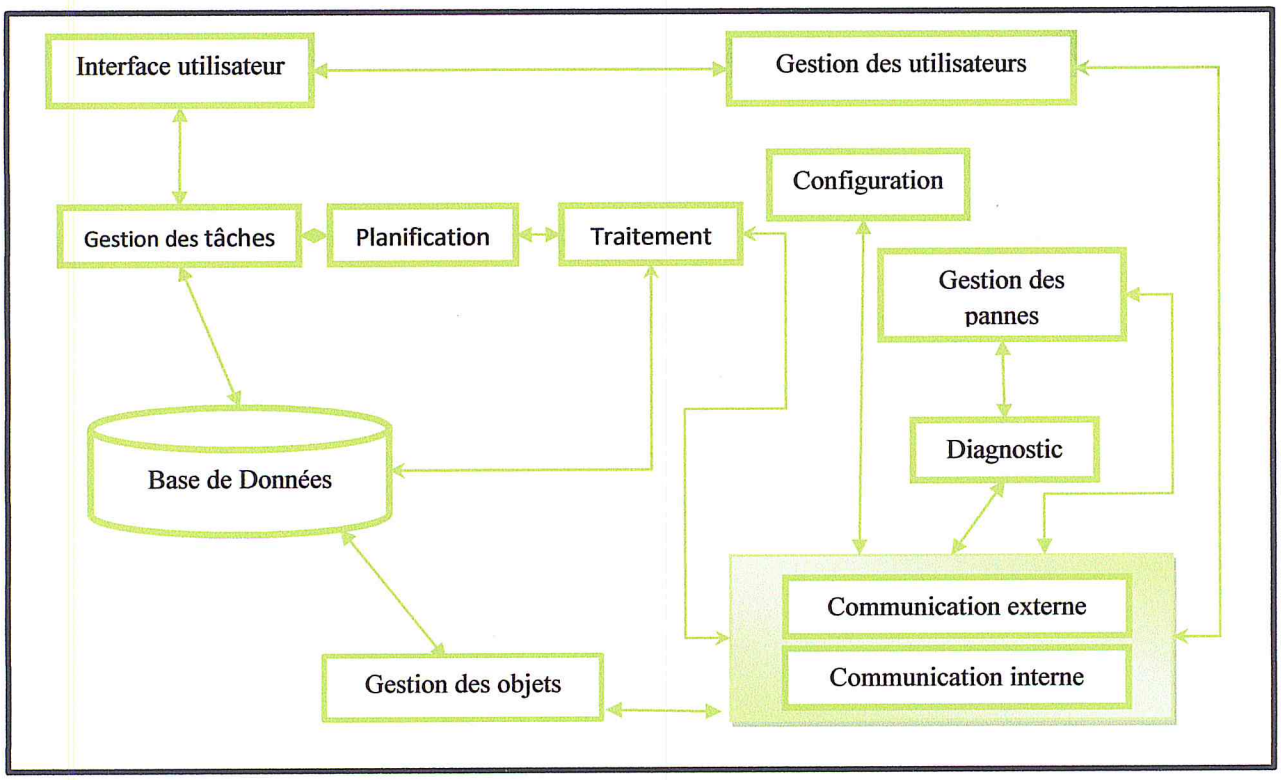



Figure 22: Structure de l'agent Superviseur.

### 3.2.1.2 .Agents Robots

L'agent Robot est un agent hybride ; son objectif est d'exécuter les consignes envoyées par l'agent Superviseur. Il comprend les modules suivants (figure 23):

- **Base de Données** : Chaque agent Robot est doté d'une base de données qui contient des informations sur lui-même et sur les autres agents du système.
- **Perception :** Ce module permet à l'agent Robot, d'une part, de percevoir l'état de son environnement en collectant toutes les informations en provenance des capteurs équipant le robot et, d'autre part, de prétraiter ces informations afin d'en extraire les informations utiles.
- **Contrôle** : Ce module permet au robot d'agir sur son environnement grâce à ces actuateurs.
- **Génération de trajectoires** : Il établit des trajectoires tout en tenant compte de l'environnement du robot par la mise en place de stratégie de navigation et d'évitement d'obstacles, de suivi de cibles, etc.
- **Surveillance** : Lorsqu'un incident de dysfonctionnement se produit, ce module réalise un diagnostic afin de localiser la source de l'anomalie. Le résultat obtenu est envoyé à l'agent Superviseur.
- **Communication** : Ce module se charge de la communication entre cet agent et les autres agents de système.

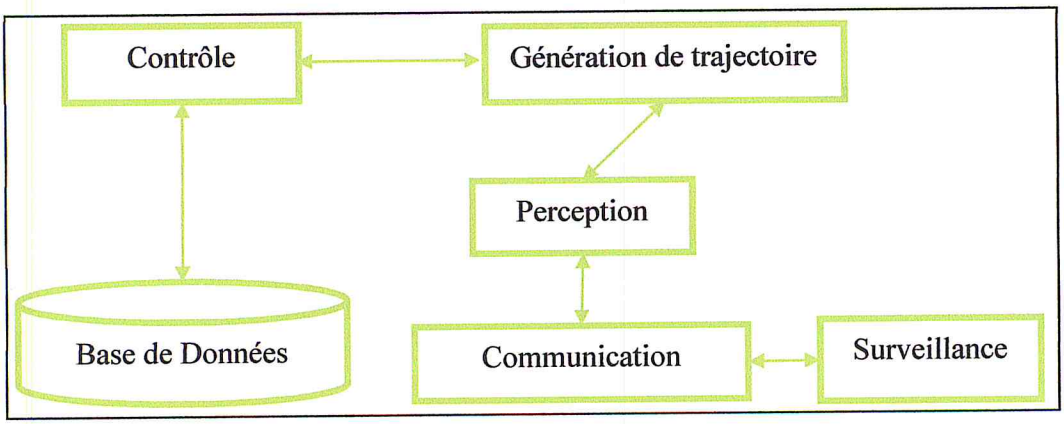



Figure 23: Structure de l'agent Robot.

### 3.3 .Interactions entre les agents du système

Après avoir détaillé chaque agent du système, nous allons décrire dans cette section les différents comportements du système sous forme d'algorithmes. Notre système se compose d'une interface qui permet aux utilisateurs, après authentification de lancer des demandes pour réaliser des tâches. Le fonctionnement général du système de contrôle est donné comme suit :

### 3.3.1. Agent Superviseur

Au lancement de cet agent, les modules Diagnostic et Gestion des pannes commencent à vérifier l'état des agents robots :

```
Algorithme 1_Diagnostic_Agent_Superviseur{
    if (dysfonctionnement) {
        Rapport_panne=Rédiger_rapport_panne();
        /*Cette fonction permet de réaliser un compte rendu sur la panne détectée au
        niveau des robots*/
        Envoyer(Rapport_panne, Gestion_pannes) ;
        // Envoyer le rapport de panne au Module Gestion des pannes
    }
}
```

#### 3.3.1.1 .Réception d'un rapport de panne

À la réception d'un rapport de panne, il sera analysé par le module Gestion des pannes en exécutant l'algorithme suivant :

```
Algorithme 2_Gestion_pannes_Agent_Superviseur{
    Entrées : Rapport_Panne;
    Sorties : Réponse ;
    switch (Type_Panne) {
        case " Logiciel"{
            Créer(Nouvel_Agent_robot) ;
        //Création d'un nouvel agent pour remplacer l'agent robot en panne*/
            break ;
        }
        case " Matériel"{
            Appeler(Maintenance);
            //Appeler le service de maintenance pour réparer la panne
        }
    }
}
```

### 3.3.1.2 .Réception d'une requête d'intégration envoyée par un nouveau robot

Quand un nouveau robot veut s'intégrer au système, il diffuse une demande d'intégration aux agents. Les agents qui ne sont pas concernés par cette demande la retransmettent à l'agent Superviseur. Le module Communication externe de cet agent ne considère que la première demande d'intégration ; il ignore les autres. Après, il l'envoie au module Configuration. Ce dernier traite cette demande selon l'algorithme ci-dessous :

```
Algorithme 3_Configuration_Nouveau_Robot_Agent_Superviseur{
        Entrées : Demande_Intégration_Robot;
        Sorties : Réponse ;
        Réponse = Traitement (Demande_Intégration_Robot) ;
        /*Cette fonction permet de traiter la demande et détermine si le robot est accepté ou
        pas selon des critères prédéterminés */
        if (Réponse == Accepter){
                Envoyer(Demande_Intégration, Module_configuration);}
                Informer(Réponse, Nouveau Robot);
        }
}
```

### 3.3.1.3 .Réception d'une requête envoyée par l'utilisateur

À la réception d'une requête envoyée par l'utilisateur, le module Interface utilisateur récupère les données de l'utilisateur et le contenu de la requête :

- Les informations de l'utilisateur seront envoyées au module Gestion des utilisateurs.
- Le contenu de la requête sera envoyé au module Gestion de tâches qui exécutera l'algorithme suivant :

```
Algorithme 4_Gestion_tâches_Agent_Superviseur{
        Entrées : Requête;
        Sorties : Réponse ;
        while (!end) {
                Réponse=!Accepter ;
                switch (Requête) {
                        case "Intégration_Nouveau_Robot":
```

```
                    if(!existe(nouveau_robot)) {
                        Réponse =Accepter
                        Envoyer (Requête, Module Configuration);
                    }
                    break;
case "Suppression_ Robot":
    if(existe (robot)) {
        Réponse=Accepter ;
        Envoyer(Requête, Module Configuration);
    }
    break;
case "Intégration_Nouveau_Objet":
    if(!existe(Objet)) {
        Réponse = Accepter ;
        Envoyer (Requête, Module Configuration);
    }
    break;
case "Suppression_ Objet":
    if(existe(Objet)) {
        Réponse = Accepter ;
        Envoyer (Requête, Module Configuration);
    }
    break;
case "Exécution_tâche": {
    if (tâche=''deplacer_Objet'') {
        if (Verifier (Emplacement_Final)){
    //vérifier si l'emplacement final donné par l'utilisateur est libre
            Envoyer (Requête, Module Planification);
            Réponse=Accepter ;
        }
        else
            Informer(Utilisateur, "Emplacement occupé");
            //informer l'utilisateur que l'emplacement est occupé
```

```
                    }
                }
                else
                    if(tâche=''Leader/Followers'')
                        Envoyer(Requete, Module Traitement)
                        Réponse= Accepter ;
                        break;
                    }
                }
            }
}
```

À la réception d'une tâche, le module Planification exécute l'algorithme suivant :

```
Algorithme 5_Module_Planification_Agent_Superviseur{
    Entrées : Tâche_à_Exécuter ;
    Sorties : Plan_Opérations ;
    Plan_Opérations=Planification(Tâche_à_Exécuter); /* décomposer la tâche selon sa
    complexité en un ensemble d'opérations primitives */
    Envoyer(Plan_Opérations, Module Traitement);
}
```

À la fin de planification d'une tâche, le module Traitement exécute l'algorithme suivant :

```
Algorithm 6_Traitement_ Plan_Opérations _Agent_Superviseur{
    Entrées : Plan_Opérations ;
    Sorties : Réponse ;
    while(!end) {
        switch (Requête) {
            case "Intégration_Nouveau_Robot" : {
                Creer_Agent_Robot();
        //Créer un Agent Robot et l'affecter au nouveau robot
                Diffuser_Intégration_Nouveau_Robot(id, @ip, position);
```

```
//informer les agents du système qu'un nouveau robot a été intégré
        }
        break ;
        case "Suppression_d'un_Robot" : {
                Supp_Agent_Robot(); // Supprimer l'Agent Robot.
                Diffuser_Suppression_Robot(id, @ip, position);
//informer les agents du système qu'un robot a été retiré
        }
        break ;
        case "Intégration_Nouveau_Objet":{
                Insérer (Objet, BDD);
                Diffuser_Intégration_Nouveau_Objet(id, Caractéristiques);
//informer les agents du système qu'un nouveau Objet a été intégré
        }
        break;
        case "Suppression_ Objet":{
                Supprimer (Objet, BDD);
                Diffuser_Suppression_Objet(id);
//informer les agents du système qu'un nouveau Objet a été retiré
        }
        break;
        case "Plan_Opération": {
                Caractérstique_Objets= Charger_Caractérstique (Objets, BDD);
                //Cette fonction permet de récupérer les caractéristiques des
                objets utilisés.
                List_robot_tâche ;
                /*Liste des robots qui vont exécuter la tâche.
                For(i=1; i<=Nbr_Opérations; i++){
                        List_Robot_tâche.Ajouter(Choisir(Opération,
                        Caractérstique_Objets);
//Cette fonction permet de choisir le meilleur robot pour exécuter l'opération
                }
                Ordoonancer(list_robot_tâche);
```

```
//cette fonction permet de choisir le meilleur plan d'exécution de la tache
                    For(i=1; i<=Size(liste_robot_tache); i++){
                        Envoyer(Opération, Robot, Ordre);
                    }
                }
                case ''Leader/ Followers'':{
                    Trier(list_agent_robot)
/* cette fonction permet de trier les robots selon des critères prédéfinis (distance à destination, capacités, etc.).*/
                    Rapprochement(list_robot) ;
//cette fonction permet le rapprochement des robots les uns aux autres
                    Envoyer(Destination, Leader) ;
//cette fonction permet d'envoyer la destination au Leader
                    while(! fin _Leader/followers ){
                        Ping= attend(Leader, temp) ;
                        if (!Ping) Alors{ // le Leader tombé en panne
                            Resoudre(Panne_Leader);
                        }
/* cette fonction permet soit de réparer le leader lui-même soit de le remplacer par un autre (robot suivant)*/
                    }
                }
            }
        }
}
```

À la fin de traitement d'une telle opération (exécution d'une tâche, intégration ou suppression d'un robot, intégration ou suppression d'un objet, Leader/Follower, ect), le module Interface utilisateur reçoit un compte rendu envoyé par le module Traitement, afin d'informer l'utilisateur du succès ou d'échec de la tâche demandée.

```
Algorithme 7_Traitement_Comptes_rendu_Agent_Supervieur{
    Entrées : Compte_Rendu ;
```

```
Sorties : Compte_Rendu_final ;
Résultat=Analyser(Compte_Rendu);
/* Cette fonction permet de vérifier l'état de l'opération qui a été exécutée avec succès ou non. */
switch (Résultat){
        case(échec_opération){
                Robot=Choisir_autre_robot(Compte_Rendu);
/*Permet à l'agent Superviseur de sélectionner un autre robot pour suivre l'exécution de l'opération*/
                if (Existe(Solution))
                Informer(Robots);
/*informer les robots concernés de l'exécution des opérations suivants de cette tâche que l'opération a été allouée à un nouveau robot*/
                else{ //pas de solution
                        Informer(List_Robot, List_Opération_Restants);
/*informer les robots concernés de l'exécution des prochaines opérations que la tâche est annulée*/
                        Informer (compt_rendu_final, User);
                        Mettre_à_jour(Etat_Robots, Libre);
/*Mettre à jour l'état des robots concernés de l'exécution des prochaines opérations dans la BDD*/
                }
        }
        case (échec_suivant_robot){
                Réordonnancer_robot();
/* cette fonction permet de réordonnée les robots afin de retiré le robot en panne */
                if(suivant(robot.suivant))
                        Envoyer(suivant(robot.suivant), Position suivante);
/*S'il existe un autre robot après le robot en panne, le superviseur va envoyer une nouvel position au robot qui suit celui en panne dans le but de l'éviter*/
        }
        case(succès){
                Mise_à_jour(BDD) ;
```

```
                    Envoyer(Compte_rendu, User) ;
                }
        }
}
```

### 3.3.2. Agents Robots

Les agents robots coopèrent entre eux pour le succès de la tâche. À cet effet, chaque agent Robot exécute l'algorithme suivant :

```
Algorithme 8_Exécution_Opérations_Agents_Robots{
        Entrées : Ordre_Exécution, Plan_Exécution;
        Sorties : Compte_Rendu;
        Exécuter (opération);
        if (Succès(Opération)) { //l'opération a été réalisée avec succès.
                Éditer(Compte_Rendu, ''Succès '');
                Envoyer("fin_exécution_opération", Agent_Robot_Opération_Suivante);
                Reponse= attend (Agent_Robot_Opération_Suivante,Temp) ;
                  If( !Reponse){
                Envoyé(Superviseur ,Agent_Robot_Opération_Suivante) }
        // informer le robot qui va exécuter la tâche suivante.
        }
        else{ // Échec de l'opération en cours.
                Éditer(Compte_Rendu, Échec);
                // Ce rapport contient des informations sur l'état de l'opération.
                Envoyer(Compte_Rendu, Superviseur);
        }
}
```

## 4. Conception de l'architecture proposée

Cette phase consiste à enrichir la description, de détails d'implémentation afin d'aboutir à une description très proche d'un programme. Nous allons modéliser toute l'architecture en diagramme de cas d'utilisation et de classes.

### 4.1 Diagramme de cas d'utilisation

Un diagramme de cas d'utilisation permet de représenter graphiquement les cas d'utilisation et d'identifient les fonctionnalités fournies par le système (cas d'utilisation), les utilisateurs qui interagissent avec le système (acteurs), et les interactions entre ces derniers. La figure suivant représente le diagramme de cas d'utilisation de notre architecture proposée :

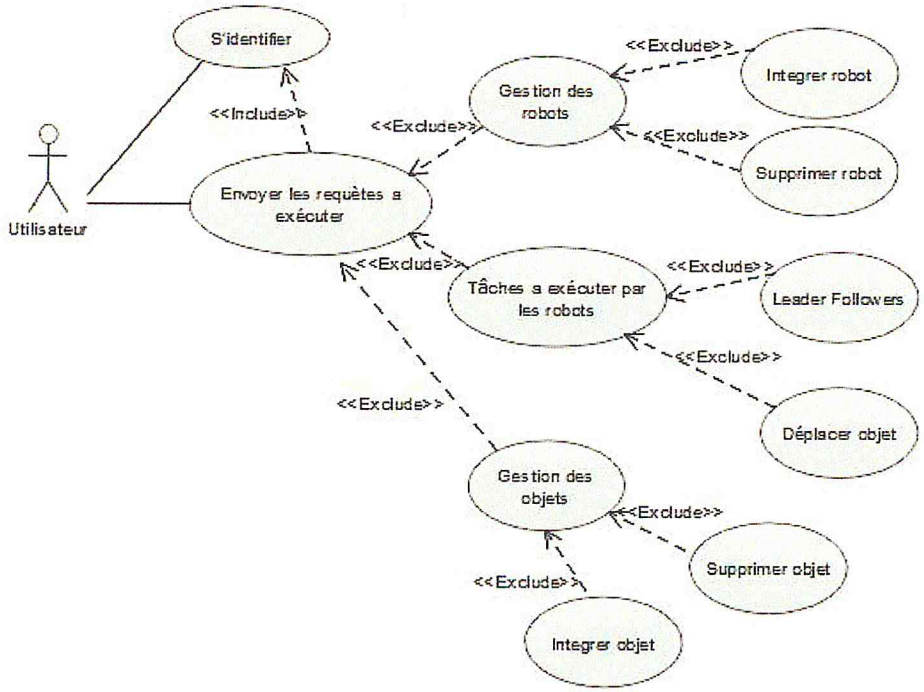



Figure 24:Diagramme de cas d'utilisation.

### 4.2. Diagramme de classes

Le diagramme de classes constitue un élément très important de la modélisation. Il permet de définir quelles seront les composantes du système de contrôle final. Il ne permet en revanche pas de définir le nombre et l'état des instances individuelles. Néanmoins, on constate souvent qu'un diagramme de classes proprement réalisé permet de structurer le travail de développement de manière très efficace. Le diagramme de classes représenté dans la figure suivante décrit les associations entre les classes et ceci afin de déterminer les dépendances entre les différentes classes.

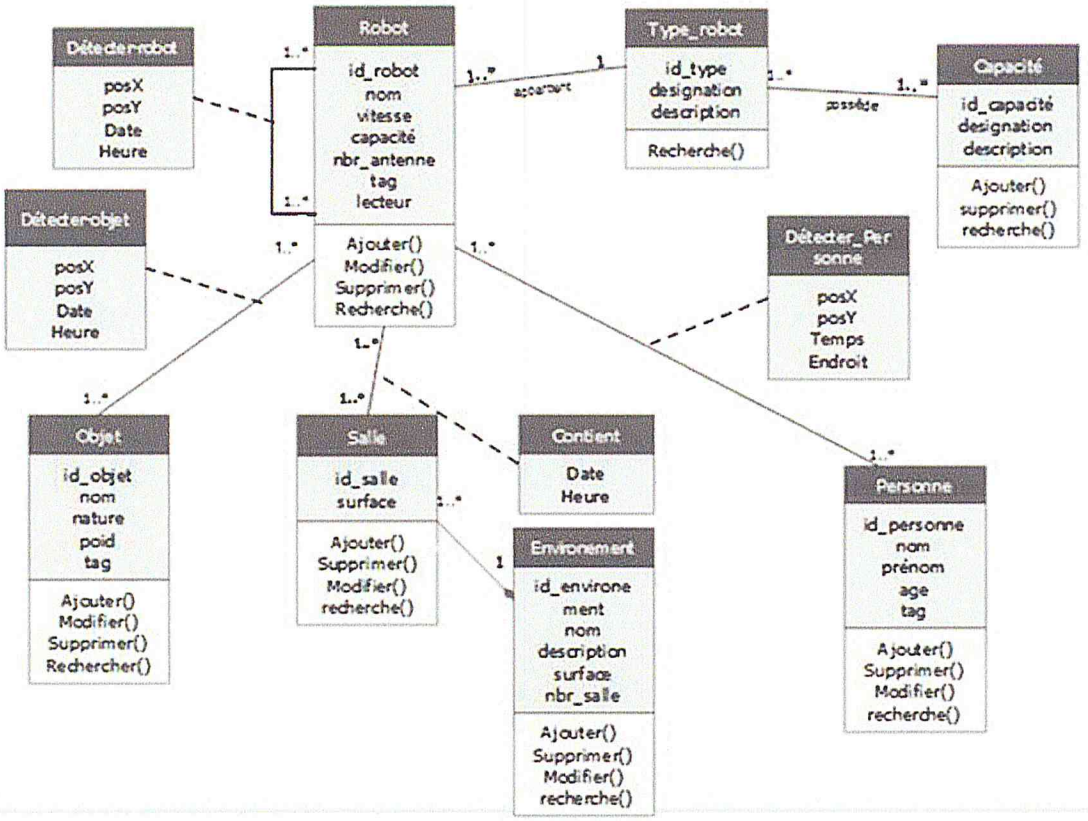



Figure 25: Diagramme de classes

## 5. Conclusion

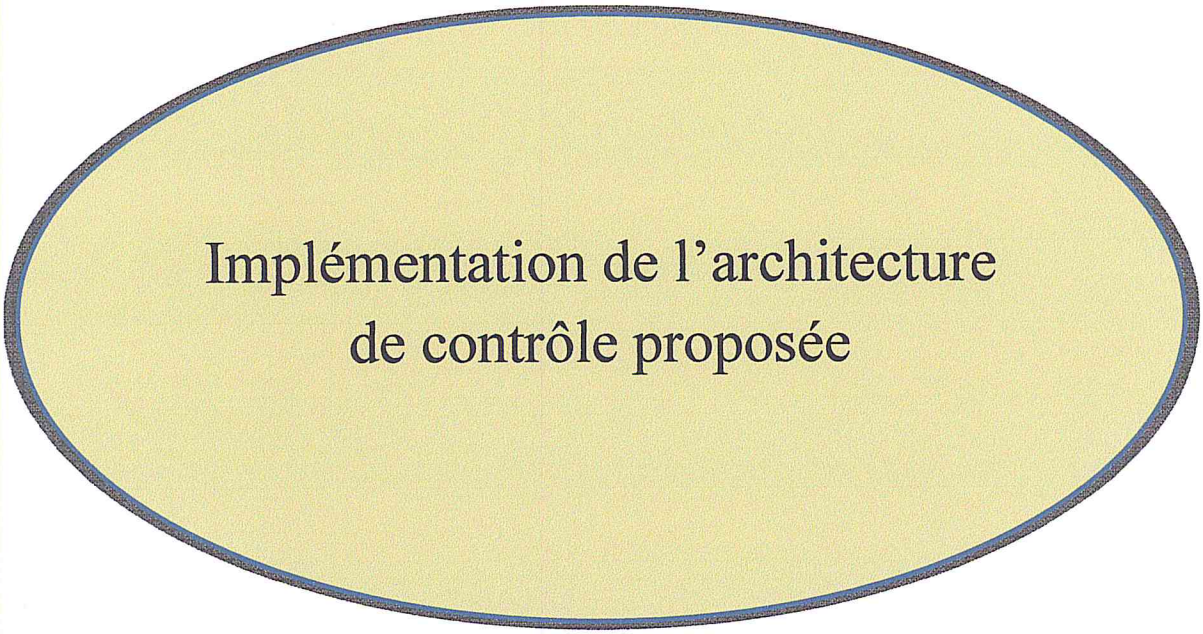
Dans ce chapitre, nous avons défini les différentes entités composant le système, les algorithmes qui s'exécutent au niveau de chaque agent, les interactions entre eux ainsi la conception de l'architecture proposée.

La particularité de cette architecture de contrôle réside en sa hiérarchisation et surtout en les méthodes utilisées à chaque niveau, pour permettre une bonne autonomie et une bonne organisation du système.

Grâce à l'efficacité et la souplesse du modèle multi-agents proposé, l'architecture peut être modifiée et étendue très facilement, par exemple en modifiant les algorithmes des différents agents du système ou encore en rajoutant d'autres modules pour améliorer ou s'adapter à l'exécution des tâches, etc.

L'implémentation et l'évaluation du système de contrôle cyber-physique proposé feront l'objet des prochains chapitres.

# Chapitre 4:



# Chapitre 4

# Implémentation de l'architecture de contrôle proposée

## 1. Introduction

Précédemment, nous avons décrit l'architecture logicielle cyber-physique multi-agents de contrôle proposée ainsi que sa modélisation, l'objectif de ce chapitre est l'implémentation de cette architecture. Nous commençons par la description de l'environnement logiciel nécessaire (JADE, Netbeans, SQLite). Après, nous détaillons les différentes étapes de la mise en œuvre de l'architecture de contrôle proposée.

## 2. Environnements de développement

### 2.1. Architecture physique de dépoilement

La mise en œuvre de l'architecture proposée a été réalisé sur un ordinateur doté d'un système d'exploitation (Ubuntu 12.04 LTS/Windows 7) sur lequel est installée le Moteur OpenJDK Java 8, la plateforme Java Agent DEvelopment Framework (JADE) et le système de gestion de bases de données SQLite. La manière dont les composants physiques du système sont organisés est illustrée dans le diagramme de dépoilement suivant :

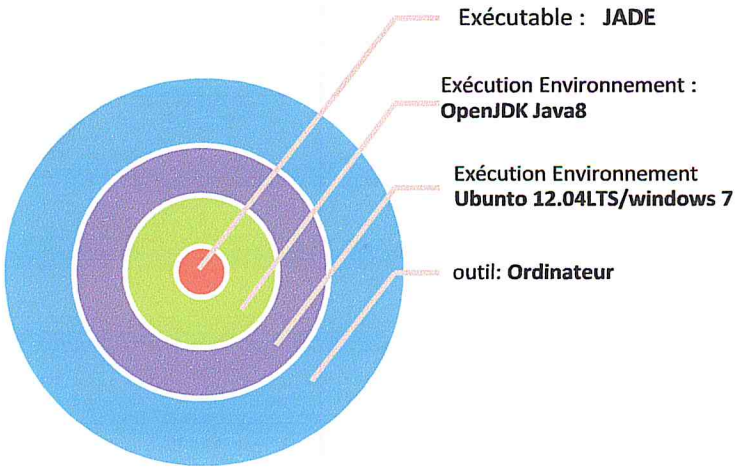



Figure 26: Diagramme de déploiement du système.

## 2.2. Outils et langages de programmation

### 2.2.1. Plateforme JADE

Il existe plusieurs plateformes pour implémenter les SMA (JADE, CORMAS, JACK). Ces plateformes peuvent être utilisées pour analyser, créer ou bien tester les SMA. Dans le cadre de notre travail, nous avons opté pour l'utilisation de la plateforme JADE (Java Agent DEvelopment Framework), vu ses avantages que nous résumons dans ce qui suit [40] :

- Simplifier la construction des SMA interopérables.
- Fonctionner sous tous les systèmes d'exploitation.
- Faciliter la communication des agents JADE avec des agents non JADE.

JADE est une plateforme multi-agents développée en Java par CSELT (Groupe de recherche de Gruppo Telecom, Italie). Elle possède trois modules principaux (nécessaires aux normes de la FIPA (Foundation for Intelligent Physical Agents) qui sont activés à chaque démarrage de la plateforme [40] :

- **AMS (Agent Management System)** : c'est en quelque sorte le cœur de la plateforme FIPA. Il enregistre les agents actifs, gère leurs identités et garde la trace de leurs états.
- **DF (Directory Facilitator)** : c'est un service d'annuaire permettant d'identifier les services utilisateurs sur une plateforme.
- **ACC (Agent Communication Channel)** : c'est un agent particulier chargé de contrôler les messages entres les différents agents issus d'une plateforme FIPA (ou encore non FIPA) éventuellement distantes.

JADE est composée de plusieurs containers (réceptacles) d'agents (figure 27). La distribution de ces containers à travers un réseau d'ordinateurs est permise. Chaque container d'agents est un environnement multi-threads d'exécution composé d'un thread d'exécution pour chaque agent, en plus des threads créés à l'exécution par le système RMI (Remote Method Invocation) pour envoyer des messages. Un seul container est principal, c'est celui qui contient les agents et la plateforme (AMS, ACC et DF).

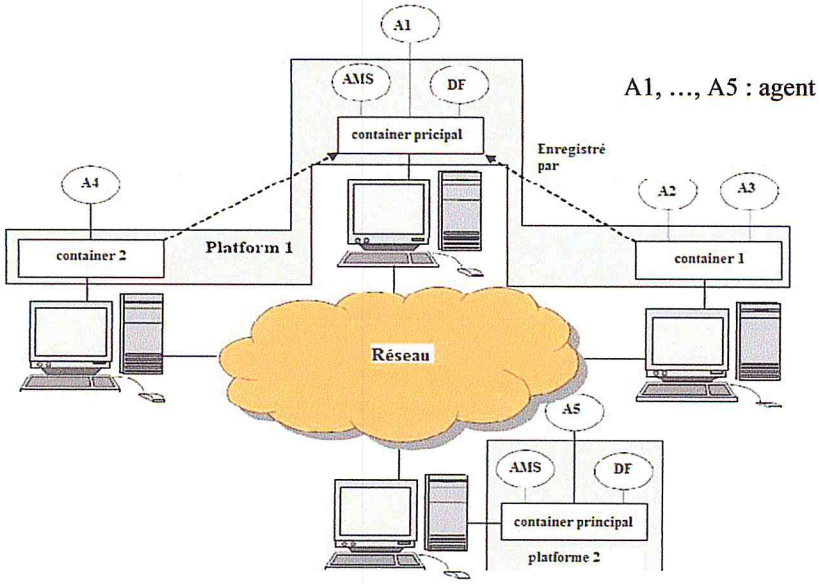



Figure 17: Plateforme et conteneurs de JADE.

La plateforme offre une interface graphique utilisateur GUI (figure 28) pour la gestion à distance des agents. Deux outils graphiques sont disponibles [40] :

- **Agent Dummy :** il qui a pour rôle d'inspecter les échanges de messages entre les agents. Il permet aussi d'éditer, d'écrire, d'envoyer, de recevoir et de sauvegarder des messages en FIPA-ACL.
- **Agent Sniffer :** il consiste en une interface graphique pour afficher les échanges des messages entre les différents groupes d'agents en utilisant une notation proche d'UML.

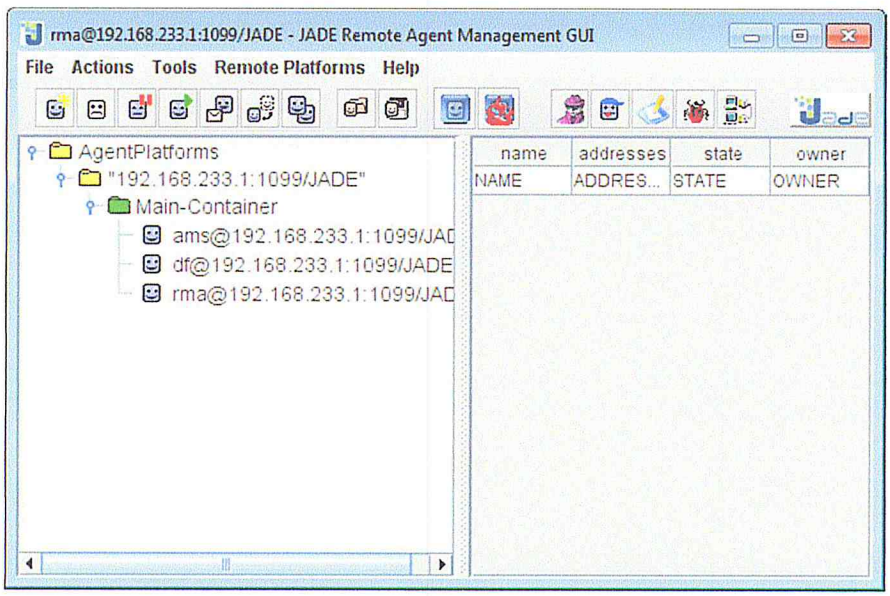



Figure 28: Interface utilisateur (GUI).

### 2.2.2. Langage Java

Afin de réaliser l'interface permettant aux utilisateurs de communiquer avec l'architecture de contrôle, nous avons choisi le langage Java. Ce choix a été motivé par les raisons suivantes:

- Les agents développés sous la plateforme JADE sont entièrement écrits en Java. Ce langage s'est donc imposé comme étant une conséquence de nos précédents choix en termes de plateforme de développement du SMA (JADE).
- Java assure une totale indépendance des applications vis-à-vis de l'environnement d'exécution, c'est-à-dire que toute machine supportant Java est en mesure d'exécuter un programme sans aucune adaptation (ni recompilation, ni paramétrage de variables d'environnement).

### 2.2.3. NetBeans

**NetBeans** est un environnement de développement intégré (EDI), placé par Sun en juin 2000 sous licence CDDL et GPLv2 (Common Development and Distribution License). En plus de Java, NetBeans supporte également de supporter différents autres langages. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages Web).

Conçu en Java, NetBeans est disponible sous Windows et Linux. Il constitue par ailleurs une plateforme qui permet le développement d'applications spécifiques (bibliothèque Swing (Java)).

### 2.2.4. SQLite

**SQLite** est une base de données locale très appréciée car elle fournit une interface SQL tout en offrant une empreinte mémoire très réduite et une rapidité de traitement satisfaisante. En outre, elle appartient au domaine public et tout le monde peut donc l'utiliser. De nombreuses sociétés (Adobe, Apple, Google, Sun, Symbian) et plusieurs projets open-source (Mozilla, PHP, Python) fournissent désormais des produits intégrant SQLite. Ce SGBD disposant d'une interface SQL, son utilisation est assez évidente pour quiconque ayant une expérience avec d'autres SGBDR.

## 3. Mise en œuvre de l'architecture proposée

En utilisant les outils cités précédemment, nous avons abouti à une implémentation composée des classes suivantes :

### 3.1. Classe Agent

La plateforme JADE est composée de « containers » pouvant être distribués sur un réseau. Un container contient des agents et procure tous les services nécessaires à l'exécution des agents. Le Main-Container est créé lors du lancement de la plateforme afin de lancer tous les autres containers.

La création d'un agent (figure 29) se fait par la programmation d'une classe héritée de la classe jade.core.Agent. Cette classe possède la méthode setup () qui sera appelée après instanciation de l'agent par le container. Chaque agent est caractérisé par un identifiant unique dans la plateforme à l'aide de la classe jade.core.AID.

```
Start_Main_Container();
Start_Nouveau_Container();
 try {
    Object []ar=new Object[1];
    ar[0]=Liste.getFirst();
    AgentController sv = Liste.getFirst().createNewAgent("SuperViseur","Type_Agent.SuperViseur_Agent", ar);
    sv.start();

 } catch (StaleProxyException ex) {
    ex.printStackTrace();
 }
```

Figure 29: Création de l'agent Superviseur.

### 3.2 Classe Agent Superviseur

La classe Agent Superviseur traite les informations avec les méthodes suivantes :

- **Leader_Follower()** : cette fonction permet au superviseur de récupère la destination. Ensuite, ordonner les robots selon un critère précis (distance à destination ou autre). Enfin, informer les robots de regrouper à une position à calculer (figure 30).

```
if(option.equals("leader")){
//get data"""
String[] sp = msg.getContent().substring(5).split("#");
 X=(int)Double.parseDouble(sp[0]);
 Y=(int)Double.parseDouble(sp[1]);
 System.out.println("x=="+X+"  y=="+Y);
    if(true){
(Scene.ens_cart[Y][X]).set_image("/Imagee/5.PNG");
resol=false;
 L_L_Follower = List_Lader_Follow(this,X, Y);

ACLMessage message= new ACLMessage(ACLMessage.INFORM_REF);
message.addReceiver(new AID(L_L_Follower.getFirst().getName(), AID.ISLOCALNAME));
message.setContent("aproc"+L_L_Follower.getFirst().getPos_X_int()+"#"+L_L_Follower.getFirst().getPos_Y_int()
send(message);
```

Figure 30: Procédure de la tâche « Leader/Follower ».

- **Communication ()** : Cette méthode envoie les requêtes aux différentes classes en utilisant des messages conformes aux spécifications de la FIPA. Ces messages sont des instances de la classe ACLMessage du package jade.lang.acl (figure 31).

```
ACLMessage message= new ACLMessage(ACLMessage.CFP);
message.addReceiver(new AID(robot.getName(), AID.ISLOCALNAME));
message.setContent("Dista"+X+"#"+Y);

send(message);
```

Figure 31: Transmission des ACLMessage de la classe Agent_Superviseur.

- **Diffuser_Msg()** : cette fonction permet de diffuser les messages pour tous les robots ainsi qu'à l'utilisateur.
- **Gestion_panne_Robots()** : cette méthode reçoit le rapport de panne envoyé par la classe Agent_Robot afin de changer son état dans la BDD et renvoyer la consigne à un autre robot.

### 3.3 .Classe Agent_Robot

Cette classe permet l'exécution de toutes les consignes envoyées par l'agent superviseur et la communication avec les autres Agents_Robots. La classe Agent_robot contient les méthodes suivantes :

- **Move ()** : cette méthode permet au robot d'avancer d'une case (en haut, en bas, à gauche, à droite).

- **Get_pos_suivre ()** : cette méthode permet au robot de reconnaitre la prochaine position à atteindre.
- **Rattrape ()** : cette fonction permet au robot de suivre la trace du robot leader.

```
if(Scene.ens_cart[x][y+1].is_marque() & Scene.ens_cart[x][y+1].est_libre(false)
```

Figure 32 : Vérification la trace du robot leader par un autre robot follower.

### 3.4. Classe Environnement

C'est l'environnement ou évoluent les robots. Il contient une matrice simulant des tags RFID (figure 33).

```
public class Environement extends javax.swing.JFrame {
public int i=35;
public int ii=20;
public Carte [][]ens_carte;
    public Environement() {
        initComponents();
        this.setSize(1200,720);
        this.setResizable(false);
        ens_carte =new Carte[ii][i];
           jPanel1.setLayout(new java.awt.GridLayout(ii, i));
           for (int j = 0; j < ii; j++) {
               for (int k = 0; k < i; k++) {
            Carte c=new Carte(k,j);
            c.setHint("");
            c.set_image("/Imagee/1.PNG");

            ens_carte[j][k]=c;
            jPanel1.add(c);
             }
        }
```

Figure 33: constriction de l'environnement

### 3.5. Classe Interface Utilisateur

Pour assurer la communication entre les utilisateurs et le SMA de contrôle, nous avons créés une classe Interface_Utilisateur qui offre un ensemble de fonctionnalités (Authentification, Intégration de nouveaux objets, ...). Les informations saisies par l'utilisateur seront envoyées à la classe Agent_Superviseur en utilisant des Acl_Message (figure 34).

```
    public  static  void Start_Container_User(String Ip){
    try {
    Runtime runtime=Runtime.instance();
    ProfileImpl profileImpl=new ProfileImpl(false);
    profileImpl.setParameter(ProfileImpl.MAIN_HOST,Ip);
      agentContainer_User=runtime.createAgentContainer(profileImpl);
    }
    catch(Exception e){e.getStackTrace();}

    }
Start_Container_User("192.168.233.1");
 try {
     agentContainer_User.createNewAgent("User_Agent","interface_user.Agent_User", null).start();
 } catch (StaleProxyException ex) {
     Logger.getLogger(Interface_User.class.getName()).log(Level.SEVERE, null, ex);
 }
```

Figure 34 : Envoi des informations utilisateur via des ACL_Message.

## 4. Échange de messages

Les agents Jade sont réunis pour assurer les tâches qui leurs ont été affectées. Chaque agent est censé réaliser un traitement propre à lui et d'une manière totalement indépendante. Cela n'est garanti qu'avec une exécution autonome pour chaque agent. Par conséquent, un comportement global, permettant la réalisation des tâches complexes, émerge de l'interaction entre les différents agents. Dans Jade, cette interaction est réalisée via un modèle de communication basé sur un échange de messages. Ces messages ont un format précis, spécifie par le langage ACL (Agent Communication Language) défini par FIPA [41] et qui représente un langage de communication entre agents. Les principales données circulées dans chaque message de communication sont :

- L'émetteur du message que le récepteur du message.
- Le type de message, qui représente aussi sa nature et son objectif.
- Le contenu du message, qui représente l'information circulée à travers ce message.

Les types de message les plus usuels sont INFORM, PROPOSE, CFP (Call For Proposal), etc. Une combinaison de cette liste nous permettra de spécifier une (des) séquence(s) prédéfinie(s) de message échangés entre les agents lors d'une communication. Notre protocole de communication est défini comme suit :

- REQUEST : l'utilisateur envoie une demande d'exécution d'une tâche.
- CFP : le superviseur diffuse une demande que l'utilisateur a envoyée.
- PROPOSE : les agents robots transmettent au superviseur leur proposition.
- INFORM : envoyer une information.

- INFORM_REF : les robots se situe entre eux en s'envoyant leurs informations réciproques.
- AGREE : signifie qu'un l'agent robot est fonctionné.
- FAILURE : le robot est tombé en panne.

La figure suivante regroupe tous les messages déclarés dans le protocole de communication dans un seul scénario avec deux robots mobiles :

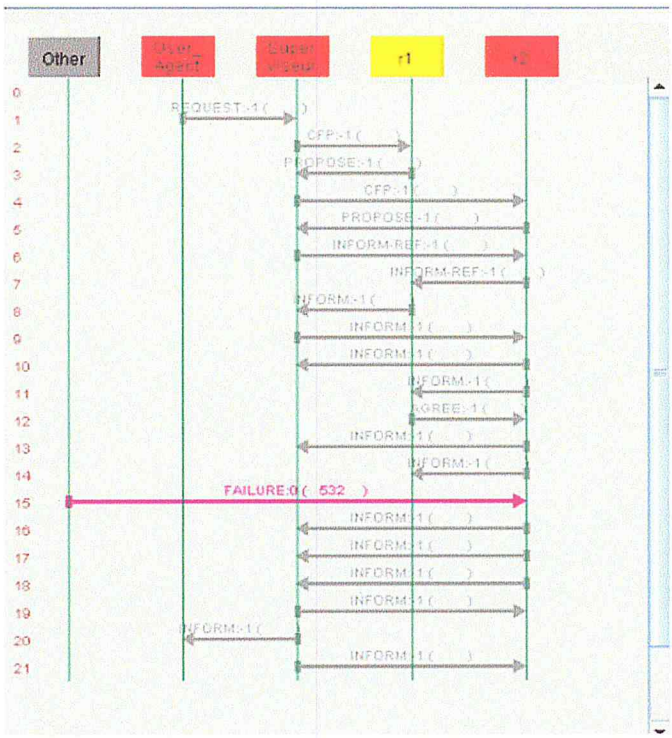



Figure 35: Exemple d'échange de messages du protocole utilisé

## 5. Conclusion

Dans ce chapitre, nous nous sommes intéressés à l'implémentation de l'architecture de contrôle proposée. Nous avons d'abord présenté les points importants de notre implémentation, en nous focalisant sur les environnements de développements utilisés. Ensuite, nous avons décrit l'implémentation des différentes classes utilisées.

Dans le prochain chapitre nous évaluerons l'efficacité et la robustesse du l'architecture de contrôle proposée grâce à différents scénario de validation.

# Chapitre 5:

# Chapitre 5
# Validation de l'architecture de contrôle proposée

## 1. Introduction

L'efficacité du fonctionnement d'une architecture de contrôle se montre par sa validation. Afin de pouvoir valider l'architecture de contrôle proposée dans les chapitres précédents, différents scénarios ont été réalisés.

Nous commençons d'abord par la description de l'application développée. Nous passons, par la suite, à la description et à l'exécution des différents scénarios de validation considérés. Le premier scénario traite le cas normal d'un SRCP sans panne avec cinq (05) robots mobiles. Le deuxième scénario traite le cas d'un ou plusieurs robots mobiles qui tombent en panne. La première variante de ce scénario est du leader tombe en panne. La seconde est celle d'un des suiveurs qui tombe en panne. Le deuxième est celle où le leader et l'un des suiveurs tombent en panne dans la même tâche. Le dernier scénario traite de l'intégration d'un nouveau robot au courant de l'exécution de la tâche leader/followers. Tous ces scénarios ont pour objectif de prouver l'efficacité et la robustesse de notre architecture proposée.

La tâche leader/followers est une tâche très intéressante en robotique mobile. En effet, il s'agit de faire mouvoir une équipe de robots mobile vers une destination tout en respectant une certaine formation. Les robots commencent tout d'abord, par sélectionner un leader qui est chargé de générer une trajectoire, éviter les obstacles, etc. Les autres robots, followers, ne font que suivre le leader en respectant une certaine vitesse de déplacement afin de préserves la formation. Cette tâche pose de nombreux problèmes tels que la synchronisation entre les robots, la planification de trajectoire, l'évitement d'obstacle, etc.

## 2. Présentation de l'application

### 2.1. Lancement de la plateforme Jade avec les agents de l'architecture

Pour lancer l'environnement JADE et exécuter les agents, nous devons exécuter la classe Initialisation_Agents. Cette classe lance la plateforme Jade avec Agent_Superviseur ; ce dernier lance les agents robots.

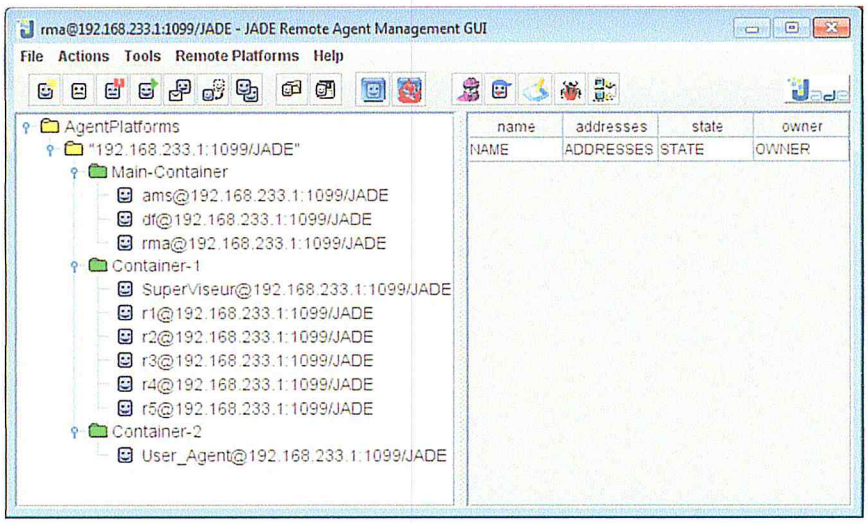



Figure 36: Lancement de la plateforme Jade avec les agents de l'architecture de contrôle

.

## 2.2 Interface utilisateur

L'interface utilisateur facilite la communication entre l'utilisateur et le système multi-agents (architecture de contrôle). L'exécution de cette interface peut s'effectuer dans une machine distante de l'architecture à travers un réseau local (fil/sans fil) tel que montrée dans la figure 37.

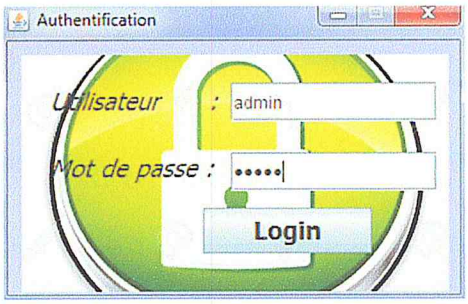



Figure 37: Authentification de l'utilisateur.

Après avoir été authentifié, l'utilisateur est autorisé à poursuivre l'utilisation du système et l'exécution des différentes tâches implémentées.


Figure 18: Interface de traitement.

### 2.3 Interfaces Intégration ou suppression d'un robot ou d'un objet

Ces deux interfaces (figures 39 et 40) ont pour le rôle de simplifier l'intégration et la suppression des robots et des objets dans le système. Donc dans le cas où l'utilisateur veut intégrer ou supprimer un robot dans le SRCP, il appuie sur le bouton intégrer/supprimer un robot ; puis, il remplit les différents champs (nom, position(x, y), etc.) afin d'envoyer une requête à l'agent Superviseur.

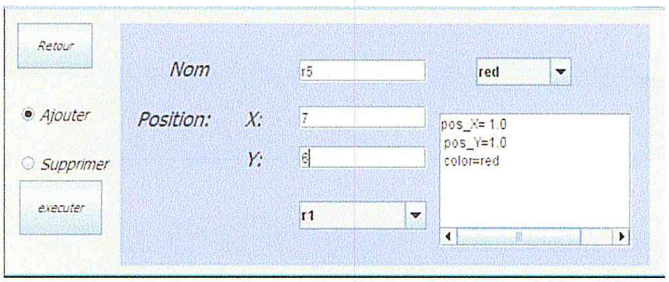



Figure 39: Interface d'intégration ou suppression d'un robot.

La même chose dans le cas où l'utilisateur veut intégrer ou supprimer système, il appuie sur le bouton intégrer/supprimer un objet ; puis, il remplit les différents champs (nom, position(x, y), etc.) afin d'envoyer une requête à l'agent Superviseur.

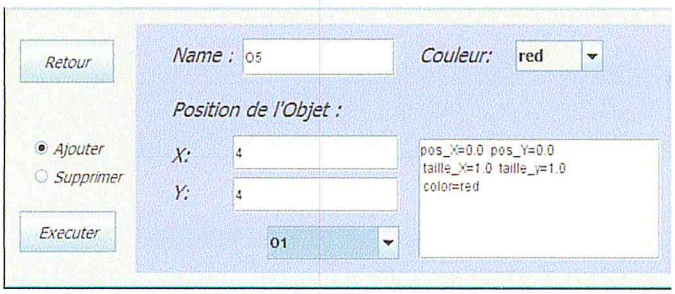



Figure 40: Interface d'intégration ou suppression d'un objet.

### 2.4. Description de l'environnement

La figure suivante (figure 41) représente l'environnement d'évolution des robots. Il est constitué des éléments suivants :

- 1 : L'environnement où les robots évoluent.
- 2 : Un robot follower.
- 3 : Un robot en panne.
- 4 : Un robot Leader.
- 5 : Un objet.
- 6 : Un obstacle.
- 7 : La destination.
- 8 : Trace de leader.

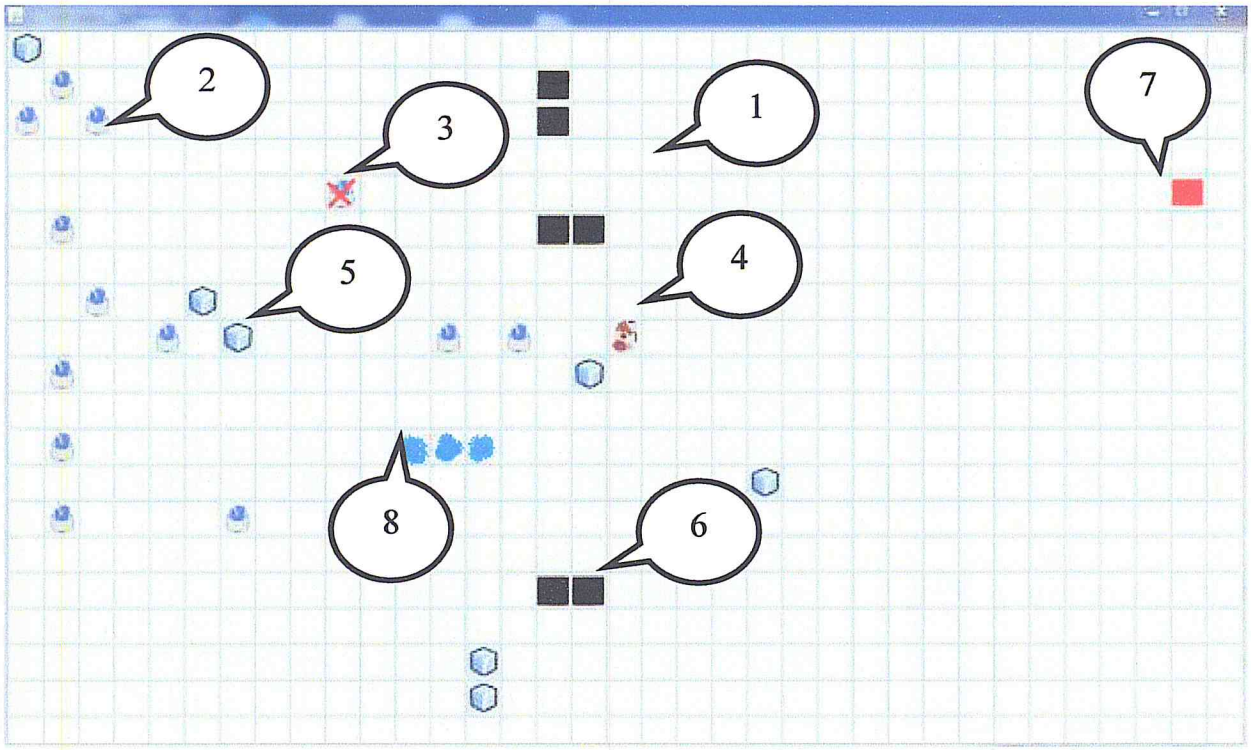



Figure 41: Description de l'environnement.

## 3 Scénarios de validation

Pour la validation de l'architecture de contrôle proposée, nous avons sélectionné différents scénarios de la tâche Leader/Followers avec cinq (05) robots mobiles et quelques objets dans l'environnement.

Le tableau suivant (Tableau 2) montre les paramètres initiaux des scénarios considérés. Chaque scénario comporte cinq (05) robots ayant une vitesse de déplacement (v=case/S) ; la

position de regroupement des robots est définie à la position (4.0, 8.0) ; la destination finale imposée aux robots est définie à la position (22.0, 8.0).

La fonction Objectif considéré est la distance euclidienne séparant les robots de leur destination. C'est en fonction de la valeur de cette fonction que sont classés les robots en leader ou en follower.

| Robot | Pos_x initiale | Pos_y initiale | L/F | Distance |
|---|---|---|---|---|
| R1 | 1.0 | 1.0 | F | 22.13 |
| R2 | 1.0 | 5.0 | F | 21.21 |
| R3 | 2.0 | 7.0 | F | 20.02 |
| R4 | 4.0 | 8.0 | L | 18.0 |
| R5 | 1.0 | 9.0 | F | 21.02 |

Tableau 2: Paramètres des scénarios considérés

### 3.1. Scénario 1 : Système sans panne

Dans le cas d'une tâche Leader/Followers, l'utilisateur sélectionne la position finale à atteindre par les robots tel que montré dans la figure 42.

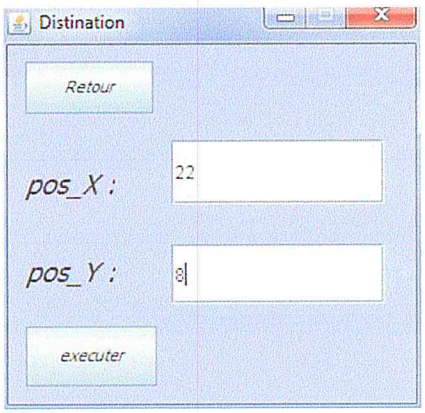



Figure 42 : Interface de saisie la destination et lancement de tâche.

Quand un utilisateur lance la tâche Leader/Followers, l'agent Superviseur la récupère et la transmet aux agents robots. Ces derniers calculent les distances séparant les robots de la destination dans le but de désigner le leader et les followers. À la fin, les robots s'alignent successivement selon les distances qui les sépare de la destination et commencent par la suite à exécuter la tâche.

Lors de l'exécution de la tâche, le leader laisse des marques (traces) sur le chemin qu'il a emprunté. Si un autre robot qui a été intégré au SRCP veut prendre la même que le leader, il n'aura qu'à suivre les déférentes traces laissées par ce robot leader. Au cours de l'exécution de la tâche les

robots communiquant à travers des ACLmessages pour échanger les informations entre eux. Les différentes étapes d'exécution de la tâche sont données, par les captures ci-dessous (figures 43) où :

- 1 : état initiale su SRCP.
- 2 : robots en position de regroupement.
- 3 : Robots en phase intermédiaire.
- 4 : Robots en phase finale.

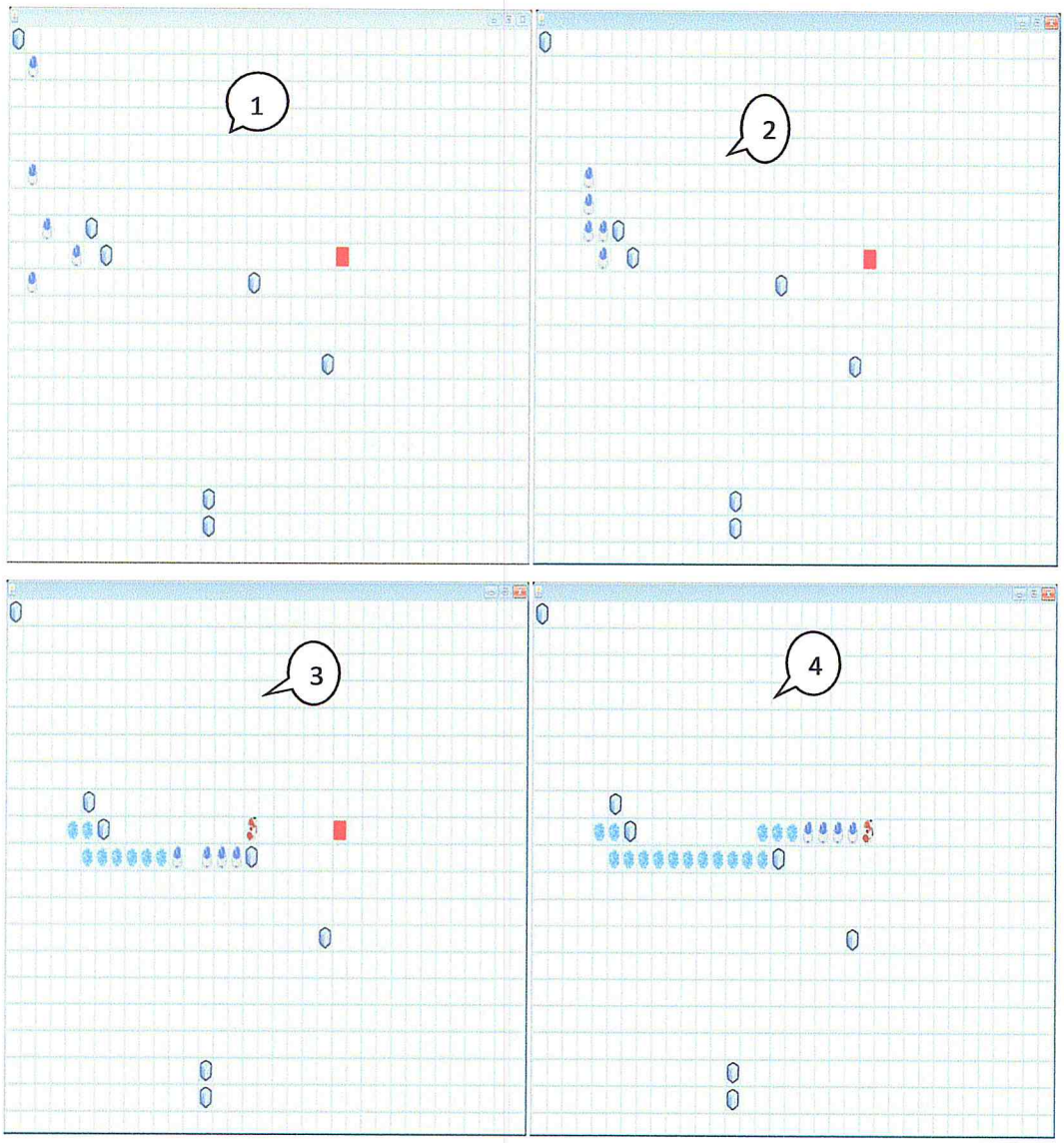



Figure 43: Résultat obtenu lors de l'exécution de la tâche dans un système sans panne.

Les différents messages échangés entre les agents robots de l'architecture de contrôle au cours de la réalisation de ce scénario sont donnés par la figure 44.

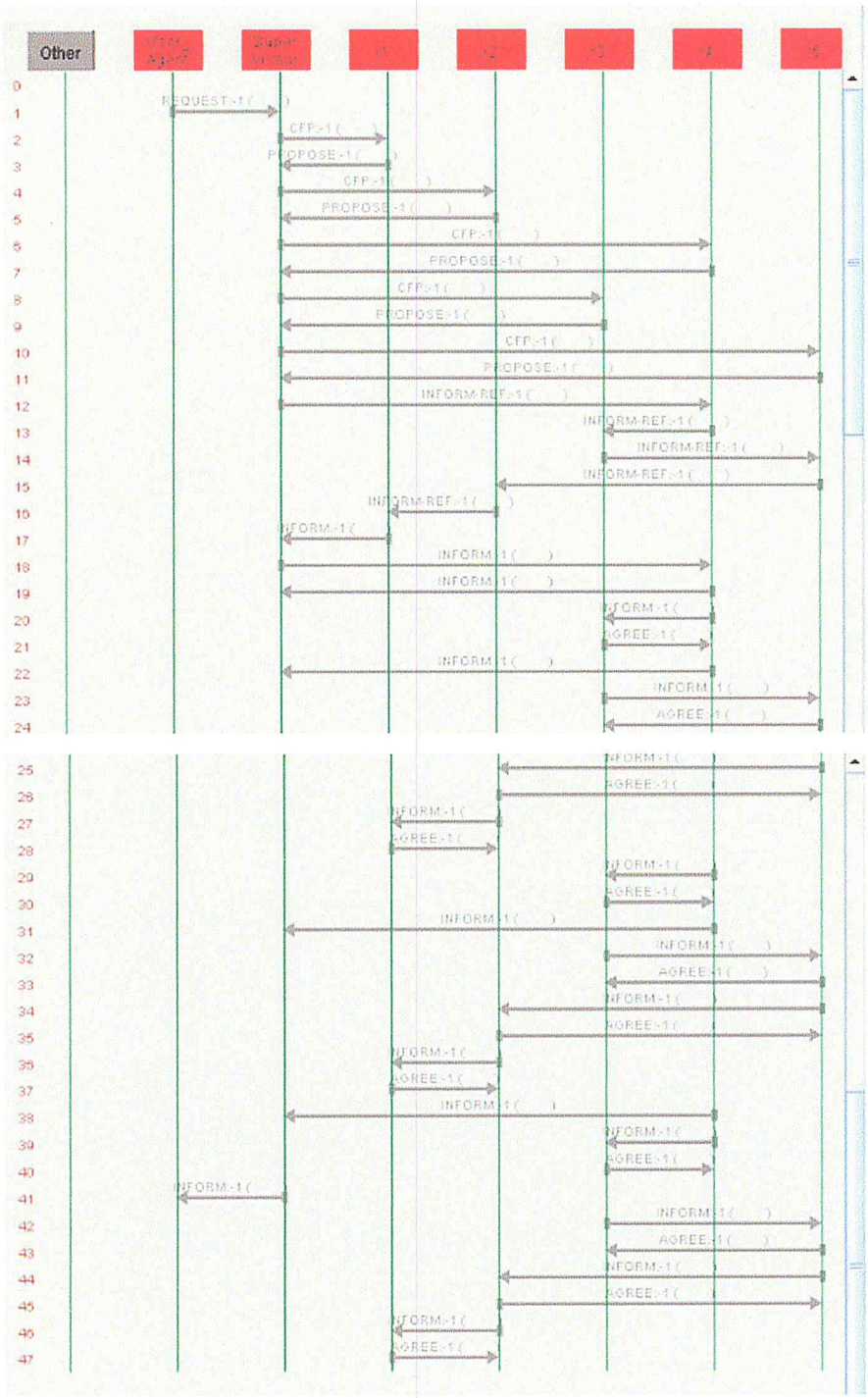



Figure 44 : Messages échangés entre les différents agents de l'architecture de contrôle

Le tableau suivant (Tableau 3) donne les différentes statistiques relatives à la réalisation de ce scénario :

| | Robot | Pos_x | Pos_y |
|---|---|---|---|
| **Position finale des robots** | R1 | 18.0 | 8.0 |
| | R2 | 19.0 | 8.0 |
| | R3 | 21.0 | 8.0 |
| | R4 | 22.0 | 8.0 |
| | R5 | 20.0 | 8.0 |
| **Temps de regroupement** | 8.95 s | | |
| **Temps de déplacement** | 26.87 s | | |
| **Temps totale de la tâche** | 35.82 s | | |
| **Nombre des messages échangés** | 210 messages | | |

Tableau 3: Robots en position finale (système sans panne)

### 3.2. Scénario 2 : Système avec pannes

#### 3.2.1 Leader tombe en panne

Dans le cas où le leader tombe en panne au cours de sa tâche, l'agent superviseur détecte la panne et informe les autres robots. Ces derniers vont négocier afin de sélectionner un nouveau Leader parmi les robots followers pour finaliser leur tâche. Le robot leader qui tombe en panne sera considéré comme un obstacle; il doit être évité par les robots followers. Les résultats de simulation de ce scenario sont donnés par la figure ci-dessous (figure 45) où :

- 1 : Robots en position de regroupement.
- 2 : Le leader tombe en panne.
- 3 : Choix d'un nouveau leader.
- 4 : Finalisation de la tâche.

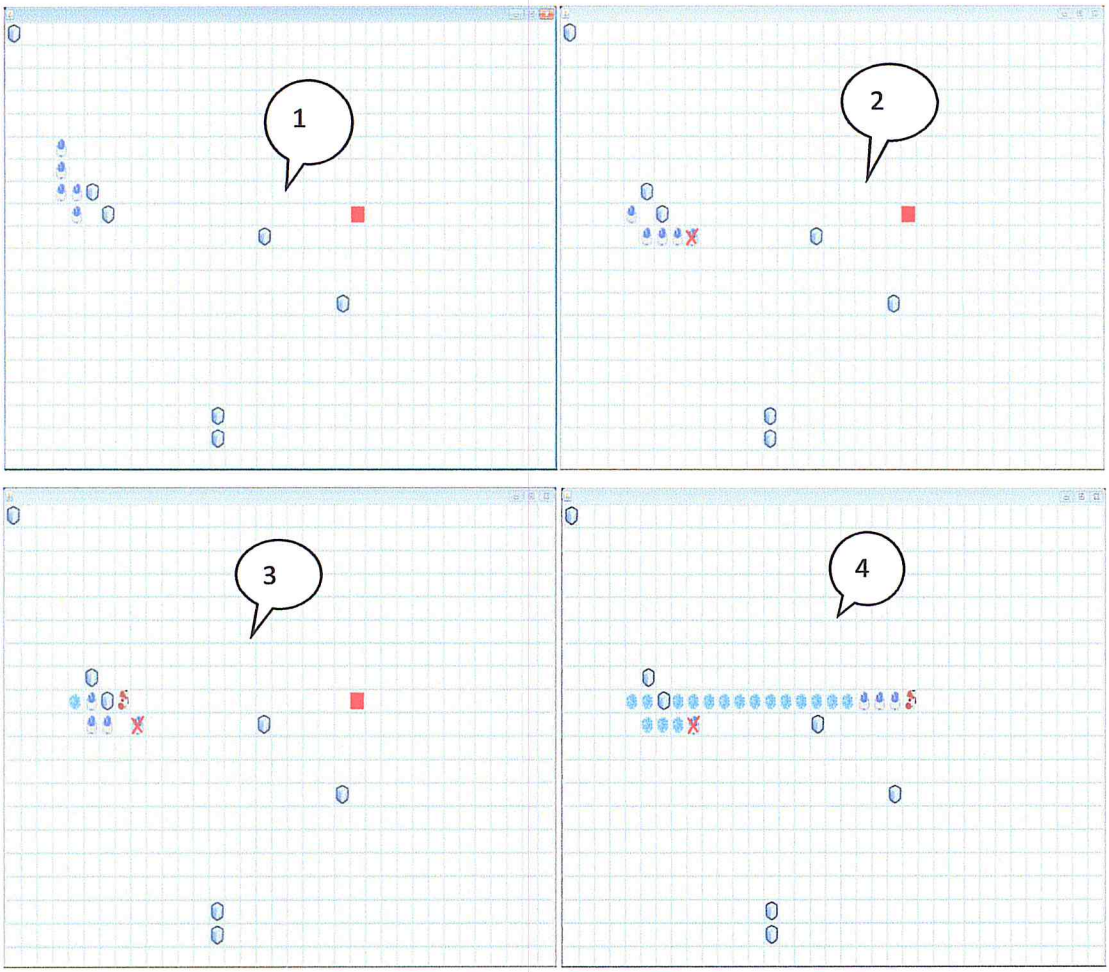
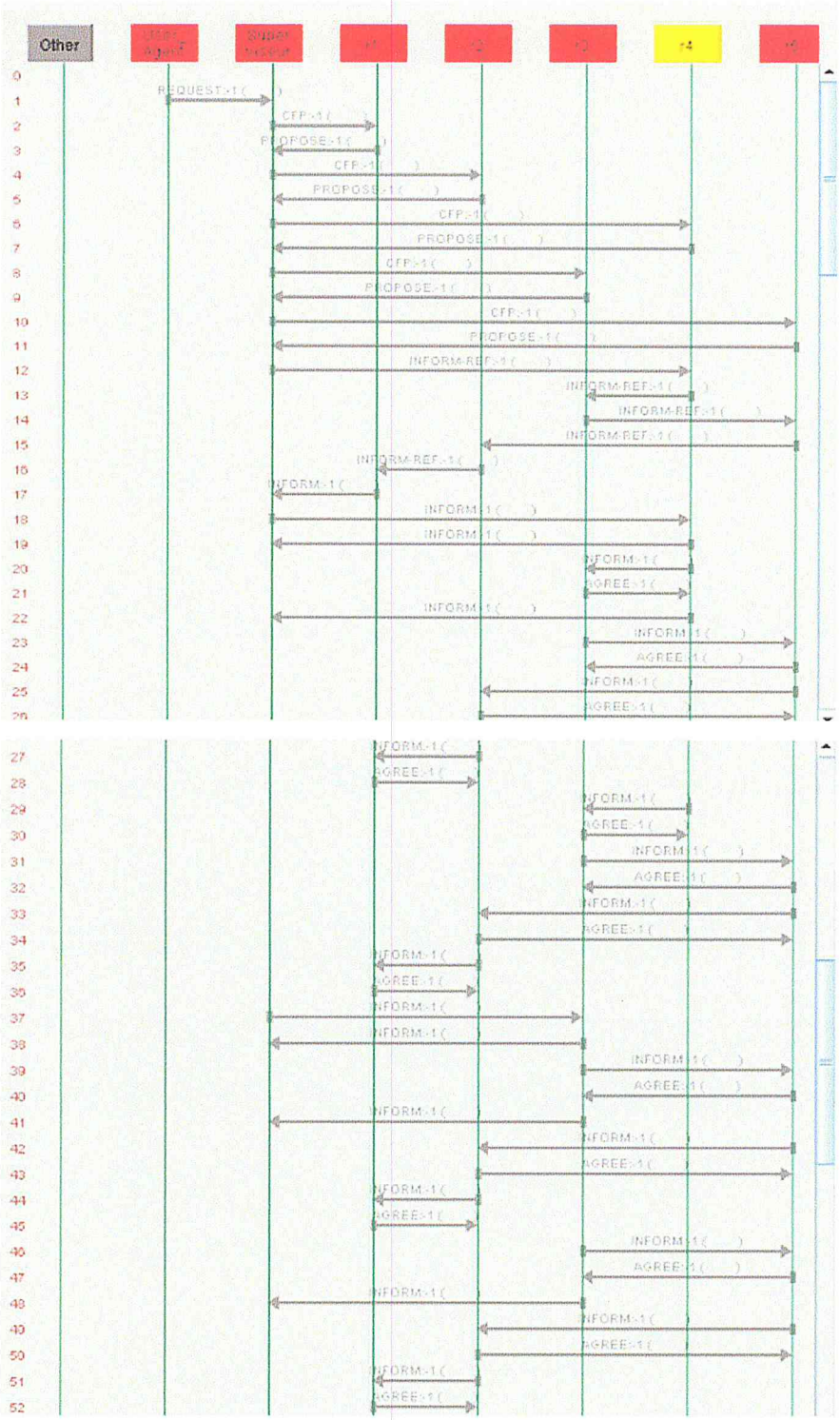



Figure 45: Résultats d'exécution lorsque le leader tombe en panne.

Les différents messages échangés entre les agents de l'architecture de contrôle dans le cas de la panne du leader sont montrés dans la figure 46 suivante :


Other
REQUEST:-1(
CFP:-1(
PROPOSE:-1(
INFORM-REF:-1(
INFORM:-1(
AGREE:-1(

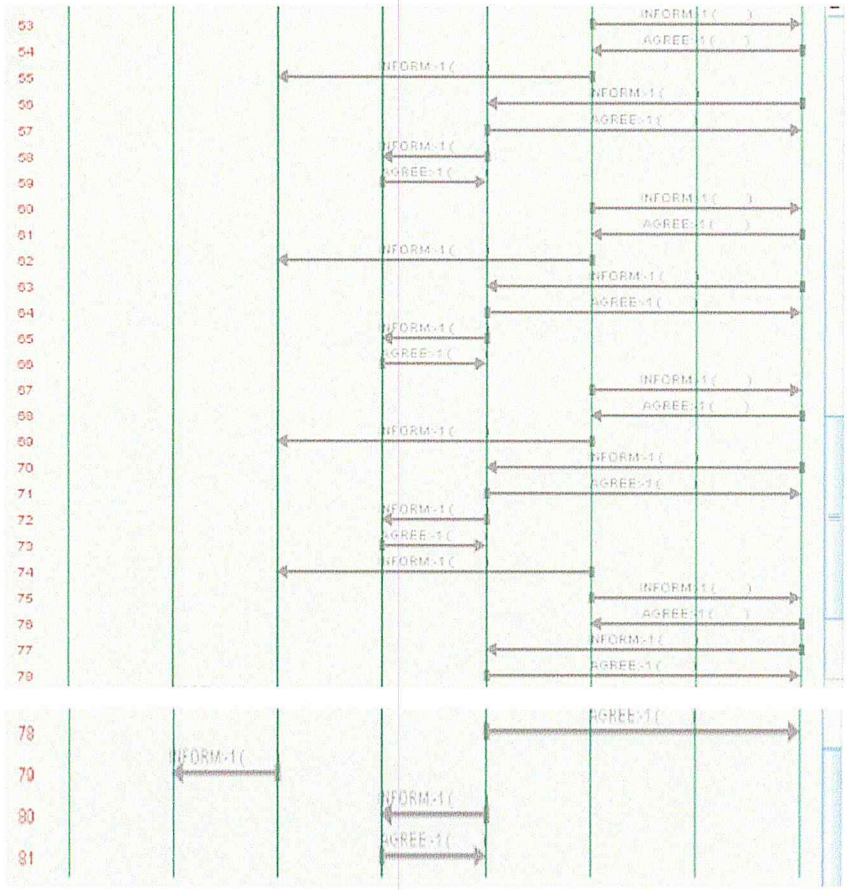


Figure 46: Messages échangés entre les différents agents de l'architecture pour le recouvrement de la panne du leader.

Les différentes statistiques relatives à l'exécution de ce scénario sont données par le tableau 4 suivant :

| | | | |
|---|---|---|---|
| **Position finale des robots** | **Robot** | **Pos_x** | **Pos_y** |
| | R1 | 19.0 | 8.0 |
| | R2 | 20.0 | 8.0 |
| | R3 | 22.0 | 8.0 |
| | R4 | 8.0 | 9.0 |
| | R5 | 21.0 | 8.0 |
| **Temps de regroupement** | 8.93 s | | |
| **Temps de déplacement** | 30.65 s | | |
| **Temps totale de la tâche** | 39.58 s | | |
| **Nombre des messages échangés** | 187 messages | | |

Tableau 4: Robots en position finale (leader en panne)

### 3.2.2. Un suiveur (Follower) tombe en panne

Au courant d'exécution de la tâche de leader/followers, nous considérons le cas où un follower tombe en panne. Premièrement le Superviseur en informe les autres robots ; par la suite, le robot prédécesseur de celui en panne transmet les informations au robot successeur (de celui en panne) afin de continuer l'exécution de la tâche.

Le robot suiveur (follower) qui est tombé en panne sera considéré comme un obstacle, Il doit être évité par les autres robots s'ils existent. Le résultat d'exécution est donné par la figure ci-dessous (figures 47) où :

- 1 : Follower en panne
- 2 : Finalisation de la tâche.

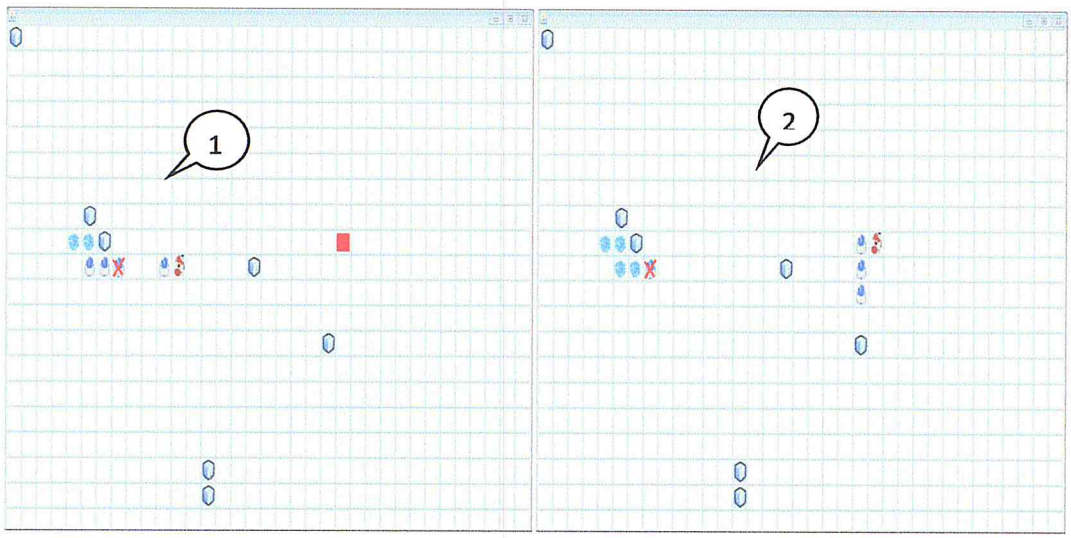



Figure 47: Résultat d'exécution lorsque l'un des suiveurs tombe en panne

Les différents messages échangés entre les agents robots de l'architecture de contrôle au courant de l'exécution de cette tâche sont donnés par la figure 48 :

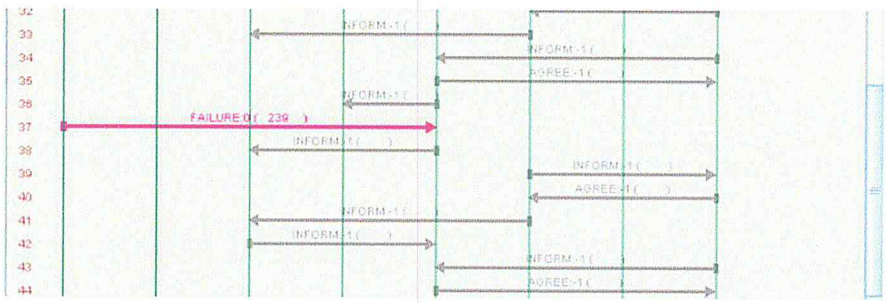


Figure 48: Messages échangés entre les différents agents de l'architecture pour le recouvrement de la panne d'un follower.

Lorsqu'un robot follower tombe en panne, le message FAILURE sera envoyé au robot prédécesseur de celui en panne, au ce moment le superviseur informe ce dernier qu'il doit être communiqué avec le robot successeur (de celui en panne) s'il existe, pour continuer l'exécution de la tâche.

Les statistiques de la réalisation de ce scénario sont données dans le tableau suivant (Tableau 5) :

| | Robot | Pos_x | Pos_y |
|---|---|---|---|
| **Position finale de robot** | R1 | 21.0 | 10.0 |
| | R2 | 21.0 | 9.0 |
| | R3 | 21.0 | 8.0 |
| | R4 | 22.0 | 8.0 |
| | R5 | 8.0 | 9.0 |
| **Temps de regroupement** | 8.95 s | | |
| **Temps de déplacement** | 32.08 s | | |
| **Temps totale de la tâche** | 41.03 s | | |
| **Nombre des messages échangés** | 179 messages | | |

Tableau 5: Robots en position finale (follower en panne)

### 3.2.3. Le leader et un suiveur (Follower) tombent en panne

Dans ce cas on rassemble les deux derniers scénarios c à d le leader et l'un des followers tombent en panne durant l'exécution de la tâche. La visualisation de recouvrement de cette situation est donnée par la figure ci-dessous (figure 49) où :

- 1 : Le leader et le follower tombent en panne.
- 2 : Choix d'un nouveau leader.

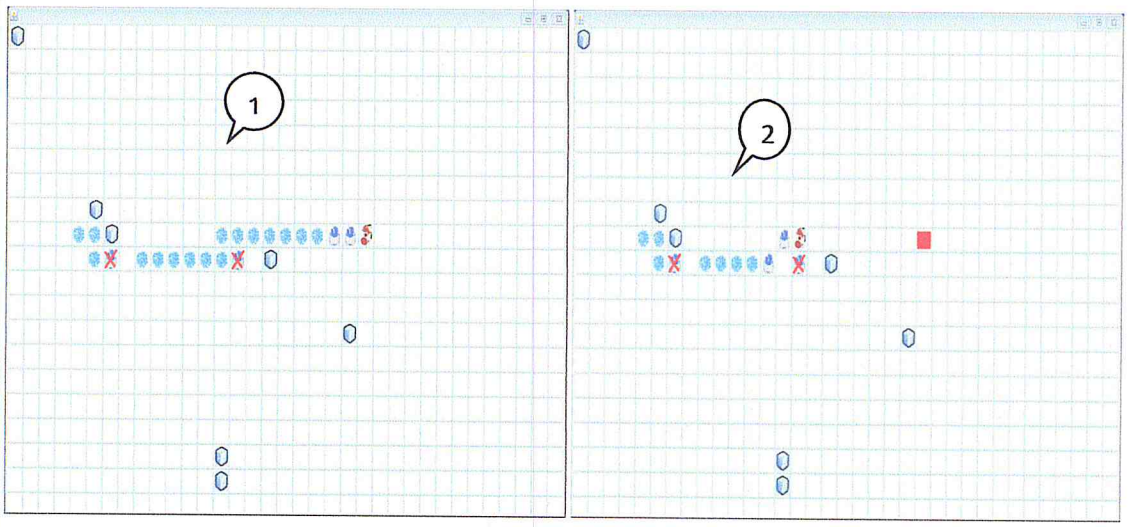



Figure 29: Résultat d'exécution lorsque un leader et un follower tombent en panne durant la même tâche.

Les différentes statistiques liées à la réalisation de ce scénario sont données dans le tableau 6 ci-dessous :

| | Robot | Pos_x | Pos_y |
|---|---|---|---|
| **Position finale de robots** | R1 | 21.0 | 9.0 |
| | R2 | 21.0 | 8.0 |
| | R3 | 22.0 | 8.0 |
| | R4 | 9.0 | 9.0 |
| | R5 | 10.0 | 8.0 |
| **Temps de regroupement** | 8.97 s | | |
| **Temps de déplacement** | 37.04 s | | |
| **Temps totale de la tâche** | 46.01 s | | |
| **Nombre des messages échangés** | 152 messages | | |

Tableau 6 : Robots en position finale (leader et follower en panne)

### 3.3 Scénario 3 : Intégration d'un nouveau robot au courant de l'exécution de la tâche leader/followers

En plein milieu d'exécution de la tâche leader/followers, l'utilisateur à l'instant (t =10s) ajoute un robot, ce dernier commencera au point de départ et cherchera à suivre le chemin emprunté par le leader .Cela est rendu possible grâce à la trace laissé par le robot leader lors de l'exécution de la tâche. Le résultat d'exécution de cette situation est donné par les figures ci-dessous (figure 50) où :

- 1 : Intégration nouveau robot.
- 2 : Le nouveau robot à la position de regroupement.
- 3 : Finalisant la tâche.

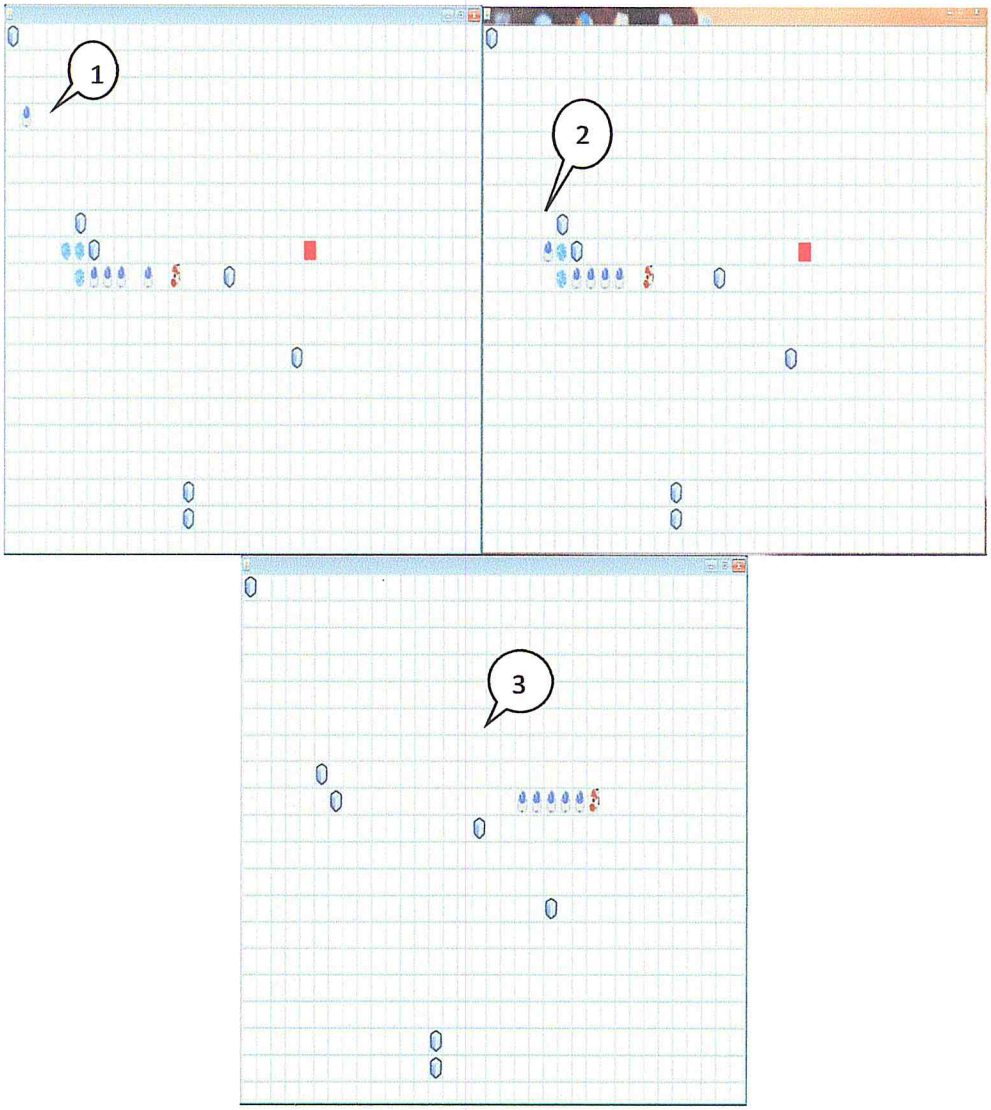



Figure 30 : Résultat d'exécution lors de l'intégration d'un nouveau robot au courant de l'exécution de la tâche de leader/followers.

À l'instant t = 10 s, nous avons simulé l'intégration d'un nouveau robot (R6) à la position (1.0, 3.0) au courant de l'exécution de la tâche de L/F. Les statistiques liées à l'exécution de ce scénario sont données comme suit (Tableau 7) :

| | Robot | Pos_x | Pos_y |
|---|---|---|---|
| **Position finale de robot** | R1 | 18.0 | 8.0 |
| | R2 | 19.0 | 8.0 |
| | R3 | 21.0 | 8.0 |
| | R4 | 22.0 | 8.0 |
| | R5 | 20.0 | 8.0 |
| | R6 | 17.0 | 8.0 |
| **Temps de regroupement** | 8.95 s | | |
| **Temps de déplacement** | 26.87 s | | |
| **Temps totale de la tâche** | 35.8 s | | |
| **Nombre des messages échangés** | 212 messages | | |

Tableau 7: Robots en position finale (cas d'intégration nouveau robot)

### 3.4. Discussion des résultats obtenus

Dans les scénarios précédents, nous avons simulé la réalisation de différents scénarios de la tâche leader/followers avec cinq (05) robots se déplaçant avec une vitesse de (v=case/S).

Nous avons aussi considéré un environnement encombré de quelques obstacles. La solution que nous avons proposée a permis d'obtenir des résultats satisfaisants pour tous les scénarios considérés.

Dans le premier scénario considéré (cas du fonctionnement sans panne du système), le temps total d'exécution de la tâche est de 35.8s ; le nombre de messages échangés entre les agents est 210 messages. Ces chiffres représentent les résultats idéaux de la réalisation de cette tâche.

Par contre dans le deuxième cas où nous avons simulé des pannes de quelques robots du système, nous distinguons trois cas :

- Dans le premier cas où le leader tombe en panne, nous remarquons que le temps total de la réalisation de tâche est monté à 39.5s ; le nombre total des messages échangés est descendu à 187 messages.
- Dans le deuxième cas où l'un des followers tombe en panne, le temps total d'exécution de la tâche est de monté à 41.0s ; le nombre de messages échangés entre les agents est descendu à 179 messages.

- Dans le troisième et dernier cas, il rassemble les deux cas précédentes (leader et un follower en panne), le temps total de la réalisation de la tâche est de 46.0s ; le nombre total des messages échangés entre les agents est devenu égal à 152 messages.

Nous pouvons justifier l'augmentation du temps total d'exécution de la tâche par la panne inopinée ; cela a engendré un temps supplémentaire perdu pendant la négociation entre les agents du système pour le recouvrement de la panne détectée. Par contre, la diminution du nombre de messages échangés entre les différents agents de l'architecture de contrôle est due à la diminution du nombre d'agents robots (fin de fonctionnement du /des robot(s) en panne).

Dans le troisième scénario considéré (cas d'intégration d'un nouveau robot au cours de la tâche), le temps total d'exécution de la tâche est de 35.8s (le nouveau robot intégré atteindre les autres robots avant d'avoir fini la tâche, c.à.d. même temps que le cas où le système non panne) ; le nombre de messages échangés entre les agents est 212 messages.

## 4. Conclusion

Dans ce dernier chapitre, plusieurs scénarios de validation de la tâche « leader/followers » ont été réalisés.

- Le premier scénario décrit le fonctionnement normal du système de contrôle (sans panne) lors de la réalisation des tâches.
- Le deuxième scénario traite le cas où quelques robots tombent en panne ; ce scénario comporte trois cas :
  - La panne de robot leader.
  - La panne de l'un des followers.
  - Panne du leader et l'un des followers.
- Le troisième scénario considère l'intégration d'un nouveau robot en cours de l'exécution de la tâche.

En examinant les différents résultats obtenus, nous pouvons affirmer de la validation de l'architecture que nous avons proposée pour le contrôle d'un SRCP.

# Conclusion générale

# Conclusion générale

Le domaine de la robotique fait face à une croissance rapide dans la complexité des besoins et des exigences pour des robots chargés de tâches multiples, capables de coordonner leurs actions. En parallèle, une évolution similaire dans le domaine des systèmes temps-réel embarqués répartis a justifié l'émergence du domaine des « systèmes cyber-physiques (SCP) » reflétant une montée similaire en complexité.

Les architectures logicielles dans ces différents domaines cherchent globalement à intégrer un système informatisé de contrôle avec d'autres traitements d'informations de plus en plus lourds (planification, analyse de données, apprentissage, etc.), et ce dans un contexte en prise constante avec le monde réel.

L'objectif de notre travail est le développement d'une architecture de contrôle d'un « système robotique cyber-physique » dans laquelle des opérateurs humains, des robots ou des objets quelconques (salles, murs, obstacles, objets étiquetés, etc.) peuvent s'enregistrer au niveau du contrôleur et échanger des informations avec toutes les entités enregistrées. Nous avons, pour cela, opté pour une architecture multi-agents pour contrôler ce type de systèmes.

Au début de ce mémoire nous avons fait une présentation des SCP et des SRCP. Ces derniers consistent en un ensemble de robots autonomes fonctionnant dans un environnement intelligent ; les robots ont la capacité de coopérer et de communiquer avec les différentes entités de ce système. Les systèmes SCP et SRCP peuvent aussi changer la façon dont les individus et les organisations interagissent et contrôlent le monde physique à cause de la combinaison de calcul, de communication et de contrôle. Par la suite, nous avons présenté les d'architectures de contrôle des systèmes multi-robots hétérogènes et leur différentes classes.

L'architecture logicielle de contrôle que nous avons proposée est une architecture générique, hybride et distribuée ; elle exploite la technologie des « systèmes multi-agents » dans un contexte robotique cyber-physique. En effet l'architecture de contrôle proposée est composée de deux couches de traitement de l'information :

- Couche physique : elle représente toutes les composantes physiques de l'environnement où à chaque robot est assigné un agent de contrôle dédié à la prise de décision et à l'exécution des consignes.

- Couche contrôle : elle joue le rôle d'intermédiaire entre l'extérieur (les opérateurs) et le système robotique. Cette couche est destinée à contrôler et à gérer la couche physique, grâce à la coordination et la négociation. Cette couche communique avec la couche physique, par l'envoi de requêtes et par réception des informations et de compte-rendu.

Nous avons testé l'efficacité de l'architecture de contrôle proposée via quelques scénarios de validation de la tâche « leader/followers ».

Nous avons, pour cela, considérer cinq cas différents.

Pour chaque cas, nous avons donné le temps de réalisation de la tâche et le nombre global de messages échangés entre les agents de l'architecture de contrôle :

- le premier scénario considéré le fonctionnement sans panne du système.
- Le deuxième scénario simule une panne au niveau du robot leader.
- Le troisième scénario simule une panne au niveau du l'un des robots followers.
- Le quatrième scenario rassemble les deux scénarios précédents.
- Le dernier scénario considère l'intégration d'un nouveau robot au courant de l'exécution de la tâche.

Les résultats obtenus pour tous les scénarios ont montré l'efficacité la robustesse et la tolérance aux pannes de l'architecture de contrôle proposée.

Les perspectives pour d'éventuels travaux futurs concernent essentiellement l'implémentation et la validation expérimentale de cette architecture logicielle dans un environnement réel cyber-physique ; il y a lieu de considérer un grand nombre de robots hétérogènes, et dans un environnement complexe (plusieurs objets/obstacles).

# Annexe

# Annexe 01

# Systèmes multi-agents (SMA)

## ❖ Concept d'agent

La notion d'agent n'est pas simple à définir. Il existe en effet plusieurs définitions ou significations données à cette notion. C'est la raison pour laquelle plusieurs auteurs essayent d'en donner une définition avant de se pencher sur l'utilisation de ce paradigme dans tel ou tel contexte.

Ferber définit un agent comme une entité physique ou virtuelle [25] :

- qui est capable d'agir dans un environnement.
- qui peut communiquer avec d'autres agents.
- qui possède des ressources propres.
- qui est capable de percevoir (mais de manière limitée) son environnement,
- qui ne dispose que d'une représentation partielle de cet environnement (et éventuellement aucune),
- qui possède des compétences et offre des services,
- qui peut éventuellement se reproduire,
- dont le comportement tend à satisfaire ses objectifs, en tenant compte des ressources et des compétences dont elle dispose, et en fonction de sa perception, de ses représentations et des communications qu'elle reçoit.

Des propriétés les plus important d'un agent permet être résume aussi [25] :

- Situé : l'agent est capable d'agir sur son environnement à partir des entrées sensorielles qu'il reçoit de ce même environnement. Exemples : systèmes de contrôle de processus, systèmes embarqués, etc.
- Autonome : l'agent est capable d'agir sans l'intervention d'un tiers (humain ou agent) et contrôle ses propres actions ainsi que son état interne.
- Flexible : l'agent dans ce cas est :
  - Capable de répondre à temps : l'agent doit être capable de percevoir son environnement et élaborer une réponse dans les temps requis.

- Proactif : l'agent doit exhiber un comportement proactif et opportuniste, tout en étant capable de prendre l'initiative au "bon" moment.
- Social : l'agent doit être capable d'interagir avec les autres agents (logiciels et humains) quand la situation l'exige afin de compléter ses tâches ou aider ces agents à accomplir les leurs.

## ❖ Système multi-agent (SMA)

Un SMA est un ensemble d'agents autonomes en interaction, capables de s'organiser d'une manière dynamique et adaptative [42].

### 1. Plateformes de développement des SMA

Les environnements de développement ou les plateformes multi-agents sont nécessaires pour renforcer le succès de la technologie multi-agents. Les plateformes multi-agents permettent aux développeurs de concevoir et réaliser leurs applications sans perdre de temps à réaliser des fonctions de base pour la création et l'interaction entre agents et éliminent, dans la plupart des cas, la nécessité d'être familier avec les différents concepts théoriques des SMA.

Il existe un nombre important d'environnements de développement des applications orientées agents. Il y a aussi bien des produits commerciaux que des logiciels dans le domaine public. Il existe un certain nombre de plateformes fournies comme logiciels libres telles que : CORMAS (COmmonResources Multi-Agent System) [43], JACK [44], et JADE [45].

### 2. Interaction dans les SMAs

Une interaction est une mise en relation dynamique de deux ou plusieurs agents par le biais d'un ensemble d'actions réciproques. C'est grâce à l'interaction que le SMA est vu comme un tout et non pas comme un ensemble d'entités indépendantes. Pour un agent, interagir avec un autre constitue à la fois la source de sa puissance et l'origine de ses problèmes [25].

Les différentes situations d'interactions entre les agents sont la communication, la coopération, l'organisation, la négociation et la coordination [25].

- **Communication**

La communication est l'élément de base de toute interaction. Elle permet l'échange des informations entre deux agents. En communiquant, les agents peuvent échanger des informations et coordonner leurs activités. Cette Communication peut être :

- Directe : par l'envoi de message point à point.
- Indirecte : qui se fait à travers l'environnement où les agents laissent des traces ou des signaux qui seront perçus par les autres agents, ou bien par le biais d'un tableau noir qui est une mémoire partagée accessible par l'ensemble des agents.

- **Coopération**

Demazeau et Müller [42] parlent de coopération pour une tâche locale, lorsqu'un agent a besoin de coopérer avec un autre parce qu'il n'est pas capable de l'accomplir par lui-même ou parce que les autres peuvent l'accomplir de manière plus efficace que lui.

En général, on dira que plusieurs agents coopèrent, ou encore qu'ils sont en situation de coopération, si l'une de ces deux conditions est vérifiée [25] :

- L'ajout d'un nouvel agent accroît différentiellement les performances du groupe.
- L'action des agents sert à éviter ou à résoudre des conflits potentiels ou actuels.

- **Coordination**

La coordination est définie comme l'acte de gérer les interdépendances des différentes activités exécutées pendant la réalisation d'un but. Les interdépendances regroupent les prés requis (résultat d'une activité est nécessaire à une autre activité), le partage des ressources et la simultanéité (il existe une synchronisation entre l'exécution des activités). Suivant cette définition, la coordination recouvre les indices de coopération se rapportant au partage des ressources, à la coordination des actions et à la parallélisassions des actions [42].

- **Collaboration**

Selon Ferber [25], la collaboration indique "l'ensemble des techniques permettant à des agents de répartir des tâches, des informations et des ressources de manière à réaliser une œuvre commune".

- **Organisation**

Les organisations permettent de structurer les entités du système. Lorsqu'un agent est amené à interagir, l'organisation lui indique généralement avec quel autre agent le faire et comment le faire, lui évitant de faire cette recherche par lui-même [42]. De ce point de vue, l'organisation permet de réduire la complexité de l'espace de recherche qu'un agent doit envisager pour atteindre son but.

- **Négociation**

La négociation est définie comme un processus de communication d'un groupe d'agents permettant d'atteindre un accord mutuellement accepté et de résoudre leur conflit en défendant leurs points de vue respectifs pour arriver à un compromis, en partageant des ressources limitées ou encore en coordonnant leurs actions. La négociation est basée sur des protocoles qui assignent des rôles aux agents. Chaque agent impliqué dans la négociation exécute le protocole avec le rôle qui lui est assigné [25].

# Bibliographie

# Références bibliographiques


[1] S. Jeschke, "Cyber-Physical Systems : History, Presence and Future, Industrial Advisory Board", Faculté de Ingénierie de Mécanique RWTH, Université d'Aachen, 2013.

[2] E.A. Lee, "Computing foundations and practice for cyber-physical systems: A preliminary report", Université de California, Berkeley, 2007.

[3] R.R. Rajkumar, I.Lee, L.Sha et J.Stankovic, "Cyber- physical systems: the next computing revolution", Design Automation Conference 2010, Anaheim, California, USA. 2010.

[4] E. Starkloff, "L'observatoire des tendances 2014 de NI : Des tendances technologiques au service de la productivité", National Instruments, 2014.

[5] A. Huebner, C. Facchi, M. Meye et H. Janicke, "RFID Systems from a Cyber-Physical Systems Perspective",2014.

[6] L. Adouane. "Architectures de Contrôle Comportementales et Réactives pour la Coopération d'un Groupe de Robots Mobiles Architecture de contrôle hybride pour systèmes multirobots mobiles". Thèse de doctorat, Laboratoire d'Automatique de Besançon.

[7] http://www.rfid360.org/fonctionnement-dun-systeme-rfid/ Dernière consultation Avril 2016.

[8] http://rfid.comprendrechoisir.com/comprendre/lecteur-rfid/ Dernière consultation : Avril 2016.

[9] http://www.astuces-pratiques.fr/electronique/principe-technique-de-la-nfc/ Dernière consultation : Avril 2016.

[10] http://www.centrenational-rfid.com/comment-fonctionne-le-nfc-article-133-fr-ruid17.html/ Dernière consultation : Avril 2016.

[11] http://www.zigbee.org/what-is-zigbee/ Dernière consultation : Avril 2016.

[12] http ://www.st.com/web/en/press/fr/t3340/ Dernière consultation : Avril 2016.

[13] P.Ilunga Katamba ,"Technologie Rfid (Radio Frequency Identification) : Concepts Et Stratégie De Mise En Oeuvre", Mémoire pour l'obtention du grade de maltrise en sciences (M. Se.), Université Laval , Québec,pp.02-145,2007.

[14] http://tectron88.en.made-in-china.com/product/WvSJXREYHQkF/China-TectronIntelligent-Parking-Guidance-System-PGS-2010-.html/Dernière consultation: Avril 2016.

[15] J. Markoff, "Can't Find a Parking Spot ? Check Smartphone", Presse, New York Times,July 12, 2008.

[16] http://www.comlive.net/Les-Technologies-Militaires,130241.htm/ Dernière consultation : Avril 2016.

[17] F. Hu, "Cyber-Physical Systems Integrated Computing and Engineering Design", 2010.

[18] J.Arlat, C.Artigues, Y.Deswarte, M.Devy, M.Diaz, J.Dilhac, K.Drira, B.Estibals,K.Kanoun, A.Nketsa, P.Pons, F.vernadat, "ADREAM Architectures Dynamiques et Reconfigurables pour les systèmes Embarqués Autonomes Mobiles Vers la Conception et l'Evaluation des Systèmes Cyberphysiques", Decembre 2012.

[19] E.Iáñez, A.Úbeda, J.M.Azorín et C.Perez-Vidal. "Assistive robot application based on an RFID control architecture and a wireless EOG interface",journal elsevier Robotics and Autonomous Systems 60 (2012) 1069–1077 , 2012.

[20] I-Mosyde " L'Industrie 4.0, l'Internet des Objets, l'Automatisation verte, l'e-Santé, l'Apprentissage en ligne, ... vers une évolution ou une révolution en MOdern SYstem DEsign?",2Seas Magazine, Novombre, 2014.

[21] K.Coble, W.Wang, B.Chu et Z.Li, "Secure Software Attestation for Military Telesurgical Robot Systems", Military Communications Conference - Unclassified Program Cyber Security and Network Management, 2010.

[22] F. Marchal, P. Rauch, J.-L. Verhaeghe, F. Guillemin, "Perspectives de la chirurgie robotique et conclusions Indications, techniques, diffusion et ergonomie", Article, 2011.

[23] H.Liu, N.Stoll, S.Junginger et K.Thurow, "Mobile Robot for Life Science Automation", 2013.

[24] T.Hasegawa, K.Murakami, R.Kurazume, Y.Senta, Y.Kimuro et T.Ienaga, "Robot Town Project: Sensory Data Management and Interaction with Robot of Intelligent Environment for Daily Life", The 4th International Conference on Ubiquitous Robots and Ambient Intelligence, 2007.

[25] J.Ferber, "Les Systèmes Multi Agents: vers une intelligence collective", InerEditions, 1995.

[26] J. Liu, and Wu, J. Multiagent Robotic Systems. CRC Press LLC. (2001).

[27] D. M. Lyons, and Arbib, M. A. A formal model of computation for sensory-based robotics. IEEE Transactions on Robotics And Automation, 5(3):280–293 (1989).

[28] L. Parker, Distributed intelligence: Overview of the field and its application in multi-robot systems. Journal of Physical Agents, 2(1):5–14 (2008a).

[29] F.Arrichiello: Coordination Control of Multiple Mobile Robots. PhD thesis, Università degli Studi di Cassino (2006).

[30] B. Gerkey, P. and Matari´c, M. J. Sold ! : Auction methods for multirobot coordiantion. IEEE Transactions on Robotics And Automation, 18(5):758–768. (2002).

[31] M. Dias, B. TraderBots: A New Paradigm for Robust and Efficient Multirobot Coordination in Dynamic Environments. PhD thesis, Robotics Institute, Carnegie Mellon University. (2004).

[32] N. Kalra, T. Stentz, and Ferguson, D. Hoplites: A market framework for complex tight coordination in multi-agent teams. Technical Report CMU-RI-TR-04-41, Robotics Institute Carnegie Mellon University. (2004).

[33] B.Khoshnevis& G. Bekey. Centralized sensing and control of multiple mobile robots. Computers & Industrial Engineering, vol. 35, pp. 503 506, 1998.

[34] K.Lerman, A. Galstyan, A. Martinoli & A. Ijspeert. A Macroscopic ANALYTICAL Model of Collaboration in Distributed Robotic Systems. Artificial life, vol. 7, pp. 375 393, 2001.

[35] R.W Beard, J. Lawton & F. Y. Hadaegh. Coordination Architecture for Spacecraft Formation Control. IEEE Transactions on Control Systems Technology, vol. 9, pp. 777 790, 2001.

[36] L. Parker, ALLIANCE: An Architecture for Fault Tolerant Multi-Robot Cooperation, IEEE Transactions on Robotics and Automation, pp.220-240. 1998.

[37] R.A. Brooks. A Robust Layered Control System for a Mobile Robot. Massachusetts Institute of Technology, Artificial Intelligence Laboratory, 1985.

[38] T.L. Huntsberger, P. Pirjanian , A. Trebi-Ollennu, H. Das, H. Aghazarian, A.J. Ganino, M.S. Garrett, S.S. Joshi, and P.S. Schenker. CAMPOUT: A Control Architecture for Tightly Coupled Coordination for Multi-robot Systems for Planetary Surface Exploration.IEEE Transactions on Systems, Man, and Cybernetics, Volume 33, Number 5,pp.555-559. 2003.

[39] T. Pilarski, M. Happold,H. Pangels, M. Ollis, K. Fitzpatrick, and A. Stentz. The DEMETER System for Autonomous Harvesting. Autonomous Robots, Vol. 13, No. 1, pp.9-20. 2002.

[40] F. Bellifemine, L., Caire, G., & Greenwood, D. Developing multi-agent systems with JADE (Vol. 7). John Wiley & Sons. (2007).

[41] A.Afshari , M.Mojahed et R.M.Yusuff , "Simple Additive Weighting approach to Personnel Selection problem", International Journal of Innovation, Management and Technology, Vol.1, No. 5, December 2010.

[42] Th.W.Maldone, "What is coordination theory In National Science Foundation Coordination Theory Workshop", The National Science Foundation Coordination Theory Workshop, 1988.

[43] http://cormas.cirad.fr/ Dernière consultation : Mai 2016.

[44] http://aosgrp.com/products/jack/ Dernière consultation : Mai 2015.

[45] F.Bellifemine, A.Poggi et G.Rimassa, "JADE – A FIPA-compliant agent framework", Article, 1999.