République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Saad Dahlab Blida

Faculté des sciences

Département d'informatique

Mémoire Présenté par :

Lalaoui Omar

En vue d'obtenir le diplôme de master

Domaine : Mathématique et informatique

Filière : Informatique
Spécialité : Informatique
Option : Ingénierie de logiciel

Sujet :

Interface homme/robot (IHR) de contrôle d'une équipe de robots hétérogènes dans un système robotique cyber-physique.

Soutenu le :

M.	Président
M.	Examinateur
M.	Examinateur
Mr HENTOUT Abdelfetah.	Encadreur.
Mr Kameche Abdelah Hichem,	Promoteur.

Promotion
2015 / 2016

Dédicaces

Je dédie ce modeste travail

A vous très cher père et très chère mère, en témoignage de mon amour et de ma reconnaissance

A ma sœur Zineb

A toute ma famille

A mon encadreur Mr Hentout Abdelfetah

A mon promoteur Mr Kameche Abdellah Hichem

A tous mes collègues, plus particulièrement : Amine, Badro, Elias, Ghanou, Fouzi, Youcef

en témoignage De mon amitié sincère;

A tous mes amis, plus particulièrement : Sofiane, Chahinaz, Lotfi, Karima, Hadjer, Faiza, Reda, Meriem, Yasser en

Témoignage de mon amitié sincère;

A tous ceux qui m'ont soutenu, qu'ils trouvent ici l'expression de mon amour et ma profonde

Gratitude

Remerciement

Tout d'abord, je tiens à rendre grâce à Dieu tout puissant pour nous avoir donné le courage et la détermination nécessaires pour finaliser ce travail et le mener à terme.

En premier lieu, Je remercie mon encadrant Mr. Hentout Abdelfetah pour avoir accepté de diriger ce travail. Sa disponibilité, sa patience, ses compétences et ses conseils avisés m'ont été d'une aide inestimable.

Je voudrais également remercier Mr. Kameche Abdallah Hichem qui nous a fait l'honneur d'être notre promoteur ainsi que pour son aide, ses conseils et son soutient durant ce projet.

Je tiens aussi à remercier Mr. Abderraouf Maoudj, Kachouane Hadjer, Hamdania Amel et Dari hanane pour leurs aides et leurs encouragements.

Je remercie les membres de jury pour nous avoir fait l'honneur de juger notre travail.

Un grand Merci à mes amis, à tous ceux qui m'ont soutenu et à tous ceux qui ont participé de près ou de loin à l'aboutissement de ce travail.

Je termine par dédicacer ce mémoire à mon père, ma mère et ma sœur qui sans eux, tout cela n'aurait pas été possible ...

Résumé

Ce travail traite l'un des problèmes de décision les plus difficiles dans les systèmes cyber-physiques celui de : la planification des tâches. Il s'agit d'affecter des tâches aux robots selon une certaine séquence.

Notre travail a pour objectif de développer une interface Homme/Robot (IHR) destinée au contrôle collaboratif multi-robots dans un système cyber-physique.

Cette interface homme/robot a comme rôles de :

- Transmettre les intentions de l'opérateur humain aux robots (tâches à exécuter par un seul robot, un sous-ensemble de robots, tous les robots du SRCP).
- Transmettre à l'opérateur les informations issues des différents capteurs implantés dans l'environnement cyber-physique où évoluent l'équipe de robots (position et nature des objets détectés, …).

Pour cet objectif, une approche distribuée est mis en œuvre sur une architecture multi-agent de contrôle dans lequel deux types d'agents autonomes sont définis: l'agent utilisateur et l'agent robot. Dans la phase d'ordonnancement, un algorithme génétique à base de règles de priorité (AG) se développe sur chaque agent robot pour les séquences d'opérations allouées.

Afin de valider l'interface de contrôle proposée et analyser ses performances, nous avons réalisé quelques scénarios.

Mots clés : Interface homme/robot, Contrôle collaboratif multi-robots, Systèmes cyber-physiques, Systèmes multi-agents.

Abstract

The objective of this work is to develop a user-friendly Human/Robot Interface (HRI) for collaborative multi-robot control in cyber-physical robotic system (CPRS). Each robot is considered as autonomous and cooperative agent. This HRI has as roles to transmit the human operator intentions to robots (task to be carried out by a single robot, a subset of robots orby all the robots), and to transmit to the operator information issued from various sensors implanted in the cyber-physical environment where evolve the robots team (position and nature of detected objects …). In addition, this work deals with one of the most difficult decision-making problems : tasks planning in multi-robot system. It concerns allocating operations and their sequencing on robots while satisfying precedence constraints and minimizing total makespan. For that purpose, a distributed approach is implemented on a control multi-agents architecture in which two types of autonomous agents are defined: user agent and robots agents.

The validation of the proposed approach and its performances analysis are realized via four scenarios. In the first three scenarios (control ofone robot, a set of robots, all the robots), the user agent sends requests to the agents robots. Each of them verifiesits state and makes a proposal; then, the user agent will react according to the various receivedproposals. If their proposals are accepted, this agent starts execution while respecting precedence constraints. Otherwise, the agent rejects the imposed task. The fourth scenario is mainly realized in two phases: allocationand sequencing. During the allocation phase, a negotiation process based on priority rules is used to guide the robots agents decision-making to distribute operations on the robots agents. During the sequencing phase, a genetic algorithm (GA),basedon priority rules, is developed on each robot agent for the sequencing of the assigned operations.

Keywords: Human/Robot Interface, collaborative multi-robot control, cyber-physical systems, multi-agent systems.

ملخص

هذه الواجهة تقوم بنقل نوايا المستعمل " الانسان" المهام التي سوف يؤديها روبوت واحد او مجموعة فرعية من الروبوتات او كل الروبوتات ، ويحيل الى المشغل المعلومات من مختلف اجهزة الاستشعار المثبة في النظام و طبيعة الاجسام .

ايضا يتناول هذا العمل واحد من مشاكل صنع القرار الاكثر صعوبة و هو التخطيط للمهام في نظام متعدد الروبوتات و القيود و كذالك من اجل تحسين وقت تنفيذ المهام. و من اجل هذا يتم اتباع نهج موزع و هندسة العملاء المتعددين التي نميز فيها نوعان من الوكلاء :وكيل مستخدم و وكيل الروبوتات.

و يتم المصادقة على النهج المقترح عن طريق اربع سيناريوهات. في السيناريوهات الثلاثة الاولى " التحكم في روبوت واحد ، مجموعة من الروبوتات ، كل الروبوتات " . و كيل المستخدم يرسل متطلباته الى وكلاء الروبوتات . كل واحد يراجع حالته الحالية و يقدم اقتراح ثم يقوم وكيل المستخدم بالاختيار حسب متطلباته.

يتحقق السيناريو الرابع على مرحلتين : التخصيص و الجدولة. حيث في مرحلة التخصيص و هي عملية التفاوض على اساس قواعد حلية ، تستخدم لتوجيه قرارات الروبوتات ، اما في مرحلة الجدولة يتم تطوير خوارزمية جينية على اساس قواعد اولوية على عامل روبوت لتسلسل عمليات التخصيص.

كلمات مفتاحية : نظام متعدد الوكلاء، نظام السيبرانية الفيزيائية، واجهة انسان/روبوت، تعاونية تحكم متعدد-روبوتات

Table des matières

Introduction générale 1
Chapitre 01 4
1. Introduction 4
2. Systèmes cyber-physiques (SCP) 4
2.1. Définition 4
2.2. Composant des SCP 5
2.3. Caractéristiques des SCP 6
3. Systèmes robotiques cyber-physiques (SRCP) 7
3.1. Définition 7
3.2. Composants du SRCP 7
3.3. Technologies de communication utilisées dans le SRCP 7
3.3.1. RFID 8
3.3.2. NFC 8
3.3.3. WiFi 9
3.3.4. GPS 11
3.3.5. Zigbee 12
4. Domaines d'application des SRCP 13
4.1. Coopération entre un robot mobile et des caméras d'ambiance pour le suivi de personnes 13
4.2. Assistance basée sur une architecture de commande RFID et une interface EOG sans fil 14
4.3. Ville de robots « Robot Town » 15
4.4. Projet ADREAM 16
4.5. RFID et robots ménagers 17
5. Contrôle d'un Système Robotique Cyber-Physique 17
6. Interfaces homme/robot 18
6.1. Définition 18
6.2. Trio interface homme/robot 19
6.3. Participant d'une IHR 20
6.4. Qualité d'une interface homme/robot 20
6.5. Cycle de développement d'une IHR 21
7.Conclusion 22
Chapitre 02 24
1. Introduction 24

2. Concept d'agent.....24
2.1. Définition.....24
2.2. Propriétés 25
2.3. Typologies des agents.....25
2.3.1. Agents cognitifs délibératifs......26
2.3.2. Agents réactifs.....26
2.3.3. Agents hybrides.....28
3. Systèmes multi-agents (SMA) 28
3.1. Définition.....28
3.2. Interaction dans les SMAs.....28
3.2.1. Communication.....29
3.2.2. Coopération.....29
3.2.3. Coordination.....29
3.2.4. Collaboration.....30
3.2.5. Organisation.....30
3.2.6. Négociation.....30
3.3. Méthodologie de conception des SMA (AUML) 30
3.4. Plateformes de développement des SMA.....31
3.5. Domaines d'application des SMA.....31
3.5.1. Recherche d'informations.....31
3.5.2. Commerce électronique.....32
3.5.3 Robotique distribuée.....32
4. Conclusion.....33
Chapitre 03 35
1. Introduction.....35
2. Description de l'environnement.....35
3. Modélisation de l'IHR.....37
3.1. Spécification des besoins.....37
3.2. Conception.....38
4. Système multi-agents.....39
4.1. Diagramme d'agents.....39
4.2. Spécification des agents.....40

4.2.1. Agent utilisateur.....41
4.2.2. Agent Robot.....41
4.3. Interaction entre les agents.....42
5. Solution proposée.....48
5.1. Premier scénario : Contrôler un seul robot.....48
5.1.1.Algorithmes.....49
5.1.2 Exemple.....53
5.2. Deuxième scénario :Contrôler un sous-ensemble de robots hétérogène.....53
5.2.1.Algorithmes.......53
5.2.2 Exemple.....55
5.3. Troisième scénario :Contrôler tous les robots du système.....55
5.3.1.Algorithmes.....55
5.3.2 Exemple.....56
5.4.Quatrième scénario :Choix autonome.....56
5.4.1. Allocation.....57
5.4.2. Ordonnancement.....62
5.4.3. Exécution.....64
6.Conclusion.....64
Chapitre 04
1. Introduction.....67
2. Environnements de développement.....67
2.1. Architecture physique de déploiement.....67
2.2. Outils et langages de programmation.....68
2.2.1. Plateforme JADE.....68
2.2.2. Langage Java.....70
2.2.3. NetBeans.....70
2.2.4.MySQL.....71
2.3. Configuration de Jade sous NetBeans.....71
3. Mise en œuvre de l'interface proposée.....72
3.1. Main-Container.....72
3.2. Classe Agent.....72
3.3. Agent Utilisateur.....73
3.3.1. Interface Utilisateur.....73
3.3.2. Gestion des tâches.....75

3.3.3. Communication 75
3.3.4. Planification 76
3.4. Agent robot 77
3.4.1. Mise à jour 77
3.4.2. Traitement 77
3.4.3. Allocation des opérations aux robots 78
3.4.4. Ordonnancement en utilisant les algorithmes génétiques (AG) 79
5.Affichage du plan d'opérations 79
5.1. Configuration 79
5.2. Description 80
5.3. Exécution 80
6.Conclusion 80
Chapitre 05 82
1. Introduction 82
2. Présentation de l'interface homme/robot de contrôle 82
2.1. Interface d'authentification et de contrôle 82
2.2. Interface menu principal 84
2.3. Interface gestion des environnements 84
2.4. Interface de configuration 85
2.5. Interface de l'historique 85
3. Déroulement de l'application développée 86
3.1. Interface utilisateur 86
3.2. Lancement de la plateforme JADE avec les agents 86
4. Scénarios de validation 88
4.1. Scénario 1 :contrôler un seul robot 88
4.1.1.Tâche acceptée 88
4.1.2. Tâche rejetée 90
4.2. Scénario 2 : Contrôler un sous-ensemble de robots hétérogènes 90
4.2.1.Tâche acceptée 90
4.2.2. Tâche rejetée 93
4.3. Scénario 3 : Contrôler tous les robots 94
4.2. Scénario 4 : Choix autonome 97
5. Conclusion 97
Conclusion générale 99

Liste des tableaux

Tableau 1 : Agents cognitifs vs réactifs.....26
Tableau 2 :Description d'interaction entre les agents (contrôler un sous-ensemble de robots) 46
Tableau 3 : Description d'interaction entre les agents (contrôler tous les robots) 47
Tableau 4 : Contraintes temporelle et de précédence entre les tâches.....64
Tableau 5 : Les nouveaux robots consistés dans le scénario.....86
Tableau 6 : Les anciens robots consistés dans le scénario. 86

Listes des figures

Figure 1 : Les trois éléments clés d'un SCP 6
Figure 2 : Schéma de SCP 7
Figure 3 : Fonctionnement de système RFID 9
Figure 4 :Principe de la communication NFC 10
Figure 5 : Composants de GPS *[14]*. 12
Figure 6 : Équipements de protocole Zigbee [10]. 13
Figure 7 : Plateforme perceptuelle, caméras d'ambiance et robot mobile 14
Figure 8 : Environnement du robot 15
Figure 9 : Ville de robots 16
Figure 10 : Vue d'intérieur du bâtiment ADREAM 17
Figure 11 :Approchement du robot PR2 aux objets 18
Figure 12 : Trio d'une interface homme/robot 20
Figure 13 : Cycle de vie d'une IHR. 21
Figure 14 : Agent cognitif 25
Figure 15 : Agent réactif 26
Figure 16 : Représentation d'un agent en interaction avec son environnement et les autres agents 27
Figure 17 : Communication directe et indirecte 28
Figure 18 : Vue global du système robotique cyber-physique considéré 37
Figure 19 : Diagramme de cas d'utilisation global 38
Figure 20 : Diagramme du cas d'utilisation « Initialiser une tâche » 38
Figure 21 : Diagramme de classes 39
Figure 22 : Diagramme d'agents du système 40
Figure 23 : Structure interne de l'agent Utilisateur 41
Figure 24 : Structure interne de l'agent Robot 42
Figure 25 : Diagramme de séquence d'interaction entre les agents pour contrôler un seul robot 44
Figure 26 : Diagramme de séquence d'interaction entre les agents pour contrôler un sous-ensemble de robots 45
Figure 27 : Diagramme de séquence d'interaction entre les agents dans le cas de contrôle de tous les robots 46
Figure 28 : Diagramme de séquence d'interaction entre les agents dans le cas choix autonome 48
Figure 29 : Prise de décision du processus d'agents robots lors de l'étape opération affectation. 58
5.4.2. Ordonnancement 62

Figure 30 :Ordonnancement. 63
Figure 31 :Mutation. 63
Figure 32 : Diagramme de déploiement du système. 68
Figure 33 : Plateforme et conteneurs de JADE [32]. 69
Figure 34 : Interface utilisateur (GUI) [32]. 69
Figure 35 :Configuration de NetBeans avec Jade. 71
Figure 36 :Création du Main-Container 72
Figure 37 : Création de l'agent Utilisateur 72
Figure 38 :Envoi des informations à l'agent utilisateur 73
Figure 39 :Récupération des informations 74
Figure 40 :La méthode d'affichage 74
Figure 41 :La commande d'affichage. 74
Figure 42 :Gestion des tâches. 75
Figure 43 :Communication. 76
Figure 44 :Planification. 77
Figure 45 :Mise à jour. 77
Figure 46 :Traitement. 78
Figure 47 :Allocation des différentes opérations aux agents robots 79
Figure 48 : Ordonnancement 79
Figure 49 :Diagramme de Gantt. 80
Figure 50 : Authentification. 83
Figure 51 : Interface de contrôle. 84
Figure 52 : Interface menu principale. 84
Figure 53 : Interface gestion des environnements. 85
Figure 54 : Interface configuration. 85
Figure 55 : Interface historique. 86
Figure 56 : Lancement de la plateforme JADE avec les agents. 87
Figure 57 : Initialisation de la tâche « Pick-and-place » dans le cas de « contrôler un seul robot ». 89
Figure 58 : Résultats obtenus pour la tâche « Pick-and-place » acceptée et exécutée par un seul robo.90
Figure 59 : Résultats obtenus pour la tâche « Déplacer-objet » rejetée par un seul robot. 90
Figure 60 : Initialisation de la tâche « Transporter-objet »acceptée et exécutée par un sous-ensemble de robots hétérogènes. 91
Figure 61 : Résultats obtenus pour la tâche « Transporter-objet » acceptée et exécutée par un sous-ensemble de robots hétérogènes 93
Figure 62 : Résultats obtenus pour la tâche « Transporter-objet » rejetée par un ou plusieurs robots...94
Figure 63 : Résultats obtenus pour la tâche « Transporter-objet » acceptée et exécutée par tous les robots du système. 97

Liste des acronymes

SCP : Système cyber-physique.

SRCP : Système robotique cyber-physique.

SMA : Système multi-agent.

SRMA : Système robotique multi-agent.

AG : Algorithme génétique.

RP : Règle de priorité.

IHR : Interface homme/robot.

RFID : Radio Frequency IDentification

NFC : Near Field Communication

WiFi : Wireless Fidelity

GPS : Global *Positioning* System

EOG : L'électro-oculographie

ADREAM : Architectures Dynamiques Reconfigurables pour les systèmes Embarqués Autonomes Mobiles

RSSI : The Received Signal Strength Indicator

UML : Unified Modeling Language

AUML : Agent Unified Modeling Language

CORMAS : COmmonResources Multi-Agent System

AR : Agent robot

JADE : Java Agent DEvelopment Framework

FIPA : Foundation for Intelligent Physical Agents

RMI : Remote Method Invocation

AMS : Agent Management System

DF : Directory Facilitator

ACC : Agent Communication Channel

Introduction générale

La robotique autonome devient de plus en plus importante dans le domaine de la recherche ; elle fait face à une croissance rapide dans la complexité des besoins et des exigences pour des robots chargés de tâches multiples, capables de se coordonner, et développés de telle manière que des garanties de sûreté et de sécurité puissent être vérifiées et certifiées pour remplacer l'homme [1]. De ce fait, les chercheurs essayent actuellement d'introduire les systèmes cyber-physiques dans la robotique.

Le terme « Systèmes Cyber-Physiques (SCP) » a été effectivement inventé par Helen Gill à la « National Science Foundation » en 2006 aux États-Unis. Elle a défini le SCP comme « des systèmes physiques, biologiques et d'ingénierie dont les activités sont intégrées, suivies, et contrôlés par un noyau de calcul. Les composants sont mis en réseau à toutes les échelles ». Elle a mentionné aussi que « l'informatique est profondément ancrée dans chaque composant physique, peut-être même dans les matériaux. Le noyau de calcul est un système embarqué, généralement qui exige une réponse en temps réel, et est le plus souvent distribué ».

Les systèmes robotiques de contrôle ont récemment évolué vers des systèmes de contrôle embarqués et coopératifs appelés Systèmes Robotique Cyber-Physiques (SRCP). Ces systèmes constituent des systèmes autonomes de perception, d'analyse et de commande des mondes physiques réels par des traitements informationnels adaptatifs et distribués.

Il existe plusieurs approches de contrôle des SRCP. Les plus intéressantes sont issues de l'Intelligence Artificielle Distribuée (IAD) qui permettent de contrôler un système en distribuant les connaissances et les tâches sur plusieurs entités et qui permettent l'interaction entre les entités de système lors du traitement. Parmi ces approches, on distingue les approches basées sur les systèmes multi-agents (SMA).

Le problème traité dans ce travail concerne l'interaction homme/robot (ou homme/équipe de robots) dans un système cyber-physique. Les principaux aspects sont les suivants : Comment interagir avec les robots ? Comment transmettre les intentions de

l'homme aux robots ? Comment afficher l'état de l'environnement de travail et des robots ?Comment afficher les résultats finaux obtenus ?

Dans ce contexte, nous nous fixons comme objectif le développement d'une Interface Homme/Robot (IHR) conviviale destinée au contrôle collaboratif multi-robots dans un SRCP.

Transmettre les intentions de l'opérateur humain aux robots (tâches à exécuter par un seul robot, un sous-ensemble de robots, tous les robots du SRCP).

Transmettre à l'opérateur les informations issues des différents capteurs équipant l'équipe de robots hétérogènes (caméras, capteurs de proximité, …).

Transmettre à l'opérateur les informations issues des différents capteurs implantés dans l'environnement cyber-physique où évoluent l'équipe de robots (position et nature des objets détectés, …).

Transmettre les feedbacks sur l'exécution des différentes opérations et tâches confiées au SRCP.

Pour assurer une meilleure présentation du travail effectué et garantir la clarté du mémoire, outre cette introduction générale, ce manuscrit se compose de quatre chapitres. Chacun met en évidence une contribution particulière du travail :

- Dans le premier chapitre de ce mémoire, nous présentons l'état de l'art des systèmes robotiques cyber-physiques (SRCP) et des interfaces homme/robot (IHR) de contrôle, leurs domaines d'applications ainsi que les technologies utilisées pour la communication dans les SRCP.
- Le deuxième chapitre est consacré à l'étude conceptuelle de notre solution. Nous présentons les différentes composantes du système de contrôle et de l'IHR, les comportements des agents qui y évoluent ainsi que les différentes interactions entre ces derniers.
- Le troisième chapitre décrit l'implémentation de notre solution 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 considérées.
- Dans le quatrième chapitre, nous évaluons la solution proposée via différents scénario de validation.

Chapitre 01

Systèmes robotiques cyber-physiques

Chapitre 01

Systèmes robotiques cyber-physiques

1. Introduction

Le déploiement des robots dans un environnement humains, répond à un enjeu sociétal majeur. Il s'agit de voir ces robots interagir de façon naturelle avec les humains et les autres entités physiques. Dans ces enivrements, l'intégration des robots dans telle environnements combine les systèmes informatiques et le réseau avec les entités physiques, ceci est appelé système robotique cyber-physique (SRCP). Ce type de système utilise plusieurs technologies pour assurer l'identification des entités intégrées et la communication telle que (RFID,NFC, Zigbee, WIFI, ...).

Ce chapitre a pour objectif de répertorier les informations fondamentales permettant de se familiariser avec les différentes notions et concepts liés aux systèmes robotique cyber-physique (SRCP). Dans une première partie, nous donnons quelques définitions relatives aux systèmes cyber-physiques. Une deuxième partie aura pour but de décrire le concept de robots dans un SRCP. Dans la troisième partie, nous exposerons les domaines d'applications les plus connus des SRCP, ainsi que les technologies utilisées. Enfin, le chapitre donne une description sur les interfaces homme/robot dans la quatrième partie.

2. Systèmes cyber-physiques (SCP)

2.1. Définition

Le terme «système cyber-physique» a été effectivement inventé par Helen Gill à la « National Science Foundation » en 2006 aux États-Unis, Elle a défini le SCP comme « des

systèmes physiques, biologiques et d'ingénierie dont les activités sont intégrées, suivies, et contrôlés par un noyau de calcul. Les composants sont mis en réseau à toutes les échelles ». Elle mentionné aussi que « l'informatique est profondément ancrée dans chaque composant physique, peut-être même dans les matériaux. Le noyau de calcul est un système embarqué, généralement qui exige une réponse en temps réel, et est le plus souvent distribué ».

Le mot SCP est utilisé dans plusieurs domaines, ce qui fait que plusieurs définitions lui sont attachées. En informatique, le SCP est défini de manières différentes; selon [2], « les systèmes cyber-physiques sont l'intégration de calcul et de processus physiques (Ordinateurs et réseaux embarqués sont intégrés pour surveiller et contrôler les processus physiques) ». D'après [3], « les systèmes cyber-physiques sont des systèmes physiques et d'ingénierie dont les opérations sont surveillées, coordonnées, contrôlées et intégrées par un noyau de calcul et de communication ».

2.2. Composant des SCP

La figure 1 ci-après présente les composants clés d'un SCP ; ils peuvent être identifiés comme suit [4] :

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

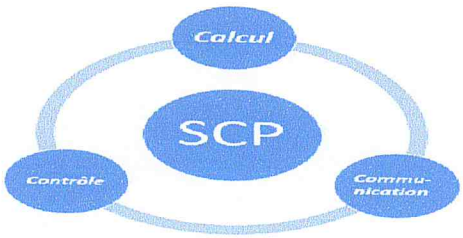


Figure 1 : Les trois éléments clés d'un SCP [4] .

La figure 2 ci-dessous représente 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.

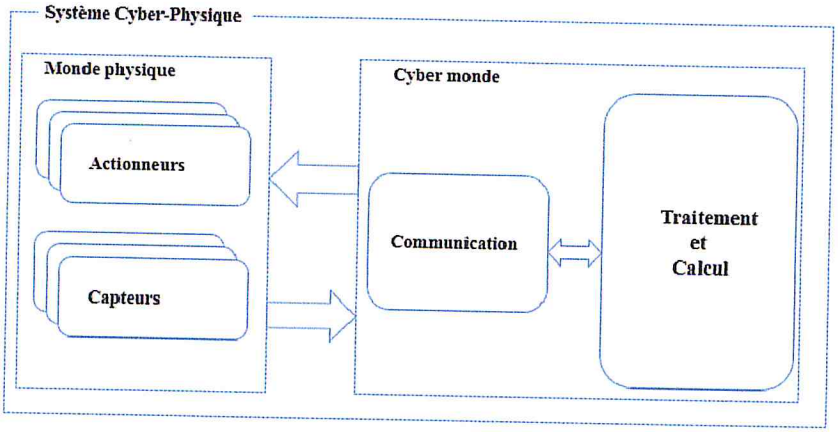


Figure 2 : Schéma de SCP [4].

2.3. Caractéristiques des SCP

Un SCP est un système logiciel intégré dans des dispositifs, bâtiments, moyens et voies de transport, systèmes de production, procédés médicaux, processus logistiques, processus de coordination, processus de gestion, etc. Un SCP permet de [5] [2] :

- Enregistrer directement les données physiques à l'aide de capteurs et influer sur les processus physiques à l'aide des actionneurs.
- Évaluer et enregistrer les données, et interagir de façon active ou réactive avec le monde physique et numérique.
- Relier avec d'autres SCP et dans les réseaux mondiaux via les moyens de communication numériques (sans fil et/ou câblés, locaux et/ou mondiaux).
- Utiliser les données et les services disponibles à l'échelle mondiale.
- Comporter une série d'interfaces homme/machine dédiées.
- Entrée et feedback possible issues de l'environnement physique.
- Contrôle distribué.
- Performances temps-réel.

Généralement, les SCP apportent de nombreux avantages [3] :

- Construire des systèmes plus sûrs et plus efficaces.
- Réduire le coût des systèmes de construction et d'exploitation.

- Construire des systèmes complexes qui fournissent de nouvelles capacités.
- Baisser les coûts de calcul, de réseautage et de détection.
- Les ordinateurs et la communication sont omniprésents, ce qui permet de développer des SCP à une échelle nationale ou mondiale.

3. Systèmes robotiques cyber-physiques (SRCP)

3.1. Définition

La notion du SRCP n'est pas simple à définir ; on peut définir les SRCP comme suite : « Les systèmes robotiques cyber-physiques sont l'intégration de calcul informatique avec les robots, c.-à-d. intégrer les ordinateurs et le réseau pour surveiller et contrôler un ensemble de robot autonomes dans un environnement intelligent. Les robots peuvent ainsi coopérer et communiquer avec d'autres robots pour accomplir des tâches en utilisant des technologies modernes (RFID, NFC, etc.) ».

3.2. Composants du SRCP

Un système robotique cyber-physique se compose des entités suivantes :

- **Entités physiques :** incluent les opérateurs humains, les objets et les robots, qui sont équipés de capteurs et tags pour faciliter la perception des robots et l'identification des objets.
- **Système informatique :** c'est la plateforme de traitement des données et de contrôle des robots. Il consiste en des ordinateurs qui sont intégrés dans les robots. Le but de ce composant est d'assurer l'efficacité des tâches et la réaction des robots pour effectuer la bonne tâche au bon moment.
- **Communication :** c'est le mécanisme de communication entre les différents robots (wifi, Zigbee, Bluetooth...). Cela assure l'échange d'informations entre les robots pour la coordination et la coopération entre eux.

3.3. Technologies de communication utilisées dans le SRCP

3.3.1. RFID

Le système RFID (Radio Frequency IDentification), 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. Un système RFID est composé de deux entités qui communiquent entre elles [6] :

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

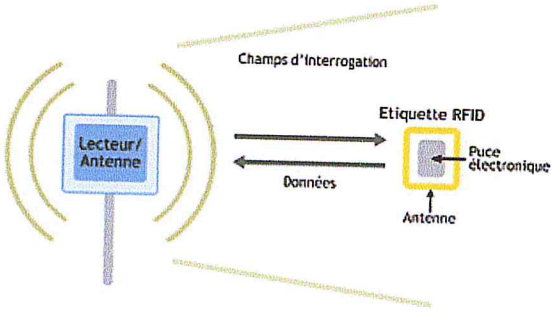


Figure 3 : Fonctionnement de système RFID [6].

Comme il est présenté 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 [7].

La plupart des SCP se composent de nombreux lecteurs RFID et des systèmes de traitement supplémentaires, comme un middleware RFID et une application [8].

3.3.2. NFC

C'est une technologie de communication de proximité (quelques centimètres) lancée par Sony et Philips (Figure 4). Le NFC (Near Field Communication) ou communications en champ proche, permet d'échanger des données entre un lecteur et n'importe quel terminal mobile ou entre les terminaux eux-mêmes et ce, à un débit maximum de 424 Kbits/s [10].

Les utilisations sont multiples : un Smartphone peut se connecter à un ordinateur pour télécharger un fichier, un téléviseur échangera des données avec un Smartphone… Au-delà,

ce sont les marchés du contrôle d'accès, des transactions en magasin ou des bornes interactives de tout type qui sont concernés [10].

Un des autres atouts du NFC, par rapport à la technologie Bluetooth notamment, réside dans les caractéristiques mêmes des puces NFC : de taille très réduite, elles sont conçues pour qu'un lecteur puisse dialoguer avec plusieurs d'entre elles de manière simultanée, sans risque de collision. Enfin, dernière promesse de la technologie : permettre le paiement sécurisé, via encodage et chiffrement embarqués [10].

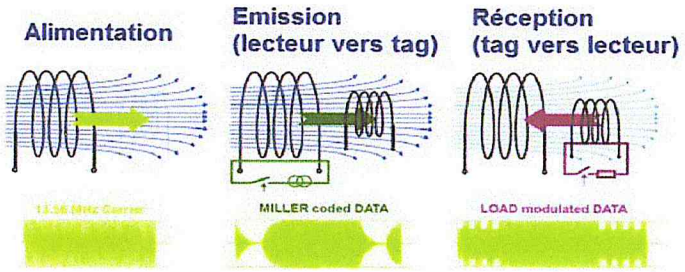


Figure 4 :Principe de la communication NFC [10].

Le NFC est une technologie sans contact qui permet de transmettre des informations digitales à un support mobile et qui fonctionne à proximité (environ 10 centimètres maximum) et à haute fréquence. Équipée sur un Smartphone, elle peut être utilisée selon différents modes (Mode émulation de carte, Mode lecteur, Mode pair-à-pair) [11].

Avantages :

- Utilisation simple et pratique.

Inconvénient :

- Sécurité de données privées.
- Rapidité : Plus lent que le Bluetooth qui a un débit pouvant aller jusqu'à 24 Mbit/s et est de manière générale de 700 Kbits/s.
- Stockage : Le système NFC permet d'échanger des données à un débit maximum de 424 Kbits/s et pas plus.

3.3.3. WiFi

Wireless Fidelity, Protocole de communication permettant de connecter des machines dans un réseau informatique, sans qu'elles soient connectées à l'aide de câbles réseau [12]. Il existe deux modèles de déploiement :

- Le mode infrastructure : c'est un mode de fonctionnement qui permet de connecter les ordinateurs équipés d'une carte réseau WiFi entre eux via un ou plusieurs points d'accès qui agissent comme des concentrateurs. Il est essentiellement utilisé en entreprise. La mise en place d'un tel réseau oblige de poser à intervalle régulier des points d'accès dans la zone qui doit être couverte par le réseau.
- Le mode Ad-Hoc : c'est un mode de fonctionnement qui permet de connecter directement les ordinateurs équipés d'une carte réseau WiFi, sans utiliser un matériel tiers tel qu'un point d'accès. Ce mode est idéal pour interconnecter rapidement des machines entre elles sans matériel supplémentaire [13].

Les principaux avantages et inconvénients à déployer un réseau sans fil WiFi sont donnés comme suit :

Avantages :

- Mobilité : les utilisateurs sont généralement satisfaits des libertés offertes par un réseau sans fil et de ce fait sont plus enclins à utiliser le matériel informatique.
- Facilité et souplesse : un réseau sans fil peut être utilisé dans des endroits temporaires, couvrir des zones difficiles d'accès aux câbles, et relier des bâtiments distants.
- Coût : si leur installation est parfois un peu plus coûteuse qu'un réseau filaire, les réseaux sans fil ont des coûts de maintenance très réduits ; sur le moyen terme, l'investissement est facilement rentabilisé.
- Évolutivité : les réseaux sans fil peuvent être dimensionnés au plus juste et suivre simplement l'évolution des besoins [13].

Inconvénients :

- Qualité et continuité du signal : ces notions ne sont pas garanties du fait des problèmes pouvant venir des interférences, du matériel et de l'environnement.
- Sécurité : la sécurité des réseaux sans fil n'est pas encore tout à fait fiable du fait que cette technologie est novatrice [13].

3.3.4. GPS

Le GPS Global Positioning System, système de localisation mondial, est actuellement le système de repérage le plus utilisé dans le monde. Le GPS se compose de trois groupes d'éléments (figure 5) [14] : des satellites en orbite autour de la Terre; des stations de contrôle

au sol et Des récepteurs (puces) GPS des utilisateurs. Les satellites GPS émettent des signaux qui sont captés et identifiés par les récepteurs. Ces derniers peuvent alors situer précisément en trois dimensions le point voulus en temps réel.

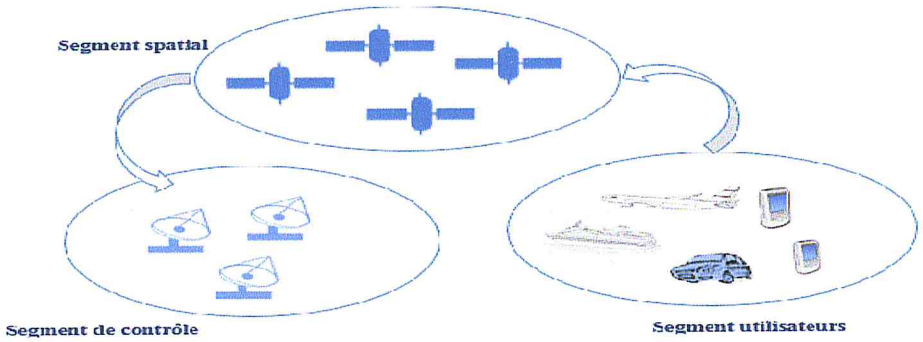


Figure 5 : Composants de GPS [14].

3.3.5. Zigbee

ZigBee est un protocole de haut niveau permettant la communication de petites radios, à consommation réduite, basée sur la norme IEEE 802.15.4 pour les réseaux à dimension personnelle (Wireless Personal Area Networks WPAN). Un système ZigBee définit trois types d'équipements (figure 6) [15] :

- Les FFD (Full Function Devices), équipements à fonctionnalité complète : routeurs ou dispositifs reliés à un capteur, ils coordonnent l'ensemble du réseau.
- Les RFD (Reduce Function Devices), équipements à fonctionnalité réduite : sont conçus pour des applications simples comme l'allumage d'une lampe.
- Les coordinateurs de réseau.

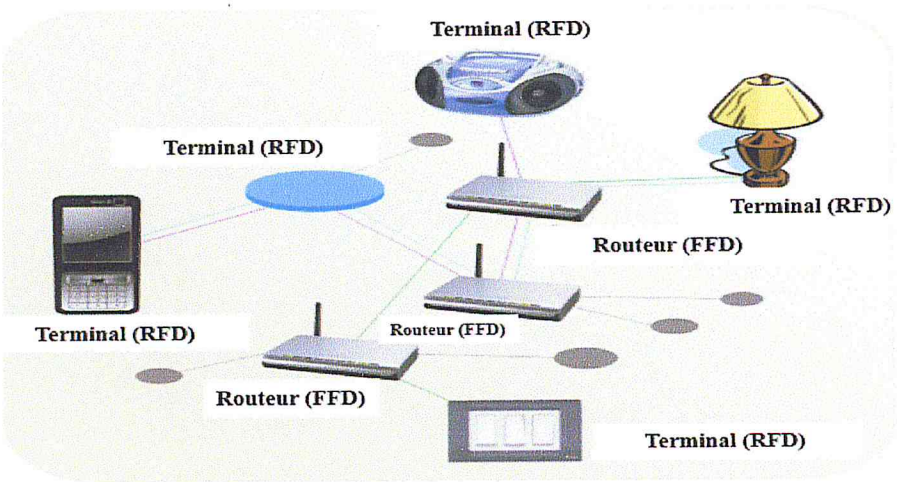


Figure 6 : Équipements de protocole Zigbee [10].

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.

4. Domaines d'application des SRCP

En dépit de son jeune âge, la science des SCP a déjà fourni des résultats spectaculaires et concrets dans la robotique. Dans ce que suit, nous en citons les plus importants :

4.1. Coopération entre un robot mobile et des caméras d'ambiance pour le suivi de personnes

L'université de Toulouse et le CNRS ont proposé un système de coopération entre les caméras d'ambiance et des capteurs embarqués dans un robot mobile. L'objectif étant de permettre aux robots d'interagir de façon naturelle avec les humains dans un environnement. Cette stratégie de perception repose à la fois sur des capteurs embarquées (laser SICK, lecteur RFID) et deux caméras d'ambiance. Les objectifs principaux de ce travail sont données comme suit :

- Suivre une personne, identifiée par un tag/badge radio fréquence.
- Faciliter la navigation de robots en présence d'un passant lors de l'exécution de sa tâche.

Ce système est composé d'un robot mobile et de deux caméras fixées sur le mur. Ces caméras sont connectées à un PC via une connexion FireWire. Le robot est de type iRobot B21r, embarquant divers capteurs tels que (laser SICK, lecteur RFID, caméra). Ce robot embarque également deux PC. La communication entre le robot et les PC est assurée par wifi. Elle s'appuie sur une détection/reconnaissance faciale, combinées à la détection RF, pour reconnaître et suivre la personne cible. La figure 7 illustre le système complet [17].

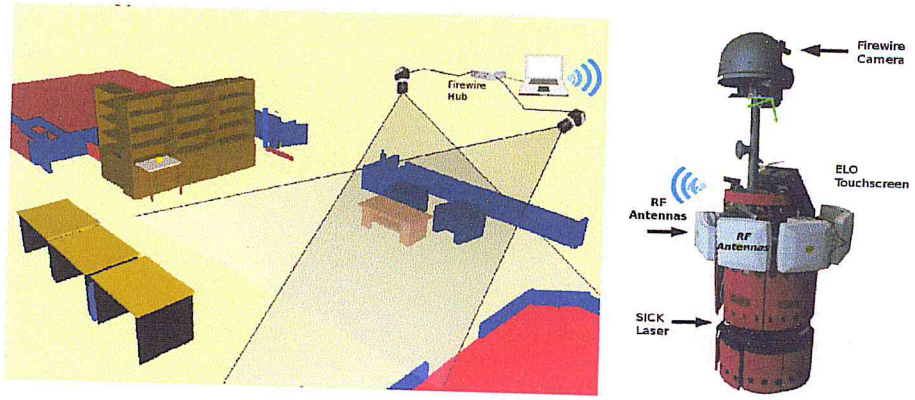


Figure 7 : Plateforme perceptuelle, caméras d'ambiance et robot mobile [17].

4.2. Assistance basée sur une architecture de commande RFID et une interface EOG sans fil

Les auteurs dans [18] décrivent une application de robot d'assistance qui combine une interface sans fil portable basée sur l'électro-oculographie (EOG) et l'identification par radio fréquence (figure 8). Cette application est destinée aux utilisateurs handicapés qui souffre d'un handicap moteur sévère. Dans l'EOG, le comportement de l'œil est considérée comme un dipôle électrique. Pour obtenir les signaux de l'utilisateur, ils ont utilisé 4 électrodes sèches et un cinquième servira comme référence. Pour enregistrer les signaux électroniques, ils ont utilisés des périphériques basés sur EOG. L'appareil envoie les signaux enregistrés via USB. Pour recevoir la direction du mouvement de l'œil et son clignote, il est nécessaire de traiter les signaux d'EOG avec un algorithme de traitement. Pour générer une commande oculaire, l'utilisateur doit effectuer un mouvement oculaire rapide vers la direction souhaitée, puis revenir au centre. Avant de travailler avec l'interface, chaque utilisateur doit suivre une formation. Pour le stockage des informations dans les balises et l'identification, ils ont utilisé la RFID; cette technologie dans ce cas permet au robot de ramasser le bon objet.

Avantages

- l'utilisation des électrodes ne nécessites pas de mettre un gel dans la peau avant les placer.
- l'utilisateur a plus de mobilité.
- l'algorithme de traitement peut détecter les quatre directions ainsi que le clignote de l'œil effectué par l'utilisateur.
- utilisation confortable et ergonomique de l'oculaire.

Inconvénients

- Ya des limites lorsqu'il ya des objets placés dans des plans différents.

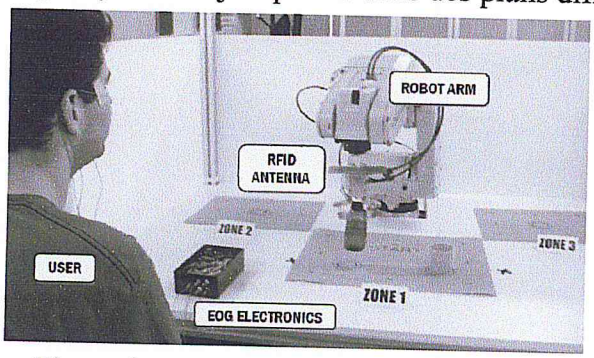


Figure 8 : Environnement du robot [18].

4.3. Ville de robots « Robot Town »

Les auteurs de [19] décrivent un système multi-robots dans un environnement réel. Cet environnement « Ville de robot » est basé sur le partage d'information. Pour exécuter la tâche demandé, ce system est composé de quatre éléments nécessaires qui collaborent entre eux (figure 9).

Le système se compose de robots autonomes et intelligents. Les caméras de vision sont distribuées et mises en place dans un bloc d'une ville pour observer et mesurer des objets en mouvement, De plus, les résultats sont envoyés aux robots dans un temps précis pour qu'ils puissent planifier et exécuter la tâche ; les robots sont reliés à un système de gestion via un réseau. Les étiquette RFID sont fixées sur des objets de sorte que les robots localisent l'environnement et reconnaissent les objets existants dans leur environnement, les tags sont aussi attachés et distribués aux murs, portes, etc. Le système de gestion qui collecte les informations à partir des caméras, unifie les résultats d'observation et les mets à jour pour fournir aux robots ce qu'ils doivent faire [19]. Des capteurs GPS ont été utilisés aussi pour permettre aux robots de se localiser précisément dans leur environnement.

Pour l'interaction avec les robots et la gestion de données, les auteurs ont développé à un système appelé « système de gestion de ville (TMS) ». TMS récupère les données, l'intègre à la base de données, et fournit les informations aux robots en temps réel. Les fonctions suivantes sont mises en œuvre dans le TMS :

- communication avec les capteurs et les robots.
- stockage, révision et récupération des données de l'environnement en temps réel.
- fournir les données relatives à un tag RFID intégré dans l'environnement.
- fournir un tag RFID relié aux données.

- maintenir l'intégrité de multiples données obtenues par les différents capteurs dans l'environnement.
- informer les robots de l'occurrence de certains événements prédéfinis dans l'environnement.
- soutenir l'acquisition des données à la fois à la demande et les réponses aux requêtes.

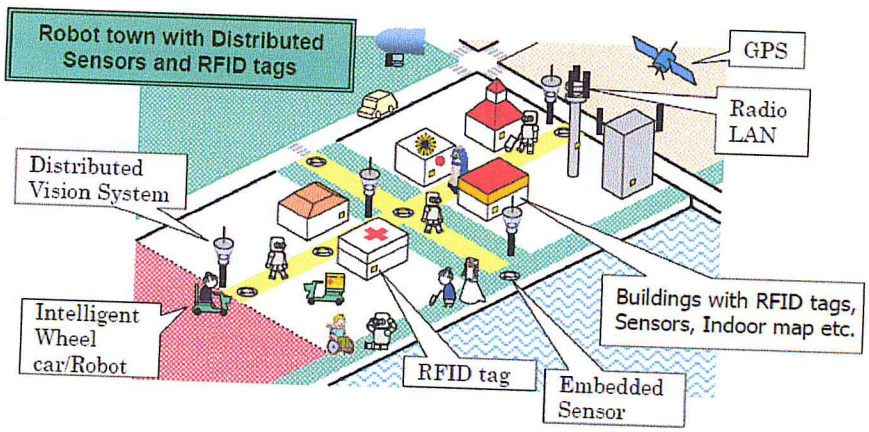


Figure 9 : Ville de robots [19].

Avantages

- effectuer des activités robotiques autonomes dans un tel environnement.
- réduire raisonnablement le coût de construction initiale et le coût de fonctionnement
- préserver un environnement naturel et confortable pour la vie humaine ordinaire.
- assurer l'exactitude des tâches.

4.4. Projet ADREAM

L'un des grands projets en Europe dans les SRCP est le projet ADREAM [20] (Architectures Dynamiques Reconfigurables pour les systèmes Embarqués Autonomes Mobiles) qui va être finalisée dans les prochaines années par le LAAS-CNRS [21] [22]. Ce dernier est dédié à des travaux de recherche sur les SCP, l'intelligence ambiante et la robotique d'assistance.

ADREAM vise à bâtir les technologies et les approches nécessaires à la mise en réseau massive d'objets intelligents munis des dispositifs RFID communicants fixes, portés par l'homme, portés par des objets d'intérêt ou embarqués sur des robots mobiles. L'objectif est d'estimer, à tout instant, l'état de l'environnement, d'interpréter les situations et de générer des actions appropriées. Il s'agit, donc, de faire communiquer des machines entre elles

(robots, téléphones portables, ordinateurs, mobiliers, ...) qui détecteront toutes sortes d'informations qu'elles seront capables de traiter dans le but de faciliter la vie des utilisateurs. Les chercheurs du projet ADREAM travaillent par exemple sur des robots qui peuvent porter assistance aux personnes âgées ou handicapées, être capables de détecter leurs chutes et contacter les secours [20].

Aussi, le bâtiment ADREAM, d'une surface de 1700 m2, est pourvu d'une façade et d'un toit munis de panneaux photovoltaïques pour assurer son autonomie énergétique et permettre des travaux sur de nouveaux systèmes de gestion de l'énergie.

Figure 10 : Vue d'intérieur du bâtiment ADREAM [14].

4.5. RFID et robots ménagers

En 2014, l'équipe de Georgia Tech a implémenté des étiquettes RFID sur plusieurs types d'objets (flacons de médicaments, télécommandes de téléviseurs, téléphones, etc.). Ils ont également équipé un robot PR2 de Willow Garage de deux antennes RFID placées sur ses épaules. Ces antennes peuvent détecter jusqu'à 150 tags par seconde et à une distance de 3m dans des conditions pratiques ou dépassant 6m sous la ligne idéale de conditions de visibilité. L'objectif est de permettre aux robots de trouver et accéder à des objets quotidiens dans l'environnement. En tant que tel, la position d'un objet étiqueté dans son environnement peut changer de façon inattendue.

Ils ont formulé la recherche et la navigation vers un objet étiqueté comme un problème d'optimisation où le robot doit trouver la position d'une antenne directionnelle qui maximise

le RSSI (The Received Signal Strength Indicator) d'une étiquette avec un ID correspondant à l'objet cible.

Enfin, ils ont démontré que cette approche peut permettre à un PR2 de trouver et accéder à différents objets marqués dans une vraie maison. Grâce à des protocoles d'anti-collisions entre le lecteur et les étiquettes, une centaine de balises peuvent coexister dans l'environnement sans interférence [23].

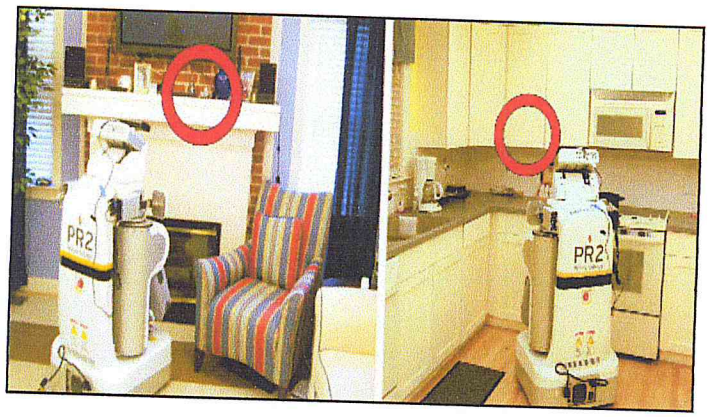


Figure 11 :Approchement du robot PR2 aux objets [23].

5. Contrôle d'un Système Robotique Cyber-Physique

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

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 l'intégration des Systèmes Multi- agents (SMA) dans les systèmes cyber-physiques.

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 classe de systèmes.

Le plus difficile et complexe est de contrôler tout sa via une interface graphique : Comment transmettre les intentions de l'opérateur pour contrôler ces entités hétérogènes, Comment collaborer entre ces entités pour effectuer des tâches très complexes et obtenir des meilleurs résultats par rapport aux technologies et travaux précédents.

6. Interfaces homme/robot

L'utilisation des robots par l'homme, et donc le rapport homme/robot, a évolué depuis les origines de la robotisation au gré des ruptures technologiques, avec des applications et un degré d'aboutissement qui varient selon le milieu (terre/air/mer). Une IHR peut signifier :

- Interface Homme/Robot.
- Interactions Homme/Robot.
- Communication Homme/Robot.
- Dialogue Homme/Robot.
- Interaction Personne/Robot.

6.1. Définition

Selon [24], une Interface homme/robot et défini comme suit :« Afin de commander efficacement un robot, l'interface homme/robot doit fournir des outils pour percevoir l'environnement, de prendre des décisions, et de générer des commandes ».

6.2. Trio interface homme/robot

On peut définir une IHR comme suite « L'Interface Homme/Robot (IHR) permet de faciliter la communication d'un utilisateur ordinaire avec le robot. En effet, pour donner l'ordre au robot d'exécuter une tâche bien définie, l'opérateur n'aura qu'à appuyer sur le bouton spécifique de l'interface ». Le trios interface homme/robot et compose de (figure 12) :

- Homme : C'est l'élément essentiel (prioritaire) du trio interface homme/robot (le robot est au service de l'humain et non le contraire).
- Robot : Son rôle est d'obéir aux consignes.
- Interface : caractérise différentes manières de dialoguer entre un humain et un robot.

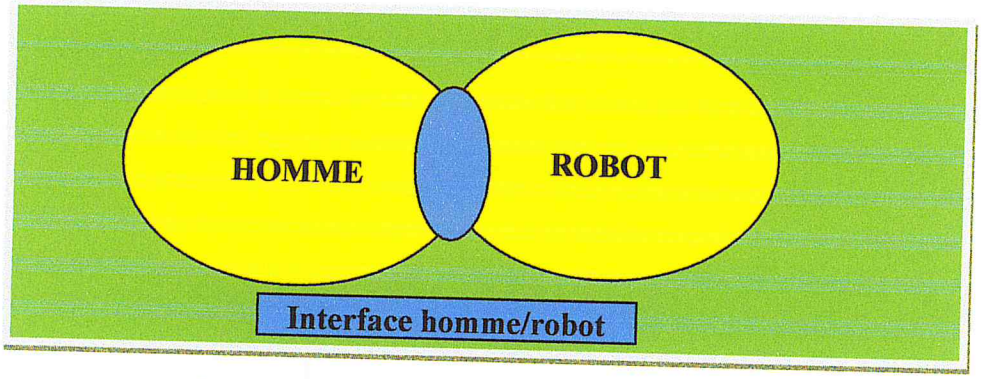


Figure 12 : Trio d'une interface homme/robot

6.3. Participant d'une IHR

Il existe quatre types de participant dans la conception d'une IHR :

- Utilisateur : participant avec choix.
- Machine (ordinateur) : participant avec programme.
- Robot : participant avec exécution des consignes.
- Concepteur : participant qui anticipe les choix possibles de l'utilisateur et les code dans un programme.

6.4. Qualité d'une interface homme/robot

D'après les travaux réalisés dans [25] [26] [27], on n'a déduit que les principales qualités d'une IHR sont les suivantes :

- Visibilité : permet à l'utilisateur de voir ce qu'il peut faire avec l'interface et de voir s'effectuer son propre travail.
- Transparence : ce qui se passe réellement dans le système informatique est caché pour l'utilisateur.
- Intuitivité : le système réalise ce que l'utilisateur pense naturellement ou naïvement qu'il va faire.
- Cohérence (consistance) : prévisible quel que soit le contexte ou l'application.
- Contrôle : retours clairs et concis sur chaque action.
- Intégrité : préserve des données et des résultats acquis par l'utilisateur.
- Automaticité : les tâches répétitives doivent pouvoir s'automatiser.

- Concision : au niveau des commandes de l'utilisateur (abréviations) et au niveau des présentations d'écran (ne doivent fournir que les seules informations utiles).
- Bonne présentation des écrans : les écrans doivent avoir une bonne apparence, pas trop chargée, claire et bien ordonnée.

6.5. Cycle de développement d'une IHR

La clé du succès du développement des interfaces utilisateur réside dans la mise en place d'un processus itératif avec la création de maquettes qui seront soumises à l'appréciation des utilisateurs puis corrigées/adaptées en fonction des résultats de ces tests [28] (Figure 13) :

- **Définition des besoins :**La définition des besoins et des exigences correspond à l'étape dans laquelle nous discutions avec les futurs utilisateurs afin de comprendre de quoi ils ont besoin : QUI doit pouvoir faire QUOI ? Lors de cette étape, nous définissions également les demandes précises, telles que le respect de certaines normes graphiques, les temps de réponse, le matériel sur lequel l'application devrait fonctionner, etc.
- **Analyse :** l'analyse du système permet d'affiner ce qui a été définie dans l'étape précédente [28]. On y détaille davantage le fonctionnement interne de la future interface (COMMENT cela doit-il fonctionner ?).
- **Conception :** la conception du système correspond à la définition des choix techniques.
- **Implémentation :** la programmation est l'étape dans laquelle les informaticiens se donnent à cœur joie ! Ils réalisent l'application à l'aide de langages de programmation, de systèmes de gestion de bases de données, etc.
- **Test :** Durant les tests, les informaticiens vérifient que l'application fonctionne et répond aux besoins définis au début. Cette phase de tests peut intégrer des validations de l'application avec les utilisateurs ; c'est même plus que souhaité.

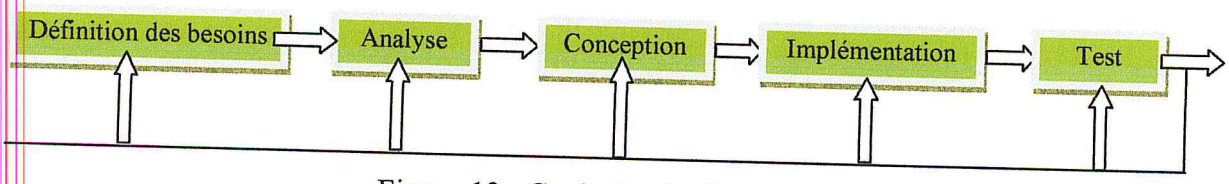


Figure 13 : Cycle de vie d'une IHR.

7.Conclusion

L'objectif de ce chapitre était de se familiariser avec les concepts liés aux système robotique cyber-physique (SRCP). Nous avons, à cet effet, essayé de donner un aperçu général sur le système cyber-physique (SCP). En outre, nous avons présenté le concept d'un robot dans un SRCP, les technologies utilisées dans le SRCP ainsi que des exemples des travaux de recherche récents.

Les systèmes robotique cyber-physique combinant le calcul, la communication, et de la dynamique physique. Ils sont difficiles à modéliser, à concevoir, et à analyser.

Et pour contrôler le système robotique cyber-physique et l'interaction de l'homme avec les robots sera faite grâce a une interface homme/robot (IHR).

Chapitre 02

Systèmes multi-agents

Chapitre 02

Systèmes multi-agents (SMA)

1. Introduction

Les systèmes multi-agents (SMA) ont émergé de la collaboration de plusieurs domaines que sont l'intelligence artificielle, les systèmes distribués et le génie logiciel. Cette discipline peut être assimilée à une mise en pratique du célèbre proverbe « l'union fait la force ». Elle se base, ainsi, sur la conception de systèmes composés d'un ensemble d'unités en interaction et qui coopèrent à la réalisation d'une fonction bien déterminée.

Dans ce chapitre, nous allons tenter de cerner le domaine des SMA. Nous l'abordons d'abord sous l'aspect "agent" en présentant les caractéristiques relatives à cette unité de base du SMA. Puis, nous passons à la dimension collective du système en présentant ces caractéristiques. Nous présenterons aussi la méthodes de conception AUML, quelques langages et plateformes de développement des SMA. Ainsi que les domaines qui utilisent ces systèmes.

2. 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.

2.1. Définition

Ferber définit un agent comme une entité physique ou virtuelle [33] :

- 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.

2.2. Propriétés

Des propriété les plus important Dun agent permet être résume aussi [29] :

- 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 ethumains) quand la situation l'exige afin de compléter ses tâches ou aider ces agents à accomplir les leurs.

2.3. Typologies des agents

Il existe principalement deux catégories d'agents : réactive et cognitive. Une troisième catégorie, appelée hybride, consiste à combiner les fonctionnalités des deux premières [29].

2.3.1. Agents cognitifs délibératifs

Les agents à capacités cognitives proviennent d'une métaphore du modèle humain [33]. Ces agents disposent d'une base de connaissances comprenant les diverses informations liées à leurs domaines d'expertise et à la gestion des interactions avec les autres agents et leur environnement.

De ce fait, ils sont capables de prendre des décisions à partir des informations dont ils disposent et de planifier leurs actions à l'avance.

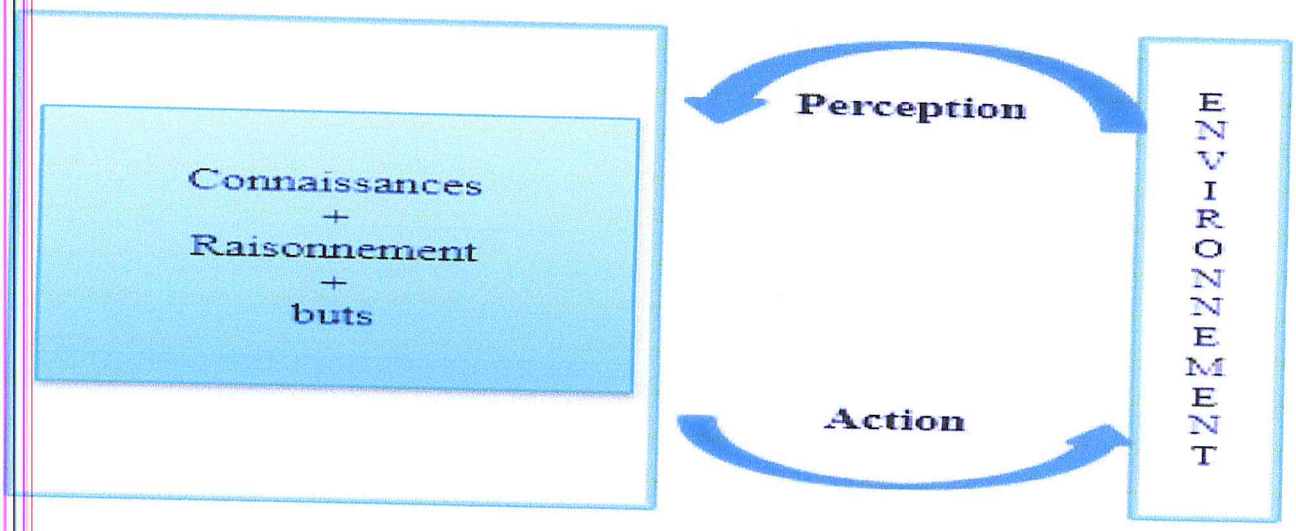


Figure 14 : Agent cognitif [33].

Par exemple, supposons qu'un robot veuille franchir une porte et que celle-ci soit fermée à clef. L'agent cognitif pourra construire un plan tel que [33] :

Plan ouvrir Porte

- aller jusqu'à l'endroit où se trouve la clef
- prendre la clef
- aller jusqu'à la porte
- ouvrir la porte avec la clef

En exécutant ce plan, l'agent cognitif ira directement au lieu où se trouve la clef pour la prendre, puis il se dirigera vers la porte pour l'ouvrir à l'aide de la clef.

2.3.2. Agents réactifs

Les agents réactifs ne sont pas « intelligents » pris individuellement [33]. Ils ne peuvent que réagir à des stimuli simples provenant de leur environnement. Leur comportement est alors simplement dicté par leur relation avec leur entourage sans qu'ils ne disposent d'une représentation des autres agents ou de leur environnement. Aussi, ils ne sont pas capables de tenir compte de leurs actions passées. Cependant, du fait, de leur nombre, ces agents réactifs peuvent résoudre des

problèmes qualifiés de complexes et leurs interactions permettent l'émergence d'une intelligence collective.

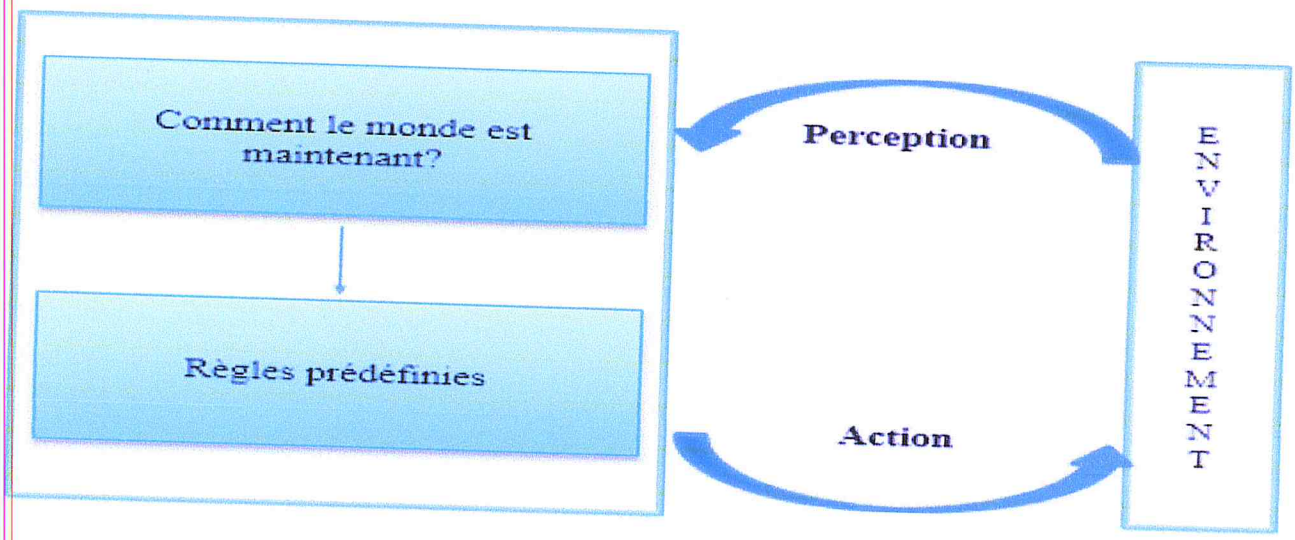


Figure 15 : Agent réactif [33].

Pour résoudre le problème de l'exemple précédent, l'agent réactif doit disposer du comportement suivant [30] :

R1 : si je suis devant la porte et que j'ai une clef, alors l'ouvrir

R2 : si je suis devant la porte et sans clef, alors essayer de l'ouvrir

R3 : si la porte ne s'ouvre pas et que je n'ai pas la clef, alors aller chercher la clef

R4 : si je cherche une clef et qu'il y a une clef devant moi, alors prendre la clef et aller vers la porte.

Ces quatre règles suffisent pour régler le comportement d'un robot réactif : si l'agent se trouve devant une porte fermée à clef, il essayera de chercher la clef, puis il reviendra pour ouvrir la porte.

Le tableau suivant résume les différences entre un agent réactif et un agent cognitif :

Agents cognitifs	**Agents réactifs**
Représentation explicite de l'environnement	Pas de représentation explicite
Peut tenir compte de son passé	Pas de mémoire locale
Agent complexes	Fonctionnement stimulus/action
Nombre réduit d'agents	Nombre élevé d'agents

Tableau 1 : Agents cognitifs vs réactifs.

2.3.3. Agents hybrides

Les agents hybrides sont des agents ayant des capacités cognitives et réactives. Ils conjuguent en effet la rapidité de réponse des agents réactifs ainsi que les capacités de raisonnement des agents cognitifs. Ce type d'agents, utilise une architecture multi-couches ; les couches de bas niveau utilisent des agents réactifs, et les couche de haut niveau utilisant un système cognitif plus complexe [29].

3. Systèmes multi-agents (SMA)

3.1. Définition

Un SMA est un ensemble d'agents autonomes en interaction, capables de s'organiser d'une manière dynamique et adaptative [34].

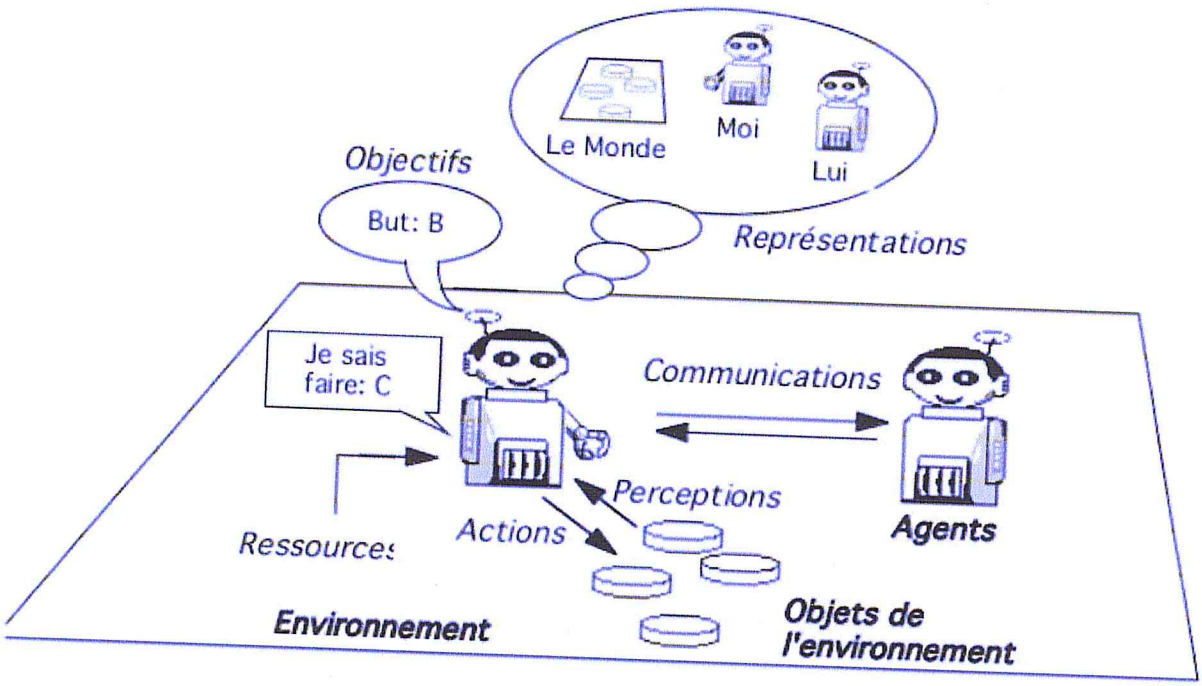


Figure 16 : Représentation d'un agent en interaction avec son environnement et les autres agents [33].

3.2. 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 [33].

Les différentes situations d'interactions entre les agents sont la communication, la coopération, l'organisation, la négociation et la coordination [33].

3.2.1. 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.

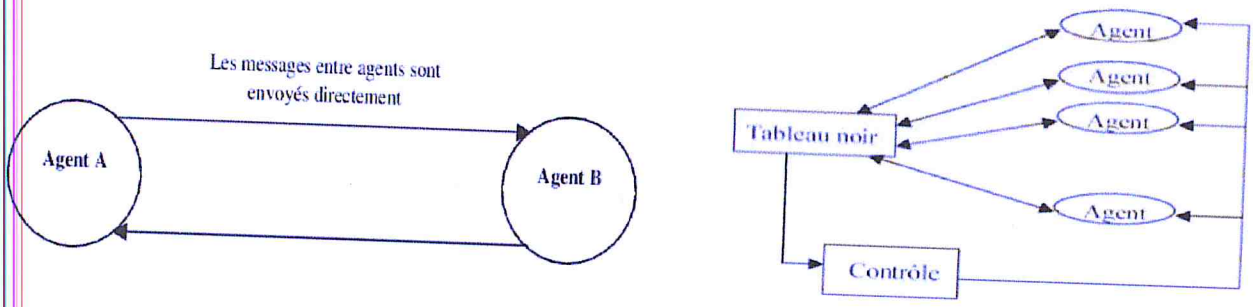


Figure 17 : Communication directe et indirecte [33].

3.2.2. Coopération

Demazeau et Müller [35] 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 [33] :

- 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.

3.2.3. 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 pré 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 [34].

3.2.4. Collaboration

Selon Ferber [33], 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".

3.2.5. 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 [35]. 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.

3.2.6. 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 [33].

3.3. Méthodologie de conception des SMA (AUML)

AUML [37] [38] est basé sur UML (Unified Modeling Language) qui est une méthode de génie logiciel utilisée pour le développement en langages orientés-objets. Elle est déjà largement utilisée par la communauté des concepteurs objets et son succès continue de croître.

Les agents ont des activités autonomes et des buts. C'est cette différence qui entraîne l'insuffisance d'UML pour modéliser les agents et les SMA. Aussi, AUML remplace la notion de méthodes par celle de services. Ses principales extensions sont :

- Diagramme d'agents qui est une reformulation du diagramme de classes.

- Diagramme de protocole qui permet une meilleure modélisation des interactions entre les agents.
- Diagramme de collaboration qui complémente le diagramme de protocole en proposant une autre lecture et vision des interactions entre agents.

3.4. 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) [39], JACK [40], GAMA [41], MaDKit [42] et JADE [43].

3.5. Domaines d'application des SMA

Les SMA répondent aux besoins des applications qui nécessitent la distribution des tâches sur un nombre d'agents, de coordonner et de faire interagir ces agents pour offrir une résolution efficace. Les domaines d'application des systèmes multi-agents sont particulièrement riches ; nous en citons en particulier : la recherche d'informations, le commerce électronique, l'aide au diagnostic médical et la robotique distribuée.

3.5.1. Recherche d'informations

La quantité d'informations disponible sur le web croît sans cesse. Les outils de recherches disponibles deviennent inadaptés car ils ne peuvent pas traiter une grande quantité de données ni de modifier la requête. Les agents de SMA peuvent visiter plusieurs sites web, coopérer et trouver les sites d'intérêts et revenir avec les meilleurs résultats; ils peuvent même modifier la requête en lançant un nouvel agent qui va informer ses collègues [47]. Citons comme exemple NetSA [48], une architecture de SMA pour la recherche d'informations dans des sources hétérogènes et réparties.

3.5.2. Commerce électronique

Avec le développement du commerce électronique, l'utilisation d'un agent qui fait des courses est de plus en plus envisageable. Un agent peut être utilisé pour fournir un accès personnalisé au magasinage en ligne. L'usager peut sélectionner un profil qui correspond à ses habitudes d'achat ou choisir des produits et lancer un agent qui va s'occuper de faire le magasinage [47].

3.5.3 Robotique distribuée

Les systèmes multi-agents ont très vite entretenu d'étroites relations avec le domaine de la robotique [49] ; que ce soit, entre autres, en robotique autonome mobile, robotique cellulaire ou productique distribuée. La conception orientée agents à toujours trouvé naturellement sa place pour chacun de ces domaines dérivant directement du paradigme de la robotique de groupes (constitué souvent de robots autonomes et mobiles). Notons, que les problèmes sont à quelques différences prés les mêmes dans le domaine de la robotique et dans celui des systèmes multi-agents [50] : autonomie, distribution, décentralisation, intelligence, mobilité, interactions, etc.

La robotique distribuée porte sur la réalisation non pas d'un seul robot, mais d'un ensemble de robots qui coopèrent pour accomplir une tâche en utilisant des agents concrets qui se déplacent dans un environnement réel.

Le domaine de la robotique distribuée recouvre en fait trois ensembles de techniques bien distinctes : la robotique cellulaire, la productique distribuée et la robotique mobile.

3.5.3.1 Robotique cellulaire

La robotique cellulaire s'intéresse à la constitution modulaire de robots. Dans ce cadre, un robot sera considéré comme un SMA et chacun de ses composants sera considéré comme un agent. La réalisation d'un mouvement sera alors le résultat de la coordination d'un ensemble d'agents.

Les recherches menées par l'équipe de Perram au Danemark [51] ont produit des modèles de déplacement de bras manipulateur dans lesquels chaque élément du bras est considéré comme un agent, les articulations décrivant des contraintes pour l'ensemble des mouvements acceptables. L'agent de tête cherche à satisfaire le but qui lui a été donné, par exemple souder deux pièces entre elles ou prendre un objet sur une table. S'il peut le faire lui-même, il effectue le déplacement, et le système s'arrête. Sinon, il entraîne l'agent qui le suit en lui donnant des buts pour que l'agent de tête se rapproche de son propre objectif. Le processus se répète récursivement, chaque agent cherchant à satisfaire les buts qu'on lui propose en transmettant ses desiderata à l'agent suivant. Les calculs s'expriment simplement et s'exécutent très rapidement. Ce type de modèle est suffisamment souple pour être effectivement utilisé dans des environnements industriels.

3.5.3.2 Productique distribuée

Supposons que l'on doive construire des produits manufacturés, A1,..,Ak à partir d'un ensemble de robots machines M1,..,Mn et de matières premières, P1,..,Pj , et que l'on dispose d'un ensemble de robots transporteurs T1,..,Tm pour transporter les produits intermédiaires d'une machine à une autre. Sachant que la matière première arrive à un bout de l'atelier de production, et que les produits manufacturés partent d'un autre bout, comment organiser cette unité de production, de manière qu'elle puisse réagir à toute modification de la demande. Même si ce problème ressemble beaucoup à un problème de recherche opérationnelle, il en diffère néanmoins par le fait que l'on ne demande pas seulement de prévoir la meilleure manière d'arranger les différentes unités de l'atelier, mais aussi de définir les programmes de chacune de ces unités de façon qu'elles puissent travailler ensemble et donc coordonnent leur travail en réagissant à des pannes éventuelles tout en pénalisant le moins possible l'ensemble de la production. Une réalisation particulièrement intéressante a été effectuée par Sohier et Bourdet de l'ENS-Cachan.

Ils ont développé un planificateur de tâches en temps réel pour un atelier flexible cellulaire fondé sur l'approche multi-agents. Chaque élément de l'atelier (tapis roulant, robot manutentionnaire, conteneurs, fraiseuses, etc.) est représenté sous forme d'un agent dont le comportement est tiré des principes de l'éco-résolution 4 Du fait de la nature hautement adaptative du système, il est possible de modifier les demandes ou les ressources en temps réel : le système se réorganise en temps réel pour que chacun des agents soit satisfait [52].

4. Conclusion

Les SMA possèdent maintenant une maturité suffisante pour pouvoir être utilisées dans le cadre d'applications réelles nécessitant le respect de contraintes opérationnelles fortes. Ainsi, l'application de ce système à des problématiques de contrôle/management de SRCP semble maintenant pertinente d'autant qu'elles permettent d'obtenir des propriétés d'adaptation, de robustesse, de résilience, etc.

Dans ce chapitre, nous nous sommes intéressés aux systèmes multi-agents, nous les avons d'abord définis. Ensuite, nous avons présenté la relation entre ces systèmes multi-agents et le domaine de la robotique.

Dans le prochain chapitre, nous réaliserons une interface de contrôle d'un système multi robots se déplaçant dans un environnement cyber-physique en s'appuyant sur le paradigme multi-agents.

Chapitre 03

La solution proposée

Chapitre 03

La solution proposée

1. Introduction

Contrôler un SRCP au sein duquel évolue un grand nombre d'entités autonomes 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 aussi une interface pour contrôler et manipuler tous ces éléments pour faciliter l'intégration entre l'utilisateur et les robots.

Suite aux notions de base présentées dans les chapitres précédents, ce chapitre s'intéresse à la conception de l'interface homme/robot (IHR) de contrôle d'une équipe de robots hétérogènes. La conception proposée utilise le modèle orienté objet et le langage de modélisation *AgentUML* (*Agent Unified Modeling Langage*). Au premier lieu, le chapitre présentera une description de l'environnement ainsi que tous les composants et les technologies du SRCP utilisés. En deuxième lieu, le chapitre donne une description détaillée sur les quatre scénarios considérés dans ce travail. En troisième lieu, le chapitre propose une modélisation du système développé. Enfin, en dernier lieu le chapitre se termine par la description de l'interaction entre les agents dans le système de contrôle.

2. Description de l'environnement

Après avoir donné la notion des SRCP dans les chapitres précédents ainsi que les technologies utilisées, notre travail consiste en le contrôle d'un système multi-robots évoluant dans un système cyber-physique. Des tags (RFID ou autres) sont fixés sur tous les robots, les opérateurs humains et les objets physiques (salles, murs, tables, ordinateurs, portes, bureaux,

etc.), alors que les lecteurs (RFID ou autres) sont déployés sur les robots. Plus de détails sont donnés par la figure suivante (Figure 18) :

- Nous avons utilisé des tags pour le stockage des informations, les lecteurs pour l'identification des entités (objets, robots, etc.) ; ces lecteurs sont exploités aussi par les robots pour l'identification des objets existant dans leur environnement.
- La communication est assurée par wifi.
- Chaque robot embarque un ordinateur afin d'assurer l'exécution efficace des tâches par les robots.
- Une interface homme/robot (IHR) pour contrôler les robots (via un PC, une tablette, etc.).
- Les entités physiques incluent les objets, les opérateurs humains et les robots :
    - Les objets : ce sont des entités passives, qui ne peuvent pas agir sur l'environnement ; par exemple : les tables, les chaises, etc.
    - Les robots manipulateurs : ce sont des robots fixés physiquement à leurs emplacements de travail et généralement mis en place pour réaliser des tâches précises ou répétitives.
    - Robots mobiles : sont des robots capables de se déplacer dans leurs environnements. Leurs rôle principal est le transport d'objets.
    - Robots manipulateurs mobiles : ce sont des robots mobiles surmontés d'un ou plusieurs manipulateurs.

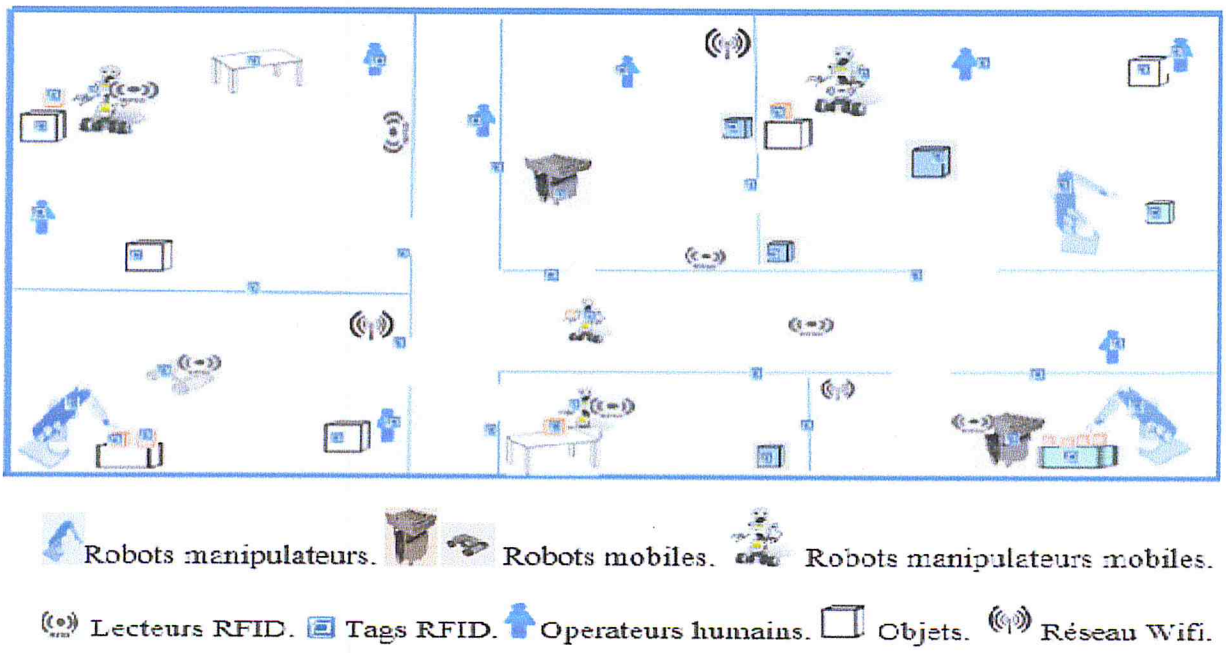


Figure 18 : Vue global du système robotique cyber-physique considéré

3. Modélisation de l'IHR

En général, le développement d'une application peut être abordé à travers différentes étapes qui vont permettre sa définition complète. Nous devons donc suivre un certain ordre dans la modélisation de la solution pour arriver à des résultats satisfaisant les besoins réels. Notre modélisation doit tenir compte des deux points suivants :

1. Satisfaire les qualités principales du l'IHR : l'IHR développée doit satisfaire certaines qualités pour assurer le bon fonctionnement du l'application et accroître son efficacité (voir le premier chapitre).
2. Communication et collaboration entre les agents.

Les différentes étapes suivies dans la modélisation de notre IHR peuvent être listées comme suit :

1. Spécification des besoins.
2. Conception.

3.1. Spécification des besoins

Avant de développer un système, il faut savoir précisément à quoi il devra servir. C'est-à-dire à quel besoin il devra répondre. Donc, la spécification des besoins permet de définir les besoins des utilisateurs du système.

Nous avons spécifié les besoins via le diagramme de cas d'utilisation.

Un diagramme de cas d'utilisation permet de représenter graphiquement les cas d'utilisation et d'identifier les fonctionnalités fournies par le système (cas d'utilisation), les utilisateurs qui interagissent avec le système (acteurs), et les interactions entre ces derniers. Les diagrammes suivants illustrent quelques cas d'utilisation du système :

- **Premier niveau :** Notre système permet de contrôler d'une équipe de robots hétérogène dans un SRCP.
- **Deuxième niveau :** Il existe une grande fonctionnalité qui est l'initialisation d'une tâche.

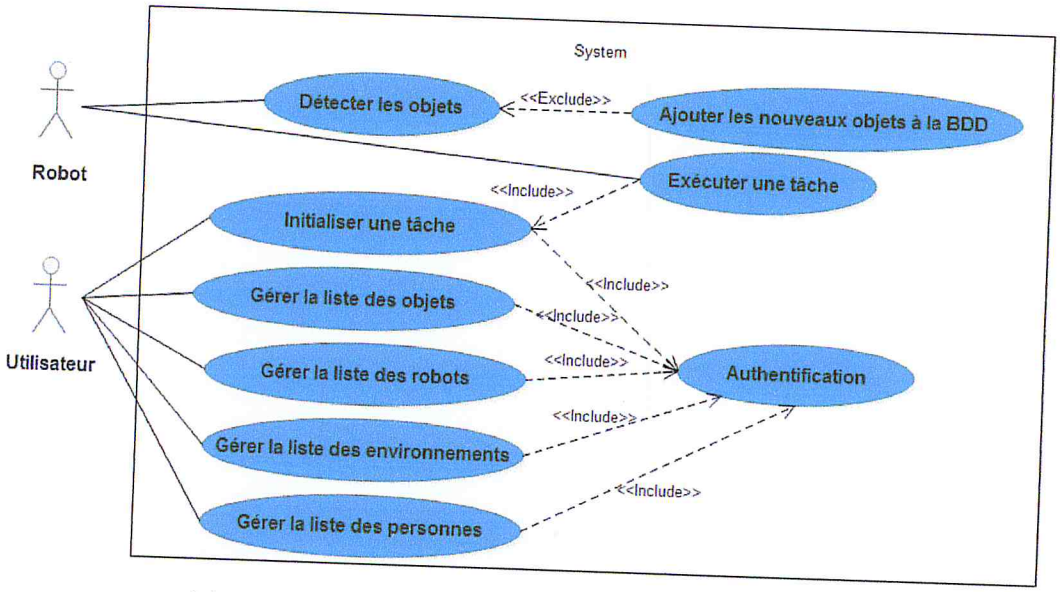


Figure 19 : Diagramme de cas d'utilisation global

Maintenant, nous allons détailler ce diagramme global en décrivant le cas d'utilisation général du système : Initialiser une tâche.

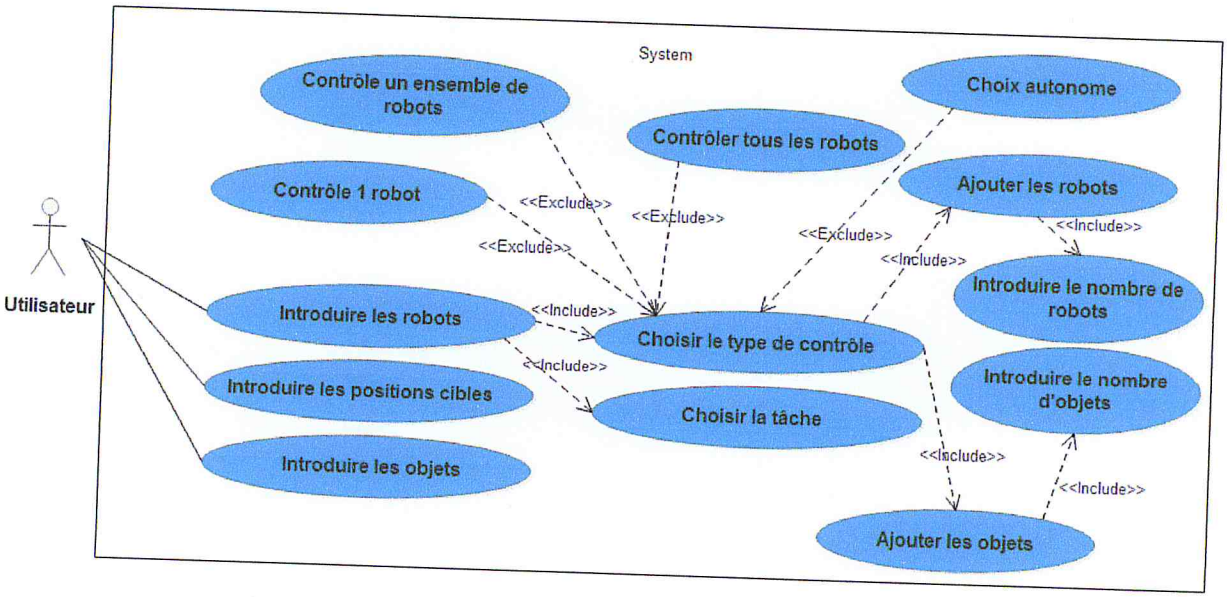


Figure 20 : Diagramme du cas d'utilisation « Initialiser une tâche »

3.2. Conception

La phase de conception permet de décrire le fonctionnement du système et de formaliser les étapes préparatoires de son développement, en vue d'en faciliter la réalisation et

d'assurer la conformité aux besoins de l'utilisateur. La conception permet la mise au point d'une solution.

La conception est exprimé dans notre cas par le diagramme de classes.

Le diagramme de classes est considéré comme l'élément le plus important dans la modélisation orienté objet ; donc, ce diagramme est considéré comme le point central en UML. Le diagramme de classes décrit la structure interne du système et de ces composants. Il décrit aussi les tâches définissant le comportement de ces derniers.

La figure 21 décrit les classes qui composent notre système ainsi que les relations, les attributs et les méthodes de ces derniers.

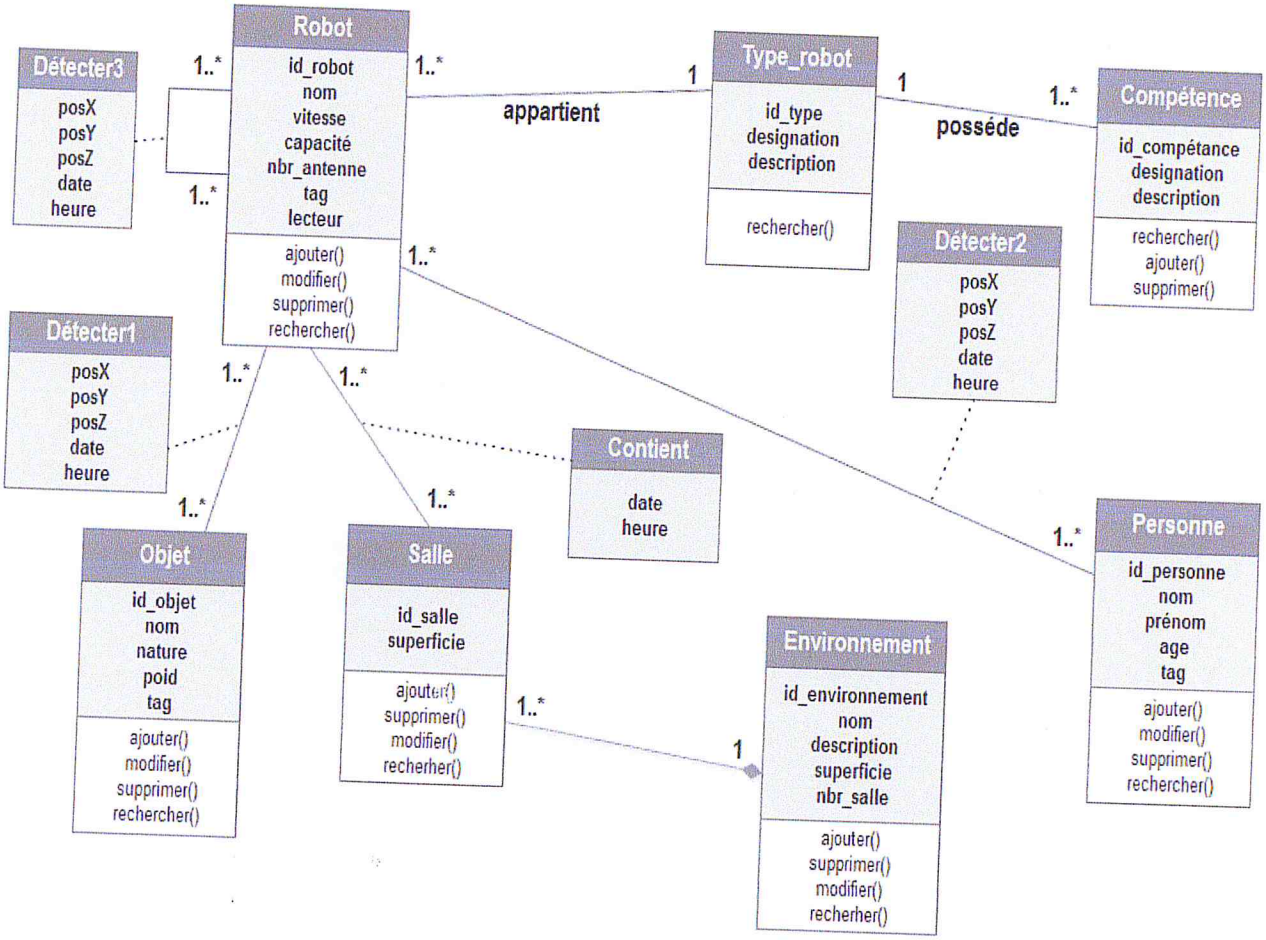


Figure 21 : Diagramme de classes

4. Système multi-agents

4.1. Diagramme d'agents

Notre système de contrôle se compose de deux différents types d'agents hybrides ayant des capacités cognitives et réactives. Ils conjuguent, en effet, la rapidité de réponse des agents réactifs ainsi que les capacités de raisonnement des agents cognitifs [29]. Chaque agent du système contient les informations suivantes : ID_agent, Nom d'agent, AdresseIP, Port. Le diagramme d'agents de la figure suivante (Figure 22) résume la relation entre les différents agents du système :

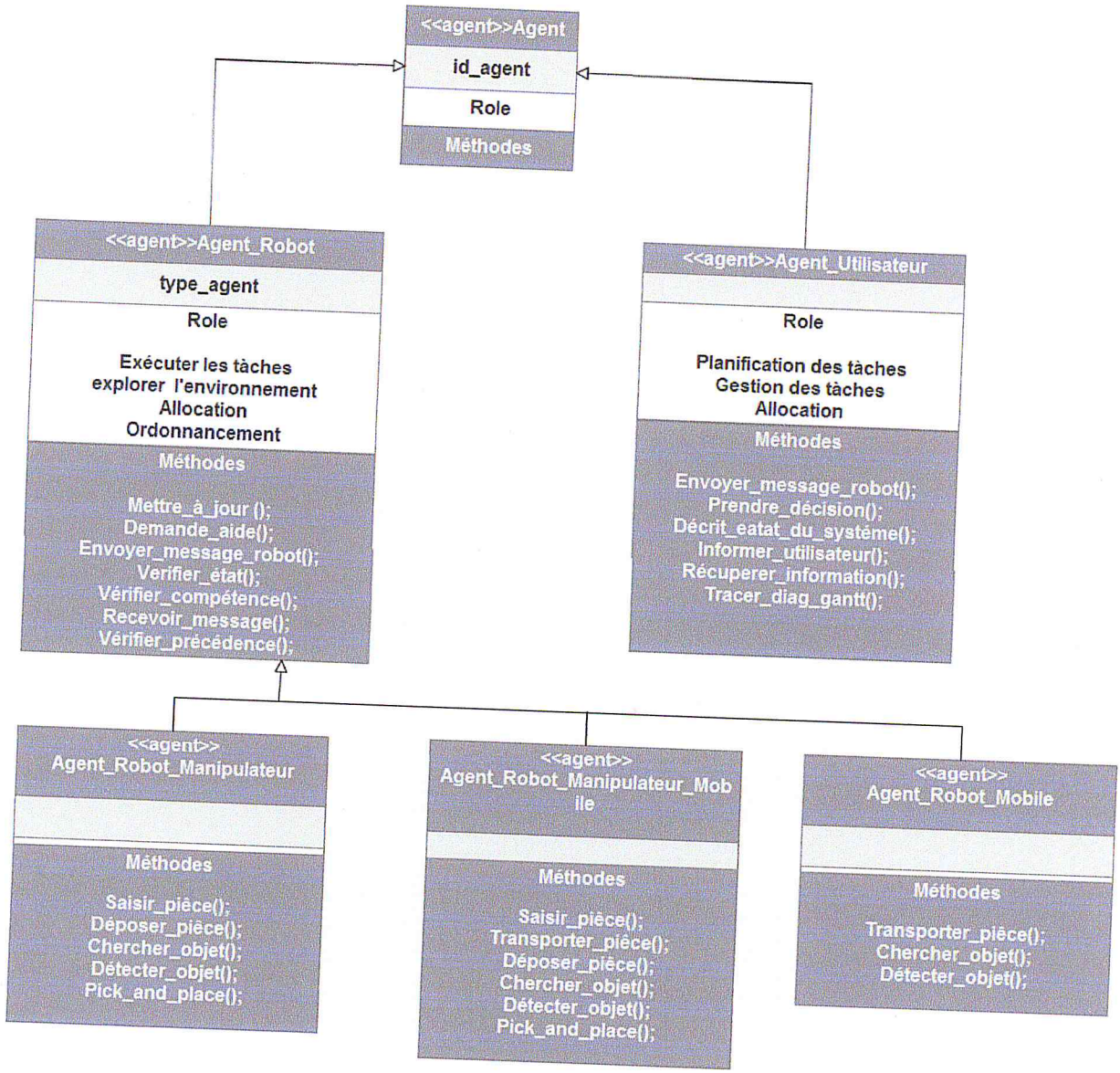


Figure 22 : Diagramme d'agents du système

4.2. Spécification des agents

4.2.1. Agent utilisateur

Le rôle de cet agent est de planifier les tâches et prendre les décisions ; il est implémenté sur un PC hôte. L'agent utilisateur regroupe les modules suivants présentés dans la figure 23 :

- **Interface utilisateur :** Ce module permet l'interaction avec l'utilisateur. Il permet de récupérer les requêtes des utilisateurs ; puis, de les envoyer au module *Gestion des tâches*. Ce module est, aussi, le responsable d'afficher les résultats obtenus aux utilisateurs.
- **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 *Communication*.
- **Planification :** Ce module interprète la tâche, la divise en ensemble de sous tâche (opérations) ; puis ils les envoie au module de *Communication*.
- **Communication :** Ce module est responsable de l'interaction de l'agent avec les autres agents du système pour envoyer et recevoir des informations.
- **Base de connaissances :** Cette base se décompose en deux parties : les connaissances individuelles et les connaissances sociales. Les connaissances individuelles reflètent la vue qu'à l'agent de lui-même (nom, adresse, objectifs individuels, protocoles de décision, etc.). Les connaissances sociales reflètent la représentation qu'a l'agent de l'environnement dans lequel il évolue. Ces connaissances particulières portent sur les agents qu'il peut contacter (Agents ID, Adresse IP, Ports d'envoi, Ports de réception, etc.).

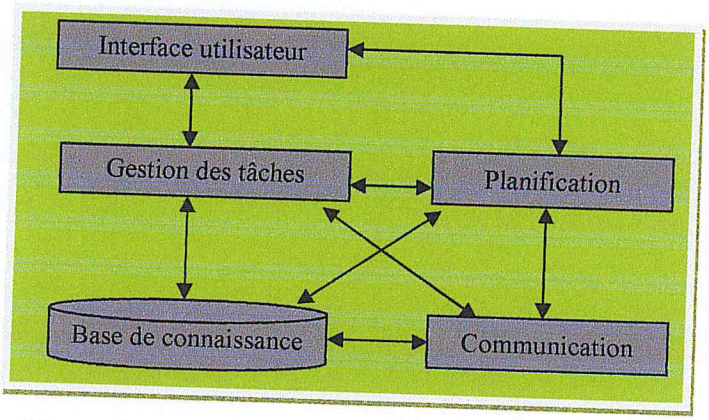


Figure 23 : Structure interne de l'agent Utilisateur

4.2.2. Agent Robot

L'agent Robot est un agent hybride ; son objectif est d'exécuter les consignes envoyées par l'agent Utilisateur. Il comprend les modules suivants (figure 24) :

- **Mise à jour :** Ce module se charge de la mise à jour des positions des objets dans la base de données ainsi que de l'insertion des nouveaux objets dans cette base.
- **Base de connaissances :** Chaque agent Robot est doté d'une base de connaissances qui contient des informations sur lui-même et sur les autres agents du système.
- **Perception :** Ce module permet à l'agent robot, de percevoir l'état de son environnement en collectant toutes les informations en provenance des lecteurs RFID équipant le robot.
- **Communication :** Ce module se charge de la communication entre cet agent et les autres agents de système.
- **Allocation :** Ce module permet d'allouer les opérations aux robots les plus adéquates selon les compétences et les engagements de ces agents (ce module utilise la règle de priorité) .
- **Ordonnancement :** Ce module permet le séquençage des opérations allouées à chaque robot (ce module utilise l'algorithme génétique afin de minimiser le temps d'exécution total).
- **Traitement** : Ce module donne l'ordre aux robots pour commencer l'exécution ou attendre en respectant les règles de précédence (dans le cas du contrôle manuel)

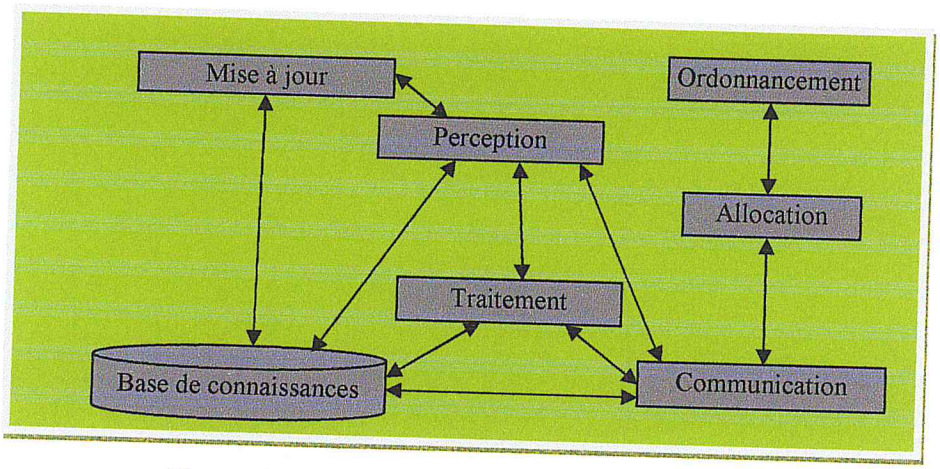


Figure 24 : Structure interne de l'agent Robot

4.3. Interaction entre les agents

Après avoir détaillé chaque agent du système, nous allons décrire dans cette section les différentes interactions du système sous forme de diagrammes.

Notre système se compose d'une interface qui permet aux utilisateurs, après authentification, de contrôler une équipe de robots pour réaliser des tâches.

4.3.1. Diagramme d'interaction

C'est un diagramme dynamique, qui décrit le comportement intérieur de l'agent [30]. Les interactions entre les agents peuvent être représentées dans le standard *Agent UML* en utilisant des diagrammes de séquence [30]. Le diagramme de la figure suivante (Figure 25) résume l'interaction entre les agents, ou l'utilisateur veut contrôler un seul robot :

- 1-Initialiser la tâche.
- 2-REQUEST(nom_agent, opération, id_objet, posX, posY, posZ).
- 3-Vérifier l'état du robot.
- 4-PROPOSE : envoyer une proposition contenant la durée ainsi que la réponse à l'agent utilisateur.
- 5-ACCEPT_PROPOSAL.
- 6-Chercher l'objet : le robot commence à chercher l'objet.
- 7, 8-CFP : le robot fait appel à propositions à d'autres robots pour demander de l'aide.
- 9, 10-PROPOSE : chaque robot, selon son état, va envoyer une proposition.
- 11-Sélectionner la meilleure proposition : le robot sélectionne la (les) meilleure(s) proposition.

On suppose que :

- 12-ACCEPT_PROPOSE.
- 13-REJECT_PROPOSE.
- 14-Si l'objet est détecté par le robot non sélectionnée par l'utilisateur, il en informera le robot concerné par un message INFORM qui contient la position de l'objet.
- 15-Si l'objet est détecté par le robot sélectionné par l'utilisateur, il en informera les autres robots pour qu'ils abandonnent la tâche.
- 16-Exécution de la tâche.
- 17-INFORM : informer l'agent l'utilisateur que l'exécution de la tâche a échoué ou réussi.
- 18-REJECT_PROPOSAL.

- 19-Afficher les résultats finaux.

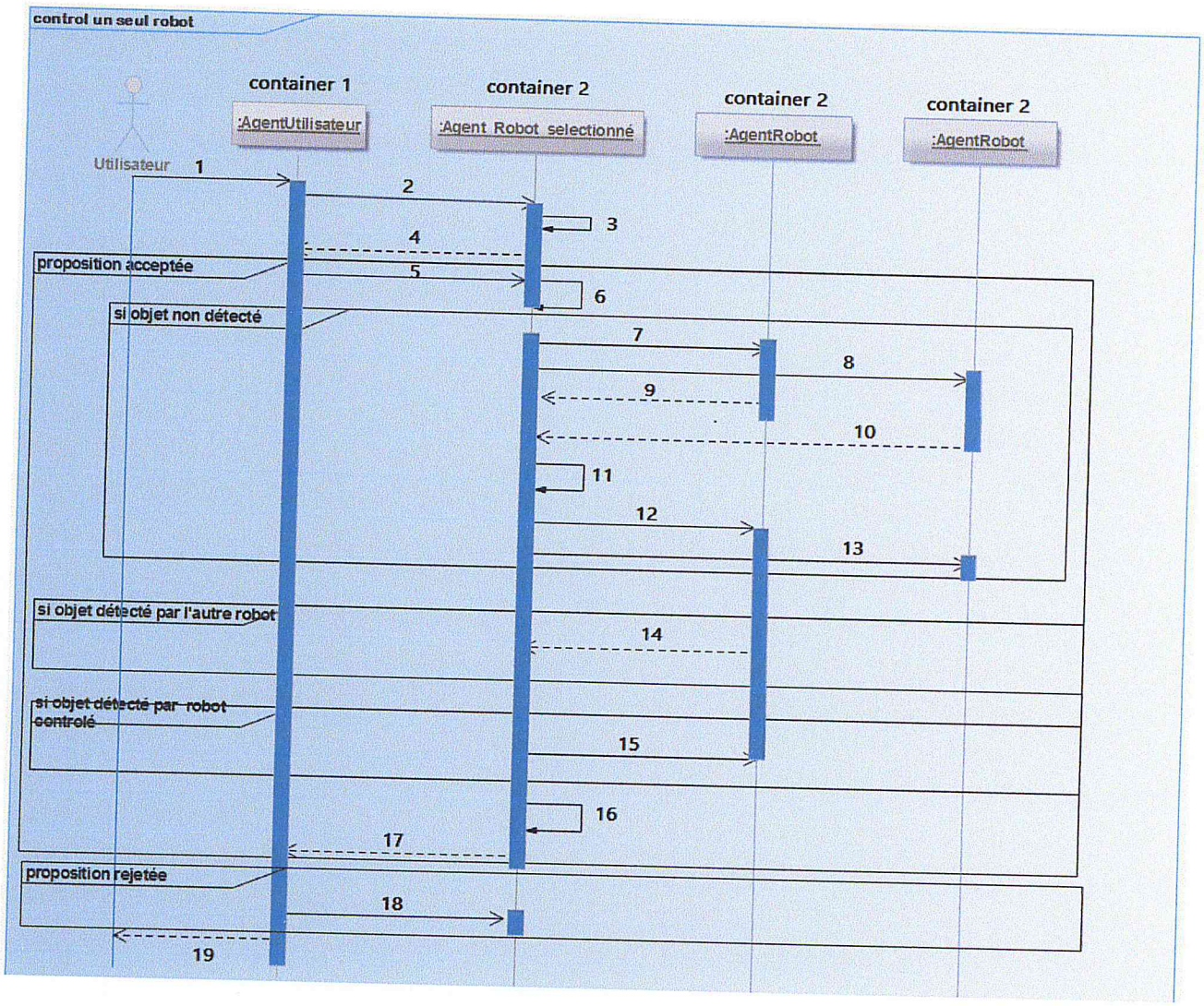


Figure 25 : Diagramme de séquence d'interaction entre les agents pour contrôler un seul robot

Le diagramme de la figure suivante (Figure 26) résume l'interaction entre les agents dans le deuxième cas (contrôler un sous ensemble de robots), où la demande de l'aide aux autres robots se fait de la même manière que dans le premier cas.

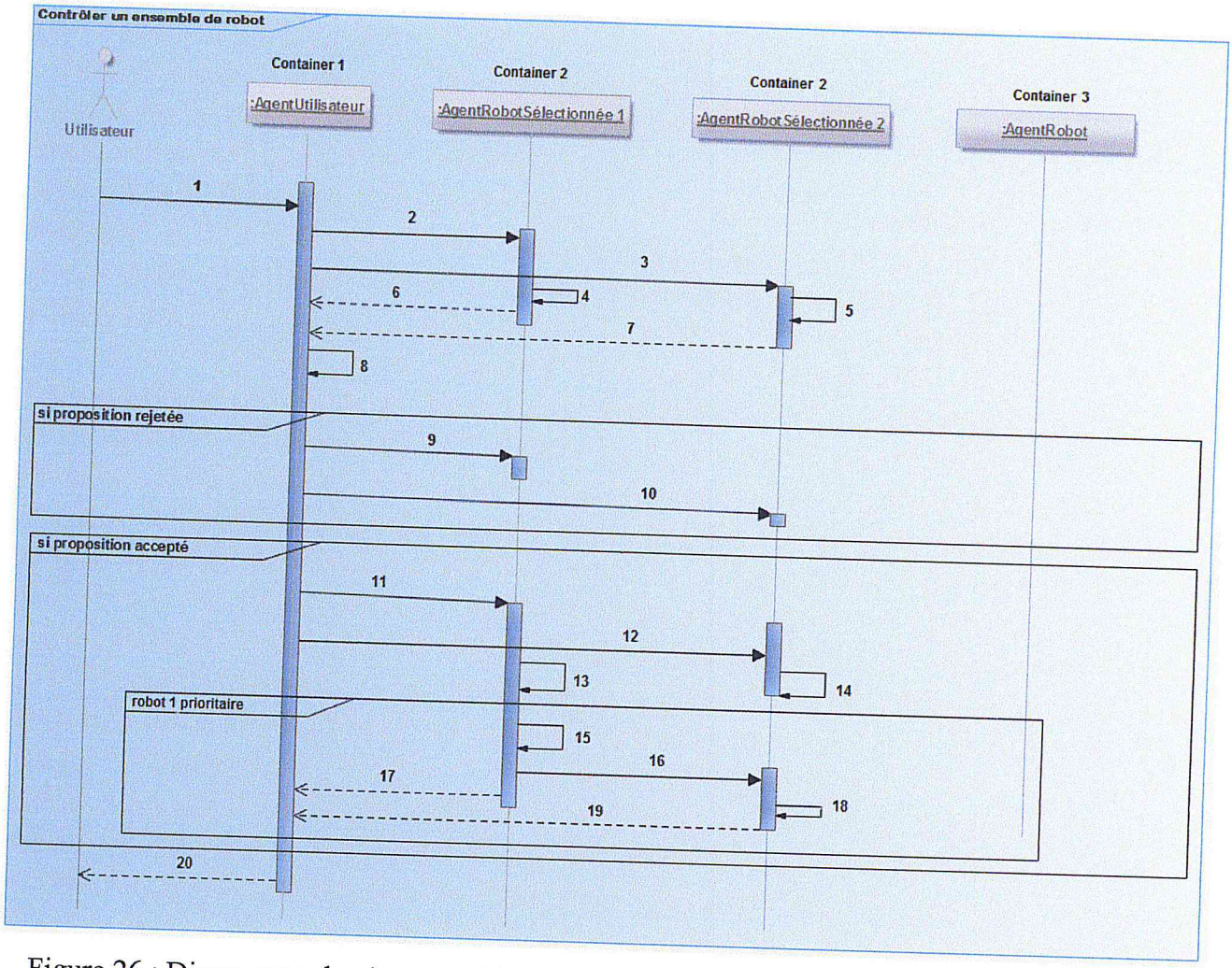


Figure 26 : Diagramme de séquence d'interaction entre les agents pour contrôler un sous-ensemble de robots

N°	Description
1	Initialiser une tâche
2, 3	REQUEST(nom_agent, opération, précédence, id_tag_RFID, posX, posY, posZ)
4, 5	Vérification l'état du robot
6, 7	INFORM : informer l'utilisateur sur l'état du robot
8	Vérification les propositions
9, 10	REJECT_PROPOSE
11, 12	ACCEPT_PROPOSE
13, 14	Vérification les règles de précédences
15	Exécuter la tache par le robot le plus prioritaire
16	INFORM : informer les autres robots que la tâche a été exécutée
17	INFORM : informer l'agent utilisateur que la tâche a été exécuté

18	Exécuter la tache par le robot suivant
19	INFORM : informer l'agent utilisateur que la tâche a été exécuté
20	Afficher les résultats à l'utilisateur

Tableau 2 :Description d'interaction entre les agents (contrôler un sous-ensemble de robots)

Le diagramme de la figure 27 résume l'interaction entre les agents dans le troisième cas (contrôler tous les robots).

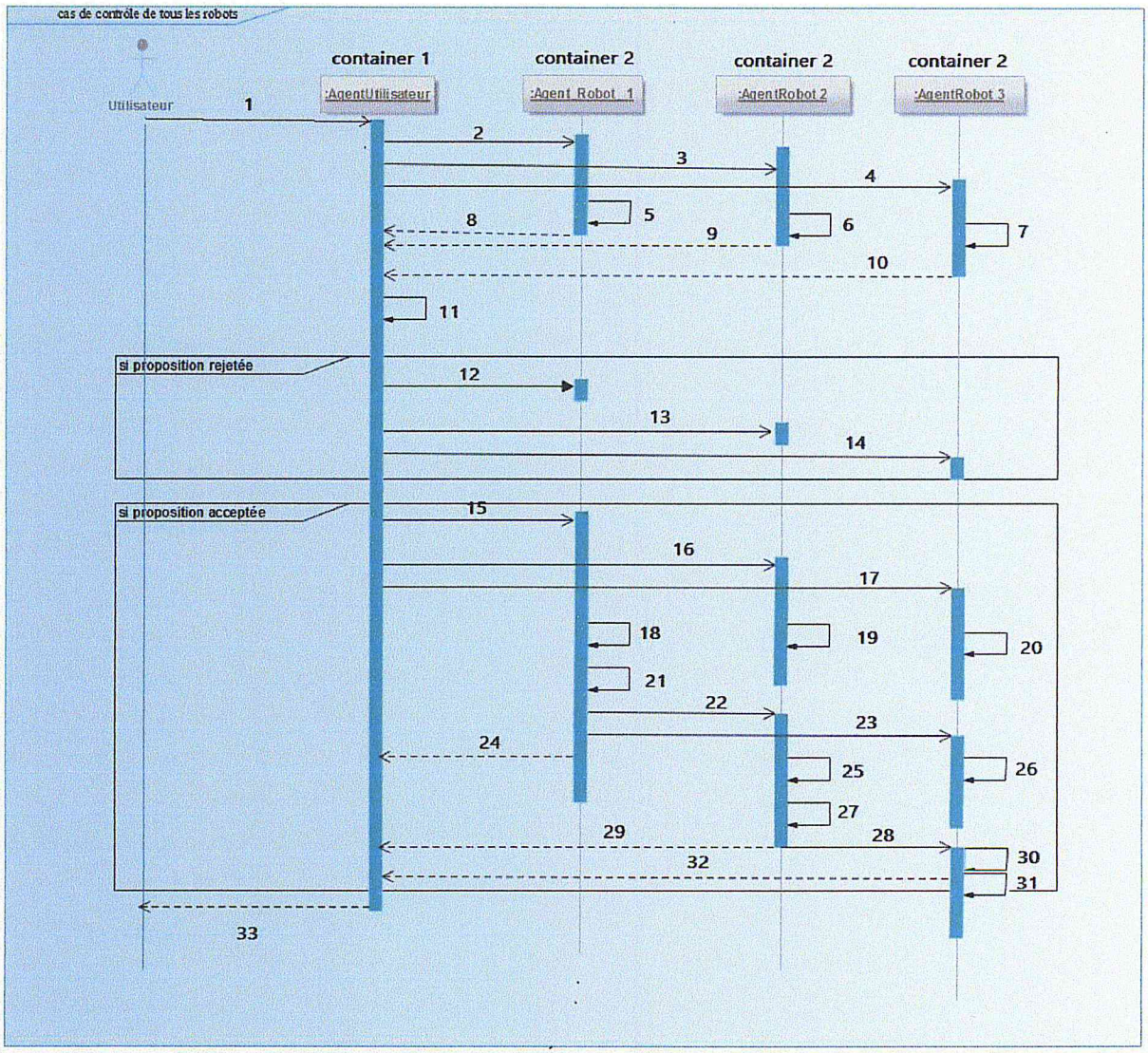


Figure 27 : Diagramme de séquence d'interaction entre les agents dans le cas de contrôle de tous les robots

N°	Description
1	Initialiser une tâche
2, 3, 4	REQUEST(nom_agent, opération, précédence, id_tag_RFID, posX, posY,

	posZ)
5, 6, 7	Vérifier l'état du robot
8, 9, 10	PROPOSE : proposer à l'agent utilisateur selon l'état du robot
11	Vérifier les propositions
12, 13, 14	REJECT_PROPOSE
15, 16, 17	ACCEPT_PROPOSE
18, 19, 20, 25, 26, 30	Vérifier les règles de précédence
21	Exécuter la tache par les robots les plus prioritaires
22, 23, 28	INFORM : informer les autres robots que la tâche a été exécutée
24, 29, 32	INFORM : informer l'agent utilisateur que la tâche a été exécuté
27, 31	Exécuter la tâche par le robot suivant
33	Afficher les résultats à l'utilisateur

Tableau 3 : Description d'interaction entre les agents (contrôler tous les robots)

Le diagramme de la figure 28 résume l'interaction entre les agents dans le quatrième cas (choix autonome).

Scénario :

- 1-Initialiser une tâche.
- 2-Décomposer la tâche.
- 3, 4, 5-Envoyer les opérations à chaque robot.
- 6, 7, 8, 9, 10, 11-Envoyer une proposition.
- 12, 13, 14-Informer l'agent utilisateur.
- 15-Afficher les résultats à l'utilisateur.

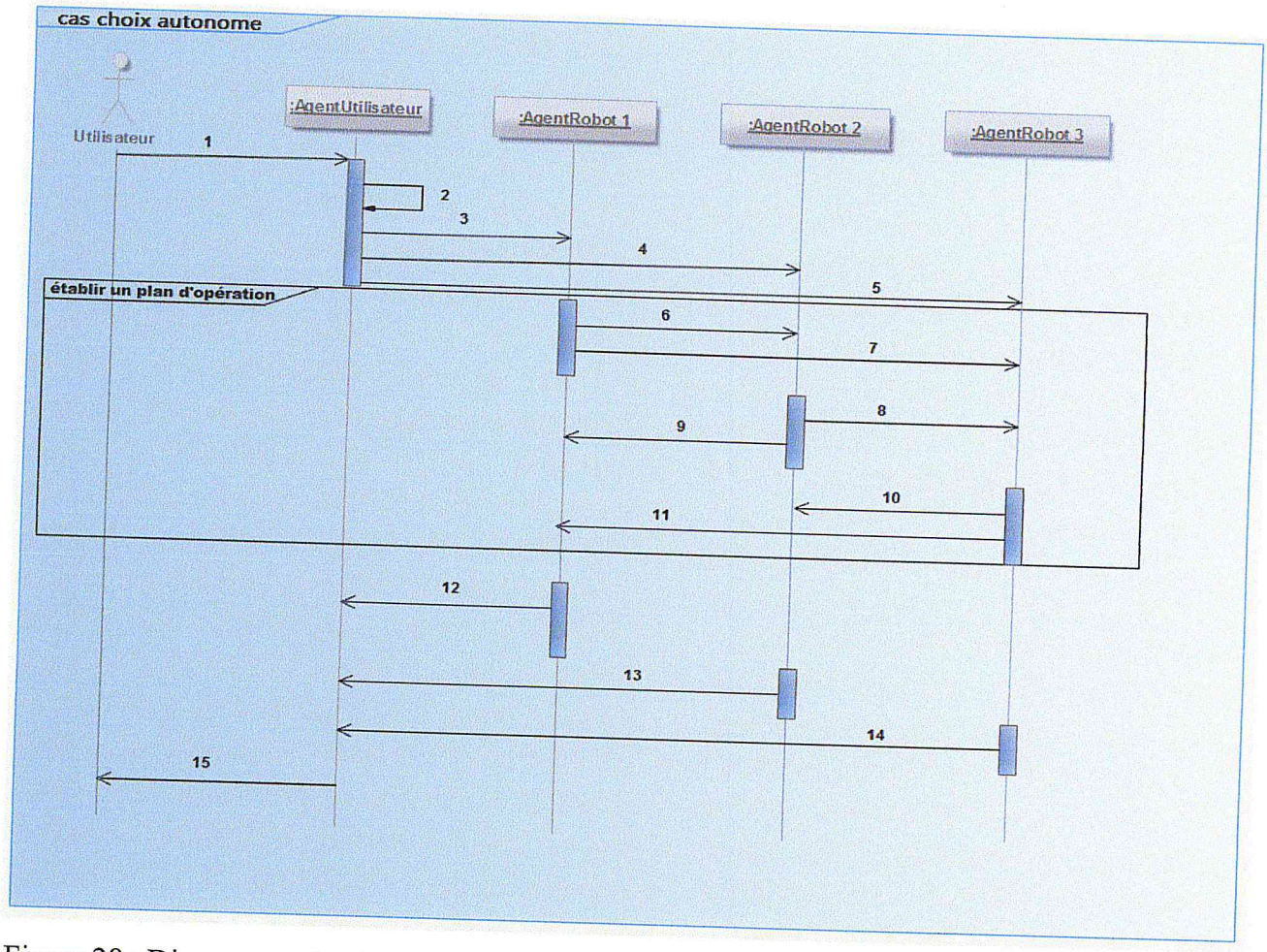


Figure 28 : Diagramme de séquence d'interaction entre les agents dans le cas choix autonome

5. Solution proposée

Notre interface proposée se base principalement sur la stratégie de l'utilisation et l'adaptation des systèmes multi-agents (SMA) dans un contexte robotique cyber-physique. Nous avons considéré les quatre scénarios suivants :

5.1. Premier scénario : Contrôler un seul robot

Il consiste à programmer l'agent Robot choisi par l'utilisateur de sorte qu'il effectue la tâche demandée. Pour cela, le robot navigue dans son environnement ; chaque fois qu'il détecte un objet, il l'ajoute à la base de données s'il n'y existe pas. Il envoie, en parallèle, une requête aux autres agents en sa proximité pour leur demander de collaborer avec lui afin de gagner du temps et assurer une bonne efficacité. Cependant, la tâche doit être exécutée par le robot sélectionné ; les autres robots ne font que chercher l'objet en question.

L'agent utilisateur récupère les données qui ont été saisies par l'utilisateur (nom du robot, opération, précédence malgré qu'elle ne soit pas nécessaire dans ce premier cas par ce qu'elle est toujours égale à 1, objet, position cible).

5.1.1.Algorithmes

Lorsque les données sont récupérées par l'agent utilisateur, ce dernier va transmettre une requête au robot sélectionné par l'utilisateur.

Algorithme 1 : Envoyer la requête au robot{

REQUEST(agent_robot, opération, précédence, id_objet, posX, posY, posZ);

}

Lorsque l'agent robot reçoit la requête envoyée par l'utilisateur ou par un autre robot grâce aux messages ACL, la requête doit contenir la tâche, l'identificateur d'objet et le nom de l'agent.

Le robot va vérifier premièrement s'il a les compétences requises pour réaliser cette tâche. Si le résultat est négatif il refusera la tâche ; sinon, il va vérifier s'il est disponible ou pas. Plus de détails sont donnés dans l'algorithme 2.

Algorithme 2 : Etat_de_robot{

```
if (pouvoir_robot==false){//le robot n'a pas les compétences pour effectuer cette tâche
        réponse = REJECT ; //tâche refusée
        date_début=maximum ;
}
else{//le robot a les compétences pour effectuer cette tâche
        if (etat_de_robot==occupé){//robot occupé
                réponse=ACCEPT ; //tâche acceptée
                date_début=temps_restant ;//le temps pour qu'il finira la tâche en cours
        }
        else{//robot disponible
                état_de_robot=occupée;//état de robot passe de disponible à occupé
                réponse= ACCEPT ; //accepter la tâche
                date_début=0 ;
        }
```

```
        }
}
```

Quand un agent robot reçoit un message de la part de l'utilisateur ou d'un autre robot, cet agent va réagir selon le type de message reçu :

1. PROPOSE : le robot va recevoir des propositions de la part des robots existants dans le SRCP ; ensuite, il va sélectionner la meilleure proposition, et rejeter les autres.
2. REQUEST et CFP : le robot va vérifier son état (algorithme 2) puis envoyer une proposition à l'agent utilisateur ou robot. Ce message contiendra sa réponse plus sa date début d'exécution.
3. ACCEPT_PROPOSAL : dans ce cas le robot va exécuter la tâche en respectant l'émetteur et les règles de précédences. Puis, envoyer un message à l'agent utilisateur pour l'informer si la tâche a été exécutée avec succès ou pas.
4. REJECT_PROPOSAL : le robot ne va rien faire. Sa proposition a été rejetée.

L'agent robot traite les types des messages selon l'algorithme 3 suivant :

Algorithme 3 : Réception d'un message{

```
  switch (type_msg){
    case "REQUEST" {
            Etat_de_robot();
            PROPOSE (nom_agent_robot, date_début, réponse);
    }
    case "CFP"{
            Etat_de_robot();
            PROPOSE (nom_agent_robot, date_début, réponse);
    }
    case "PROPOSE" {
            choisir_meilleure_propose();
     //date_début doit être supérieur ou égal à zéro
            ACCEPT_PROPOSE(nom_agent, date_début_minimum);
            REJECT_PROPOSE(Autres_robots);
    }
    case "ACCEPT_PROPOSAL" {
```

```
            if (!tache_en_cour_terminée){
                insérer_la_tache();//insérer la tâche dans la file d'attente
            }
            else{
                Exécuter_tache(sender, contrôle, tache, précédence, objet, posx, posY,
                posZ, acl);
                //contrôle est le type de contrôle (un seul robot, ensemble de robots)
            }
        }
        case "REJECT_PROPOSAL" {
            block();//ne rien faire
        }
    }
}
```

Quand l'agent robot sélectionné reçoit ACCEPT_PROPOSAL, il va exécuter l'algorithme 4 pour choisir la tâche.

Algorithme 4 : Choisir tâche{

```
    switch (tâche){
        case"transporter" :{
            transporter_objet (id_objet , posX, posY, posZ);
            break;
        }
        case"pick" :{
            pick_and_place (id_objet , posX, posY, posZ);
            break;
        }

        Case"déplacer" :{
            déplacer_objet (id_objet, posX, posY, posZ);
            break;
        }
        case"détecter": {
```

```
                Détecter_les_objet ();
                break;
            }
        //Détecter les objets permet d'insérer dans la base de données les objets non existants.
            case"chercher" : {
                Chercher_les_objet (id_objet, detecté);
                break;
            }
        //détecter est un booléen ; s'il est faux, le robot cherche encore ; sinon il s'arrête.
            case"saisir":{
                saisir_objet(id_objet, posX, posY, posZ);
                break;
            }
        }
}
```

Lorsque le robot commence à chercher l'objet et dans le but de gagner du temps, il envoie des CFP pour demander de l'aide aux autres agents robots qui se trouvent à proximité s'ils sont disponibles ou si la tâche est plus prioritaire. Dans ce cas, l'algorithme 5 suivant sera exécuté :

Algorithme 5 : Demande de l'aide{

```
    while(objet_detecté=false){//objet non détecté
        Chercher_les_objets(id_objet , false);
    }
    if (objet_detecté==true){//
        INFORM(Autre_agents, id_objet, true);
    }
}
```

À la réception d'un message de la part d'un robot, nous en distinguons deux types :

1. Une information (ACLMessage.INFORM) : pour informer l'agent utilisateur que la tâche a été exécutée ou a échoué ; puis, transmettre l'information au module *Interface utilisateur* pour afficher les résultats obtenus.
2. Une proposition (ACLMessage.PROPOSE) : l'agent utilisateur récupère la réponse et la date_début proposée par le robot. Puis, il va vérifier si la proposition lui convient ; ensuite il va répondre par un ACCEPT_PROPOSAL ou REJECT_PROPOSAL.

5.1.2 Exemple

Le robot sélectionné (un manipulateur mobile) doit transporter un objet d'un point A à un point B donné par les coordonnées (x, y, z).Ainsi, le robot commence à chercher et envoyer des messages aux autres agents pour chercher aussi l'objet en question ; le premier qui le détecte (si ce n'est pas le robot sélectionnée) va envoyer un message au robot concerné par cette tâche contenant les cordonnées (Xo,Yo,Zo) de l'objet. Après, le robot sélectionné va envoyer un message à tous les autres robots leur informant que l'objet a été trouvé.

5.2. Deuxième scénario :Contrôler un sous-ensemble de robots hétérogène

Un ensemble de robots signifie que l'utilisateur a besoin de contrôler deux robots ou plus afin d'effectuer une tâche. Ces robots peuvent demander de l'aide à d'autres robots du système afin de minimiser le temps d'exécution de cette tâche. Ces robots détectent les objets en lisant les informations stockées dans les tags grâce aux lecteurs installés sur les robots. De plus, ils communiquent entre eux grâce au langage de communication ACL. Néanmoins, la tâche doit être réalisée par les robots sélectionnés.

5.2.1. Algorithmes

L'utilisateur commence par le choix de la tâche à réaliser ainsi que la sélection des robots. Par la suite, il va affecter à chaque robot une opération précise selon ses compétences. À la fin, une requête sera envoyée à tous les robots sélectionnés (Algorithme 6).

Algorithme 6 : Envoyer la requête aux robots{

```
REQUEST(liste_agents, tâche, précédence, id_objet, posX, posY, posZ);
//la liste des agents contient les robots sélectionnés pour accomplir cette tâche
}
```

Quand tous les robots reçoivent la requête et selon leurs états, chaque robot va envoyer une proposition, sauf que cette fois la réponse sera envoyée à l'agent utilisateur qui va lui-même sélectionner les robots qui vont exécuter cette tâche (selon la disponibilité de tous les robots). Dans le cas ou tous les robots sont prêts et disponibles pour l'exécution de la tâche, l'agent utilisateur accepte la tâche ; sinon, il rejette les propositions de tous les robots.

Si les robots reçoivent un ACCEPT_PROPOSAL, chaque robot va vérifier les prédécesseurs de son opération pour qu'il puisse commencer l'exécution ou attendre son tour. Lorsque le robot termine l'exécution de l'opération, il va en informer les autres. L'algorithme 7 sera exécuté comme suit :

Algorithme 7 : Arrivée d'un Message{

```
case "INFORM"{
        Exécuter(sender, opération, précédence, id_objet, posX, posY, posZ,
aclMessage);
}
case "ACCEPT_PROPOSE"{
        Exécuter(sender, opération, précédence, id_objet, posX, posY, posZ,
aclMessage);
}
}
```

Lorsque l'agent utilisateur reçoit toutes les réponses émanant des autres agents, il vérifie si tous les agents ont accepté de réaliser ces opérations. Dans ce cas, il va en informer chaque robot afin de commencer son opération. Dans le cas contraire(un refus), il va envoyer un message aux robots les informant que la tâche a été annulée (Voir l'algorithme 8 pour plus de détails).

Algorithme 8 : Réception des messages{

```
recevoir (ACLMessage);//il contient la réponse, date_début ainsi que l'émetteur
if (réponse.equals("ACCEPT")&&(durée<durée_agent)){
        nbr_accept=nbr_accept++;
}
```

```
        //durée_agent est une durée calculée par l'agent utilisateur
        if (nbr_accept==nbr_agent_selectioné) {
                ACCEPT_PROPOSAL(liste_agent_robot);
        }
        else{
                REJECT_PROPOSAL(liste_agent_robot);
        }
}
```

Lorsque un robot commence à chercher un objet et afin de minimiser le temps d'exécution de la tâche, il envoie des messages demandant de l'aide aux autres robots du système s'ils sont disponibles (ou dans le cas où la tâche est plus prioritaire). Dans ce cas, l'algorithme 04 sera ré-exécuté.

5.2.2. Exemple

L'utilisateur veut déplacer un objet quelconque d'un point A vers un point B(Xb, Yb, Zb). De plus aucun des robots manipulateurs mobiles n'est disponible ou bien les robots manipulateurs mobiles disponibles ne peuvent pas supporter le poids de l'objet sélectionné. Dans ce cas, nous pouvons faire appel à un robot manipulateur afin de soulever l'objet et le déposer sur un robot mobile. Ce dernier va déplacer cet objet à la position cible (Xb, Yb, Zb).

5.3. Troisième scénario :Contrôler tous les robots du système

Le troisième scénario consiste à contrôler tous les agents robots du système pour qu'ils collaborent entre eux à réaliser une tâche (très complexe) qui nécessite les compétences de tous les robots (tous les robots du système sont concernés par cette tâche). Dans ce cas, ils communiquent entre eux pour assurer l'efficacité et accomplir la tâche avec succès en un temps précis. Il faut que tous les robots du système soient disponibles pour cette tâche.

5.3.1. Algorithmes

Lorsque l'utilisateur veut réaliser une tâche complexe qui ne peut pas être réalisée par un seul ou un sous ensemble de robots, il fait appel à tous les robots existants dans le système.

Considérons un environnement qui contient un système robotique un ensemble de *n* tâche T={T1,T2,T3,...,Tn} doivent être exécutées par *m* robots R={R1, R2, ..., Rm} avec des capacités et compétences différents (type de robot, vitesse, capacité de saisie). Chaque tâche est composé de plusieurs opérations non-préemptives O={O11, O12, O13, ..., Oij}, chaque opération Oij peut être traitée par un robot disponible (Oij signifie l'opération j de la tâche i).

L'objectif principal est de reparti l'ensemble des tâche sur les robots en minimisant le temps d'exécution total. Les hypothèses retenues dans ce problème sont comme suite :

- Chaque robot peut exécuter une seule opération à la fois.
- Les opérations appartenant à différentes tâches peuvent être traitées en parallèle.
- Les opérations sont non-préemptives et chaque opération doit être traitée sur un et un seul robot parmi l'ensemble de robots R.
- Chaque opération pourrait être traitée plus d'une fois sur le même robot.
- L'ordre des opérations pour chaque tâche est prédéfini et ne peut être modifié.
- Il n'y a pas de contraintes de priorité entre les opérations de tâches différentes.

Le problème peut être divisé en deux sous-problèmes étroitement liés. Le premier consiste à affecter chaque opération à un robot. Le second consiste à séquencer, sur chaque robot, les opérations assignées tout en répondant aux contraintes de précédence du problème et minimiser le temps global d'exécution (makespan).

5.4.1. Allocation

C'est la première étape nécessaire. Lorsque l'utilisateur choisit la tâche à exécuter, l'objectif est de répartir cette tâche sur les différents robots existants dans le système selon leurs compétences d'une manière intelligente et autonome (sans intervention humaine). Donc, l'agent utilisateur va récupérer la tâche il la décompose en un ensemble d'opérations et, enfin, transmettre ces opérations accompagnées de leur durée de chaque opération aux robots disponibles selon leurs compétences et capacités.

Chaque robot reçoit un ensemble d'opérations. Ensuite, chacun va envoyer une proposition à tous les agents robot en suivant des règles de priorité. Enfin, chaque robot aura son propre plan local bien équilibré.

5.4.1.1. Algorithme

Pour l'allocation des opérations aux robots, nous nous sommes basés sur les travaux décrits dans [46].

La figure 29 résume la procédure de la prise de décision du processus d'agents robots lors de l'étape opération affectation.

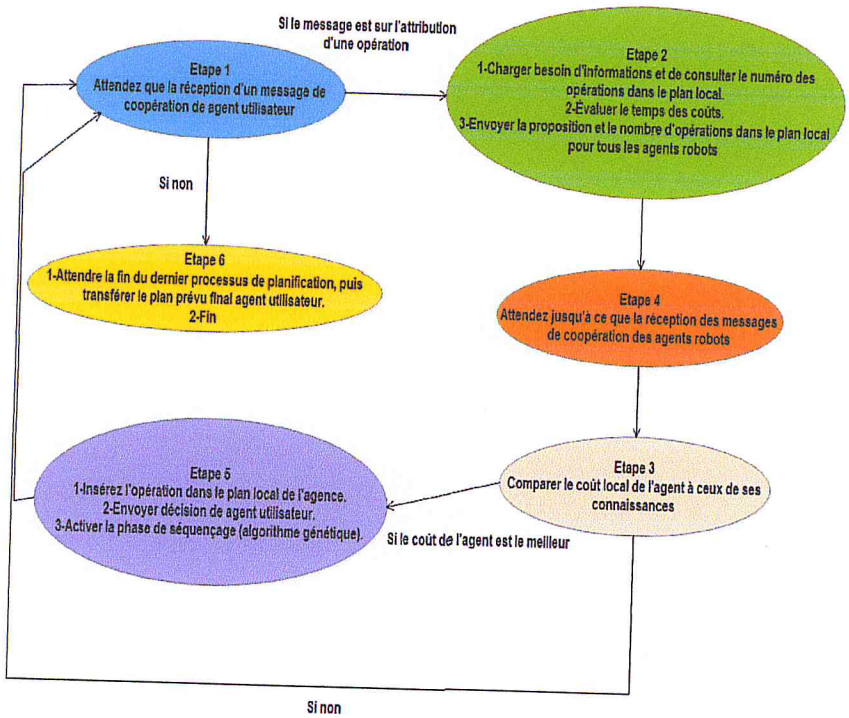


Figure 29 : Prise de décision du processus d'agents robots lors de l'étape opération affectation.

Ce travail exploite huit type de règle de priorité sont utilisées; ils sont donnés comme suit:

- Le plus petit temps de traitement (SPT).
- First In First Out (FIFO).
- Plus petit temps de disponibilité (SRT).
- Plus grand temps de traitement (GPT).
- Plus grand temps de traitement (GPT).
- Plus Meilleur temps de traitement (GBPT).

- Plus grande valeur de priorité (GPV).
- Plus petite valeur de priorité (SPV).

où:

- Meilleur temps de traitement (BPT) est le plus petit temps de traitement de l'opération entre ses différents temps de traitement sur les robots.
- Valeur de priorité (PV) d'une opération. Elle est donnée comme suit :
    - Si l'opération est sans successeur, son PV est égal à son temps de traitement
    - Si l'opération a des successeurs, son PV est égal à son temps de traitement additionné à la valeur de priorité de ses successeurs.

Concernant l'agent utilisateur, les détails sont donnés dans l'algorithme 10.

Algorithme 10 : Envoyer les tâches aux robots{

```
Nbr_agent();
Définir_nombre_opération(tâche); //tâche a exécuter
Créer_nouvelle_matrice(nombre_opération, nombre_tâche);
Créer_tableau_proposition(nbr_agent);
Décomposer_une_tâche();
Classer_opération_par_niveau(opération, niveau);
/*Cette méthode permet de classer les opérations qui n'ont pas de prédécesseurs dans le niveau 1, celles qui ont un seul prédécesseur dans le niveau 2, celles ayant deux prédécesseur dans le niveau 3 et ainsi de suite */
Ajouter_dans_matrice(opération, tâche);
Allocation_opération();
/*Allouer chaque opération à un et un seul robot selon ces compétences ; puis, remplir le tableau qui contient ces opérations allouées*/
INFORM (nom_agent, opération, niveau);
}
```

Quand l'agent utilisateur reçoit un message, il réagira selon le contenu du message ; l'algorithme 11 représente la réaction du l'agent.

Algorithme 11 : Arrivée d'un message agent utilisateur{

```
switch (contenu_message){
    case "absent" :
        Supprimer_agent(); //cas d'une panne l'agent sera éliminée
        break;
    case " set_temps_traitement " :
        mettre_a_jour_processing_time(temps_traitement);
        break;
    case " date_début " :
        mettre_a_jour_start_time(date_début);
        break;
    case "date_fin" :
        mettre_a_jour_End_time(date_fin);
        break;
    case "Récéption_Ok" :
        mettre_a_jour_récéption_proposition (true);
        //Cette méthode permet de définir si le robot a bien reçu son plan  local
        break;
    case "Alloc" :
        mettre_a_jour_Allocation(agent_robot, opération);
        //Cette méthode permet de définir le plan local final de chaque robot
        break;
}
}
```

Concernant l'agent robot, quand il reçoit un message, il réagira selon son contenu. L'algorithme 12 représente la réaction du l'agent robot.

Algorithme 12 : Arrivée d'un message agent robot{

```
switch (contenu_message){
    case "config" :
        Initialiser_local_plan();
```

```
/*Cette méthode permet de initialiser son plan local en fonction du tâche et les règles de priorité*/
                break;
            case "plan" :
                Calculer_proposition(opération);
            /* Cette méthode permet d'évaluer la distance entre la position actuel et l'emplacement de la cible et il calcule le temps de coût.*/
                INFORM(list_agents_robot, local_plan, temps_ coût);
                break;
            case "proposition" :
                chercher_meilleur_proposition();
                if (nombre_ meilleur_propo==1){
                    if (meilleur_proposition==True){
                    ajouter_opération_dans_local_plan(opération);
                    }
                } else {
                    AR=qui_prendre_opération ();
                    /* Cette méthode permet de retourner l'agent qu'ila le plus petit nombre d'opération*/
                    ajouter_opération_dans_local_plan(RA,  opération);

            }
             INFORM (agent_utilisateur, set_temps_traitement , temps_de_traitement);
             INFORM (agent_utilisateur, date_début , date_début);
             INFORM (agent_utilisateur, date_fin , date_fin);
        break;
    }
    }
}
```

5.4.1.2. Exemple

L'utilisateur veut transporter des objets de la salle une (01) à la salle n° trois (03). Le système dispose de sept robots : trois (03) robots manipulateurs, deux (02) robots mobiles et deux (02) robots manipulateurs mobiles. La tâche sera partagée sur les robots comme suite :

- Opération1 : le robot manipulateur 0l (MA001) va soulever l'objet et le mettre sur le robot mobile n° 01.
- Opération 2 : le robot mobile 01 (MO001) va transporter l'objet à la salle n° 02.
- Opération 3 : le robot manipulateur (MA002) qui se trouve devant la porte va ouvrir la porte.
- Opération4 : le robot manipulateur mobile n° 01 (MM001) va ouvrir la deuxième porte.
- Opération5 : le robot manipulateur (MA008) va soulever l'objet et le mettre sur un autre robot mobile MO022.
- Opération 6 : le robot mobile (MO022) va transporter l'objet à la salle n° 03.
- Opération7 : le robot manipulateur mobile (MM001) va ouvrir la porte.
- Opération 8 :le robot manipulateur mobile n° 02 (MM002) va ouvrir la porte de la troisième salle.
- Opération 9 : déposer l'objet à sa place.

L'objet se trouve devant un robot manipulateur; le système va lui affecté l'opération de soulever l'objet et le déposer sur le robot mobile afin de le transporter. Il va aussi allouer une autre opération à un robot manipulateur pour ouvrir la porte (ce robot doit se trouver devant la porte). Un autre robot se chargera de l'ouverture de la deuxième porte. Un quatrième robot(mobile) va se joindre à un autre robot manipulateur qui va soulever l'objet et le mettre sur un autre robot mobile pour déplacera l'objet la salle n° 03. Enfin, un robot manipulateur mobile se chargera d'ouvrir la porte et de saisir l'objet.

5.4.2. Ordonnancement

Après l'affectation des opérations à chaque robot, cette étape permet l'organisation et le séquençage des opérations en prenant en considération les contraintes de précédence. Chaque robot doit exécuter son plan à une durée précise, le système peut exécuter plusieurs opérations simultanément ou bien une opération ne peut pas démarrer avant qu'un autre robot ait terminé son opération.

L'objectif de l'ordonnancement est de minimiser le temps d'exécution total en appliquant les algorithmes génétiques.

Un algorithme génétique à base de règles priorité est utilisé pour séquencer les opérations attribuées en minimise le temps d'exécution total du robot.

Les chromosomes dans la plupart de la population initiale sont séquencés en utilisant les règles de priorité comme indiqué dans la figure 30, dans lequel chaque niveau est séquencé en utilisant une règle. De plus, nous générons des chromosomes au hasard (séquencés) afin de maintenir la diversité de la population. Dans l'AG, la taille de la population n'est pas constante ; elle augmente pour chaque génération après avoir effectué les opérateurs de croisement et de mutation.

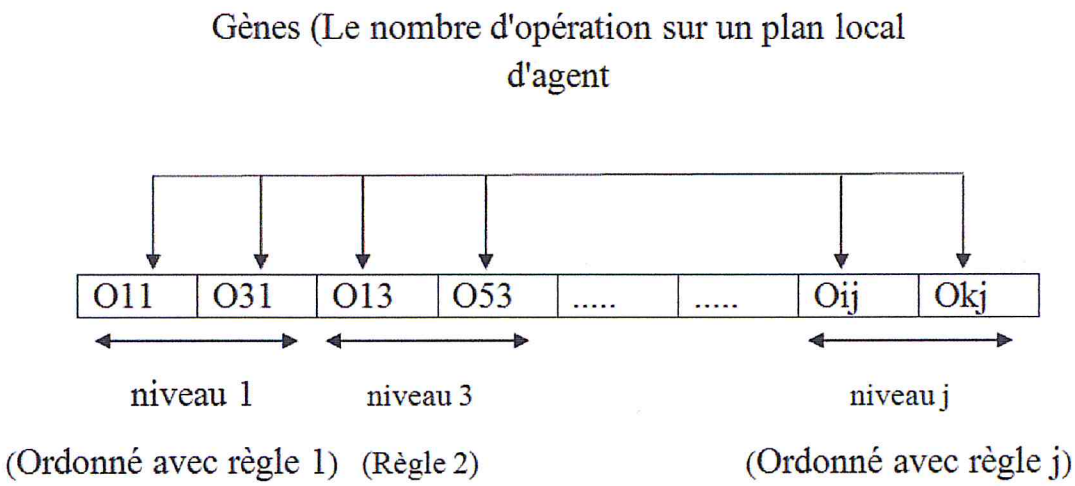


Figure 30 :Ordonnancement.

- **Opérateur de sélection et de croisement :** Nous utilisons l'approche de sélection Roulette. Une fois que la paire de chromosomes est sélectionnée, deux points sont générés de façon aléatoire afin de réaliser l'opérateur de croisement.
- **Opérateur de mutation :** la mutation est réalisée de la manière suivante (figure 31) :
  - Sélectionner aléatoirement deux gènes dans le chromosome.
  - Sans violer les contraintes de précédence, permuter les gènes sélectionnés.
  - Évaluer l'aptitude de la progéniture mutée alors cette progéniture sera ajoutée à la population en respectant la conservation des parents dans la population.

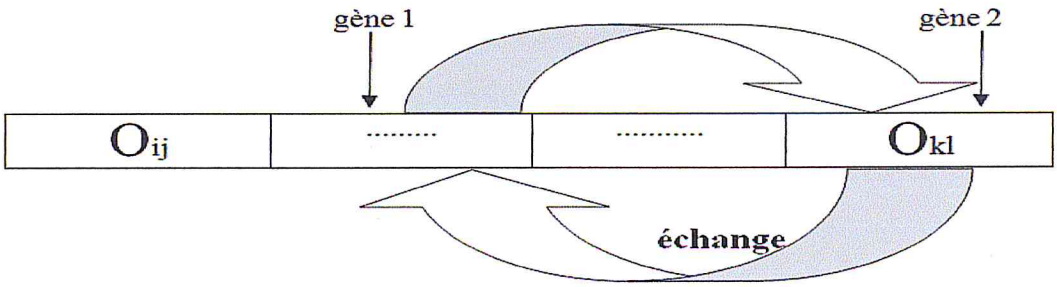


Figure 31 :Mutation.

5.4.2.1. Exemple

Le système doit mettre en ordre les opérations comme suite :

Opérations	Désignation	Prédécesseur	Durée
Opération 1	soulever l'objet et le mettre sur le robot	–	10s
Opération 2	transporter l'objet à la salle n° 02	Opération 1	30s
Opération 3	ouvrir la porte de la salle n° 01	Opération 2	15s
Opération 4	ouvrir la porte de la salle n° 02	Opération 3	15s
Opération 5	soulever l'objet et le mettre sur un autre robot mobile	Opération 4	15s
Opération 6	transporter l'objet à la salle n° 03	Opération 5	30s
Opération 7	ouvrir la porte pour sortir de la salle n° 02	Opération 6	15s
Opération 8	ouvrir la porte de la salle n° 03	Opération7	30s
Opération 8	déposer l'objet à sa position finale	Opération 8	20s

Tableau 4 : Contraintes temporelle et de précédence entre les tâches

5.4.3. Exécution

Après l'allocation et l'ordonnancement, le(s) premier(s) robot(s) commence(nt) l'exécution selon le plan généré lors de la phase d'ordonnancement. Lorsqu'un robot termine son opération, il en informe les autres robots qui sont entrain de l'attendre afin de commencer les leurs.

6.Conclusion

Dans ce chapitre, nous avons décrit la conception de la solution proposée en commençant par la description de l'environnement, ainsi que les différentes entités composant le système, les algorithmes qui s'exécutent au niveau de chaque agent, et les interactions entre les agents. Ensuite, nous avons présenté avec les différents diagrammes de conception en utilisant le modèle orienté objet et le langage de modélisation Agent UML.

Grâce à l'efficacité et la souplesse du modèle multi-agents proposé, le travail peut être modifié et étendu très facilement ; par exemple, en modifiant les algorithmes des différents

Chapitre 04

Implémentation de la solution proposée

agents du système ou encore en rajoutant d'autres modules et fonctionnalité pour améliorer ou s'adapter à l'exécution des tâches, etc.

L'implémentation de la solution proposée fera l'objet du prochain chapitre.

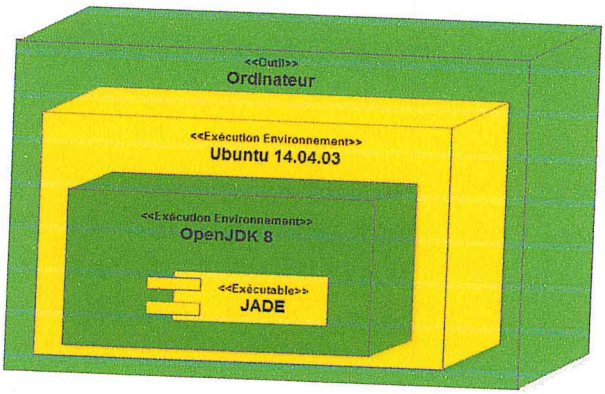


Figure 32 : Diagramme de déploiement du système.

2.2. Outils et langages de programmation

2.2.1. Plateforme JADE

Il existe plusieurs plateformes pour implémenter les SMA. 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, vu ses avantages que nous résumons dans ce qui suit [32]:

- simplifier la construction des SMA interopérables.
- fonctionner sous tous les systèmes d'exploitation.
- inclure tous les composants obligatoires qui contrôlent un SMA.
- 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 [32]:

- 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 de plateforme FIPA (ou non FIPA) éventuellement distantes.

Chapitre 04

Implémentation de la solution proposée

1. Introduction

Après avoir exprimé les objectifs de notre travail ainsi que les solutions proposées en utilisant le modèle orienté objet et le langage de modélisation AgentUML (Agent Unified Modeling Langage), nous allons passer maintenant à l'implémentation de notre IHR développée en utilisant le langage de programmation JAVA, en utilisant l'EDI NetBeans et la plateforme JADE (Java Agent DEvelopment Framework).

L'objectif de ce présent chapitre est l'implémentation de cette interface. Nous commençons par la description des environnements logiciels nécessaires (JADE, Netbeans et MySQL). Après, nous détaillons les différentes étapes de la mise en œuvre de l'interface homme/robot de contrôle proposée.

2. Environnements de développement

2.1. Architecture physique de déploiement

La mise en œuvre de l'interface proposée est réalisée sur un ordinateur doté d'un système d'exploitation Ubuntu 14.04.03 sur lequel est installé le Moteur OpenJDK Java 8, la plateforme Java Agent DEvelopment Framework (JADE) et le système de gestion de la base de données MySQL. La manière dont les composants physiques du système sont organisés est illustrée dans le diagramme de dépoilement suivant (Figure 32) :

JADE est composée de plusieurs containers (réceptacles) d'agents (figure 33). 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) [32].

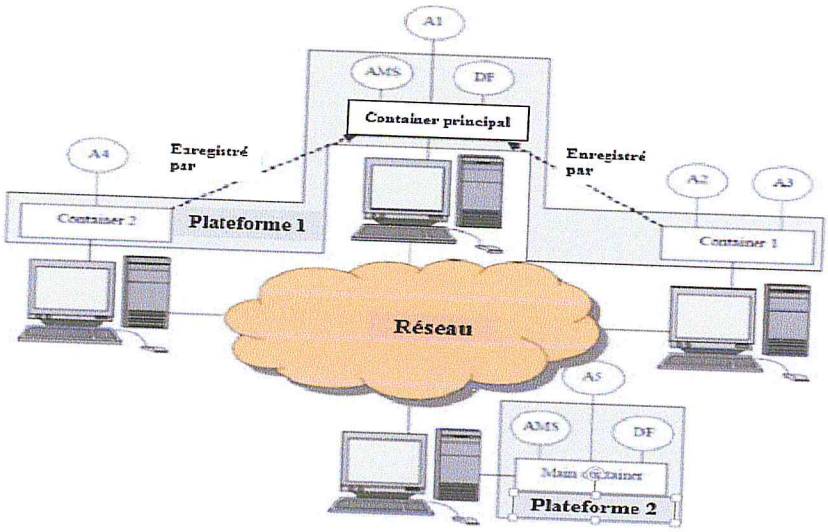


Figure 33 : Plateforme et conteneurs de JADE [32].

La plateforme offre une interface graphique utilisateur GUI (figure 34) pour la gestion à distance des agents, la communication entre les agents et l'interface (GUI).

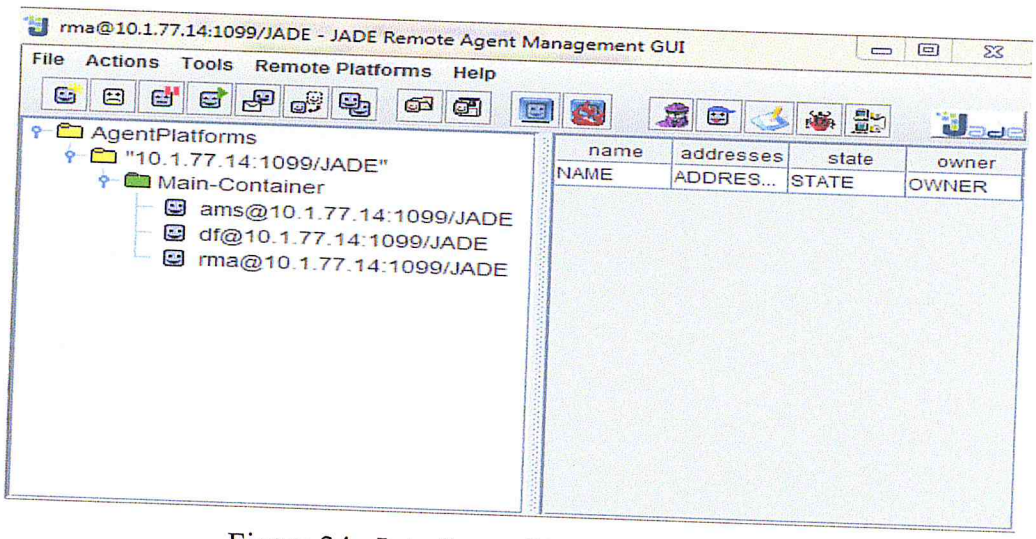


Figure 34 : Interface utilisateur (GUI) [32].

Deux outils graphiques sont disponibles :

- Agent Dummy : il 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 de messages entre les différents groupes d'agents en utilisant une notation proche d'UML.

2.2.2. Langage Java

Afin de réaliser notre interface permettant aux utilisateurs de contrôler les robots, 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).
- Une programmation orientée objet et une bibliothèque immense d'objets. Dès sa naissance, les programmeurs de Sun ont doté leur langage d'une des plus grandes bibliothèques d'objets prête à l'emploi.

2.2.3. NetBeans

NetBeans est un projet open source ayant un succès et une base d'utilisateurs très large. Conçu en Java, NetBeans est disponible sous Windows, Linux, Solaris (sur x86 et SPARC), Mac OS X. Aujourd'hui, deux projets existent [31] :

- EDI NetBeans : c'est un environnement de développement, un outil pour les programmeurs pour écrire, compiler, déboguer et déployer des programmes. Il est écrit en Java, mais peut supporter n'importe quel langage de programmation. Il y a également un grand nombre de modules pour étendre l'EDI NetBeans.
- Plateforme NetBeans : c'est une fondation modulable et extensible utilisée comme brique logicielle pour la création d'applications bureautiques. Les partenaires

privilégiés fournissent des modules à valeurs ajoutées qui s'intègrent facilement à la plateforme et peuvent être utilisés pour développer ses propres outils et solutions.

2.2.4.MySQL

MySQL est un système de gestion de bases de données relationnelles (SGBDR). Il fait partie des logiciels de gestion de base de données les plus utilisés au monde. Il est un serveur de base de données SQL très rapide, multithread, multi-utilisateurs et robuste. MySQL est développé, distribué et supporté par MySQL AB.

2.3. Configuration de Jade sous NetBeans

Pour configurer Jade avec NetBeans, nous avons procédé de la manière suivante (voir la figure 35 pour plus d'information) :

- Télécharger le package "jade.jar" à partir de lien suivant : http://www.java2s.com/Code/Jar/j/Downloadjadejar.htm.
- Démarrer NetBeans et ouvrir le projet.
- Ajouter les fichiers jar dans : ProjectProperties->Librairies -> Compile, Properties -> Librairies ->Run, Properties -> Librairies -> Compile Tests, en cliquant sur le bouton Add JAR/Folder, puis sur parcourir "jade.jar".

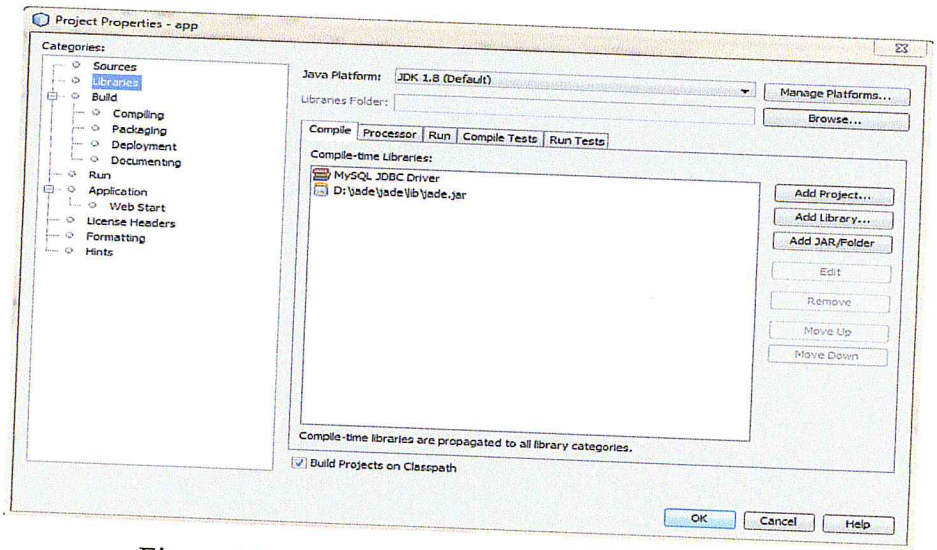


Figure 35 :Configuration de NetBeans avec Jade.

3. Mise en œuvre de l'interface proposée

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

3.1. Main-Container

Le Main-Container est créé lors du lancement de la plateforme afin de lancer tous les autres containers (figure 36).

```
public class MainContainer {
  public static void main(String[] args) {
                try {
              Runtime rt=Runtime.instance();
              Properties p=new ExtendedProperties();
              p.setProperty("gui", "true");
              ProfileImpl pc=new ProfileImpl(p);
              AgentContainer ag=rt.createMainContainer(pc);
              ag.start();}
                  catch (Exception e){
                        System.out.println(e);
                  }

        }
}
```

Figure 36 :Création du Main-Container.

3.2. 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. La création d'un agent (figure 37) 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.

```
public class Container {
    public static void main(String[] args) {
            try {
              //Création du container
               jade.core.Runtime rt=jade.core.Runtime.instance();
               ProfileImpl pc=new ProfileImpl(false);
               pc.setParameter(ProfileImpl.MAIN_HOST,"localhost");
               AgentContainer ac=rt.createAgentContainer(pc);
               //Création du l'agent
              AgentController ac1=ac.createNewAgent
            ("Agent Utilisateur", AgentUtilisateur.class.getName(), new Object[]{});
              ac1.start();

            } catch (Exception e) {
               System.out.println(e);
            }
```

Figure 37 : Création de l'agent Utilisateur

3.3. Agent Utilisateur

3.3.1. Interface Utilisateur

Ce module a pour rôle d'assurer la communication entre les utilisateurs et les agents.

3.3.1.1. Envoi des informations

Les informations saisies par l'utilisateur seront envoyée à la classe agent utilisateur par un objet GUIEvent qui contient les informations. Cet objet nécessite deux informations : la source (l'interface ou les information sont saisies) et le type d'événement qui est un entier.

Dans notre cas, le type d'événement de contrôler un robot est 1,un ensemble de robots est 2,tout le système est 3,choix autonome est 4.Nous stockons les paramètres dans un objet HashMap qui doit contenir l'agent robot, l'opération, l'objets concerné ainsi que les positions cibles.

```
ajj();//m"thode pour ajouter les données saisis a la BDD
String tache = tache1+ "#" +priritaire+ "#" +objet+ "#" +posx+ "#" +posy+ "#" +posz;
Boolean a=nn(nom);//nn est une methode qui retourne true si le robot choisit est selectionnée deja
if(a==true){//Si le robot qu'on veut controler est déja selectionné
if (controle==1){//Si on veut controler un seul robot
        GuiEvent gev=new GuiEvent(this, 1);
        Map <String,String> params=new HashMap<>();//déclaration map
        params.put("nom_agent", nom);//le nom
        params.put("tâche", tache);//la tache contient la tache, l'objet, les position cibles
        gev.addParameter(params);//on ajoute les param dans GuiEvent
        user.onGuiEvent(gev);//user une variable de type AgentUtilisateur
        }
      if (controle==2){//Si on veut controler un ensemble de robot
        GuiEvent gev=new GuiEvent(this, 2);
        params2.put("nom_agent", nom);
        params2.put("tâche", tache);
        gev.addParameter(nom);
        gev.addParameter(tache);
       user.onGuiEvent(gev);
        }}
```

Figure 38 :Envoi des informations à l'agent utilisateur

3.3.1.2.Récupération des informations

Au niveau de l'agent utilisateur récupère l'événement. Pour gérer ces événements, nous avons utilisé un Switch. Il réagit selon le type d'événement récupéré.

```
@Override
public void onGuiEvent(GuiEvent ge) {
   switch (ge.getType()) {//ge.gettype pour récupérer l'évenement
case 2 :
   nom=(String)finale.params2.get("nom_agent");//nom d'agent robot
    tache=(String)finale.params2.get("tâche");//les informations
    tableau=tache.split("#");//décomposistion du message
    tach=tableau[0];
    priorité=Integer.parseInt(tableau[1]);
    objet=tableau[2];
    posX=Integer.parseInt(tableau[3]);
    posY=Integer.parseInt(tableau[4]);
    posZ=Integer.parseInt(tableau[5]);
     y=2;//indice controler un ensemble de robot
    String y3=Integer.toString(y);
   ACLMessage ac20=new ACLMessage(ACLMessage.REQUEST);
   ac20.addReceiver(new AID(nom,AID.ISLOCALNAME));
   String message1=tableau[0]+ "#" +tableau[1]+ "#" +tableau[2]+ "#" +tableau[3]+ "#" +tableau[4]+ "#" +tableau[5]+ "#" +y3,
   ac20.setContent(message1);
   send(ac20);

   break;
```

Figure 39 :Récupération des informations

3.3.1.3. Affichage des résultats

Ce module permet aussi l'affichage des résultats par la méthode afficherMessage (String message, Boolean ligne), ligne prend la valeur true si le textArea n'est pas vide.

- Au niveau d'interface : c'est une méthode on lui passe en paramètre le message ainsi un booléen qui indique si c'est le premier message ou non pour le retour à la ligne (figure 40).

```
public void AfficherMessage(String msg,Boolean ligne){
 if(ligne==true){
     aaa.append(msg+" \n");
 } else {
     aaa.append(msg);
 }
}
```

Figure 40 :La méthode d'affichage.

- Au niveau de l'agent : l'interface est associée à l'agent, donc il appelle la méthode afficherMessage.

```
f.AfficherMessage("_____");
f.AfficherMessage(sen+" a effectuer l'opération "+opp+" sur l'objet "+obb+" il a comencé le travail a ",true);
f.AfficherMessage(dee+ " secode et il a fini l'exécution a "+dff+" seconde il a prend une durée de "+duu+" seconde", true);
f.AfficherMessage("avec une capacité de saisie "+caa+"g et une vitesse de "+vita+"m/s", true);
f.AfficherMessage("_____", true);
```

Figure 41 :La commande d'affichage.

3.3.2. Gestion des tâches

Cette méthode permet de décider de l'acceptation ou du refus d'une tâche selon le cas du contrôle :

- **Contrôler un seul robot :** le robot propose la durée à partir de laquelle il va commencer l'exécution de la tâche. Si la proposition convient à l'agent utilisateur (l'agent utilisateur choisit la durée à partir de laquelle il va commencer l'exécution de la tâche de façon autonome sans intervention d'utilisateur), il va acceptera la tâche ;sinon, il rejette sa proposition.
- **Contrôler un ensemble de robot :** chaque robot propose la durée à partir de laquelle il va commencer l'exécution de la tâche (seulement les robots sélectionnes par l'utilisateur), l'agent utilisateur recevra toutes les propositions vérifiera chaque proposition avec la durée qu'il a calculé. Si toutes les durées sont inférieures ou égale à sa durée (toutes les durées sans exception) il acceptera la tâche, sinon il refusera la tâche.
- **Contrôler tous les robots :** se fait de la même manière que le deuxième cas (contrôler un ensemble de robot) sauf que cette fois tous les robots qui se trouvent dans le système sont concerné. Un vue partiel du code dans la figure 42.

```
ACLMessage ac=receive();//methode pour recevoir le message
if (ac!=null){
    String type_message=ACLMessage.getPerformative(ac.getPerformative());//type de message
    switch (type_message){
        case "PROPOSE" :
        String req=ac.getContent();//le contenu de la proposition
       tableau=req.split("#");//décomposition du contenu
        int k1=Integer.parseInt(tableau[1]);//la durée à partir de laquelle il va commencer l'exécution de la tâche
       if (y==1){//controler un seul robot
        if (k1<random){//random durée calculé pat l'agent robot
        int att=attendre(k1);//convertir la durée (k1) en ms
        f.AfficherMessage("La tâche a été "+tableau[0]+" par le robot "+ac.getSender().getLocalName()+" Elle comencra dan
         try {
        Thread.sleep(att);//attendre que le robot finira son travail
       } catch (InterruptedException ex) { JOptionPane.showMessageDialog(null, ex); }
        String message=tableau[2]+ "#" +tableau[3]+ "#" +tableau[4]+ "#" +tableau[5]+ "#" +tableau[6]+ "#" +tableau[7]+ "
        ACLMessage ac1=ac.createReply();
          ac1.setPerformative(ACLMessage.ACCEPT_PROPOSAL);
          ac1.setContent(message);
          send (ac1);}
```

Figure 42 :Gestion des tâches.

3.3.3. Communication

Cette méthode envoie les messages 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.

```
ACLMessage ac20=new ACLMessage(ACLMessage.REQUEST);//créer un aclMessage et nous choisissons le type de message
ac20.addReceiver(new AID(nom,AID.ISLOCALNAME));//on entre le nom d'agent qu'on veut l'envoyer le message
String message1=tableau[0]+ "#" +tableau[1]+ "#" +tableau[2]+ "#" +tableau[3]+ "#" +tableau[4]+ "#" +tableau[5]+ "#" +y3;
ac20.setContent(message1);//on ajoute le contenu de message
send(ac20);//envoyer le message
```

Figure 43 :Communication.

3.3.4. Planification

Le module planification concerne seulement la partie choix autonome, le rôle de l'agent utilisateur est la décomposition des tâche puis envoyer un par un au agent robot en respectant les compétences de chaque robot.

Tout d'abord, agent utilisateur ajoute à chaque opération d'un niveau de priorité, qui représente son degré de priorité. Par exemple, soit T1={O11, O12 ...} être O1j Tâche 1 qui consiste en une séquence prédéterminée d'opérations j à être successivement traitées (O11, O12 ... O1j). Agent utilisateur ajoute O11 au niveau L1, O22 au niveau L2 ... O1j au niveau Lj, et ainsi de suite pour les autres tâches T2, ..., Ti.

Agent utilisateur commence les opérations de niveau L1, L2 suivie par l'envoi, jusqu'à atteindre le dernier niveau. La sélection d'une opération d'envoi de chaque niveau est effectuée selon un procédé proposé en utilisant les règles décrites ci-dessus. Agent utilisateur ordonne les opérations de ce niveau en utilisant des règles de priorité, de la plus haute priorité à la plus faible priorité. Les hypothèses suivantes ont été envisagées :

- Opérations sans prédécesseurs sont classées dans le premier niveau en utilisant des règles de priorité; ils sont les premiers à être expédié.
- Les successeurs des opérations du premier niveau sont classés dans le second; leurs successeurs sont classés au troisième niveau et ainsi de suite jusqu'à ce que le dernier niveau. Ces niveaux sont commandés en utilisant également des règles de priorité.
- Pour les chaque niveau commandé, les opérations de répartition commence à partir de la première opération du niveau vers le dernier, du premier niveau au dernier niveau.

```
// startTime = Environment.TickCount;
    for (int t = 0; t < n_operation; t++) {
        for (int tt = 0; tt < n_jobs; tt++) {
            OperationMatrix[t][tt].id_operation = t + 1;
            OperationMatrix[t][tt].id_job = tt + 1;
            OperationMatrix[t][tt].release_time = 0;
            OperationMatrix[t][tt].end_time = 0;
            OperationMatrix[t][tt].AllocationOk = false;

            // Sup
            OperationMatrix[t][tt].Processing_time=get_smallest(processTime_4_5(tt,t));
        }


    Operations_Announcing(level, 0, 1); //the first level
    break;
```

Figure 44 :Planification.

3.4. Agent robot

3.4.1. Mise à jour

Lorsque le robot navigue dans son environnement et il détecte un nouvel objet qu'il n'existe pas déjà dans la base de données, ce robot envoie les caractéristiques de cet objet à ce module qui va l'y insérer grâce à la requête INSERT. Ce module permet aussi de garder l'historique ; lorsque le robot détecte un autre robot, un objet ou une personne, il va l'ajouter à la base de données. La MAJ est présenté dans la figure 45.

```
 ArrayList al = new ArrayList();
try {st=con.obtenirConnection().createStatement(); }//connexion a la BDD
catch (Exception e){JOptionPane.showMessageDialog(null, e.getMessage()); }
    try {rs=st.executeQuery("select tagRFID from objet");//selectionne les objet qui existe
    while (rs.next()){al.add(rs.getString("tagRFID"));}//ajouter les objet dans un tableau
 } catch (SQLException ex) {
    JOptionPane.showMessageDialog(null, ex);
 }   Boolean x=al.contains(tag);//on verifier si l'objet detecté existe déja dans la BDD
  if (x==false){//s'il n'existe pas on l'ajout a la BDD
    String sql = "insert into objet (id_o,nom,nature,poid,tagRFID) "
        + "VALUES  ('" +o1+ "','" +o2+  "','" +o3+  "','" +o4+ "','" +tag+"')";
     try {
      st.executeUpdate(sql);//Exécution de la requet sql
}catch (Exception e){JOptionPane.showMessageDialog(null , "erreur d'ajout " +e.getMessage());}
    }
```

Figure 45 :Mise à jour.

3.4.2. Traitement

Lorsque l'agent utilisateur envoie un ACCEPT_PROPOSAL ce module va traiter les priorités de chaque opération, et il autorise les robots qui ont une priorité la plus inférieur. Ou lorsqu'un robot finira son travail, il informera tous les agents robots, quand l'agent robot

reçoit cette information ce module va faire un autre traitement en respectant les précédence pour autoriser le robot pour qu'il commence le travail.

```
public void exécution (String sender,int cc,String tac1,String ob1,int x1,int y1,int z1,int pp,ACLMessage bb2){/
    if (sender.equals("Agent Utilisateur")){//sender est l'agent utilisateur
        int prioA=1;//les opération de premier niveau
        if (cc==1){ //controle un seul robot
            Boolean hh=false;
            if (hh==false){
                aide(this.getLocalName());
            }
            else { choisirtache(tac1, ob1, x1, y1, z1);//exécuter selon la tache qu'il a recoit
                envoyeruser();}//informer agent utilisateur
        }
        if (cc==2){//controle un ensemble de robot
           if (pp==prioA){//vérification les regles de précédence
            choisirtache(tac1, ob1, x1, y1, z1);
            prioA++;//incrementer le niveau  des opération
              envoyer2cas(prioA);//envoyer aux autre robot qu'il a exécuter sa tâche
               envoyeruser();//informer agent utilisateur
           }
```

Figure 46 :Traitement.

3.4.3. Allocation des opérations aux robots

Quand un agent robot reçoit une opération, il évalue la distance entre la position actuel et l'emplacement de la cible et il calcule le temps de coût, puis cette agent transmet ca proposition a tous les agents robot, lorsque il reçoit tous les coûts, chaque agent robot compare son coût avec les autre, l'agent qui a proposé le meilleur temps des coûts aura l'opération. Si deux ou plusieurs agents ont proposé le même meilleur coût, l'agent ayant le plus petit nombre d'opérations dans son plan local, il obtiendra l'opération. Ceci est fait pour calibrer la charge sur tous les robots. Enfin, l'agent robot qui détient l'opération envoie sa décision de l'agent utilisateur.

La figure 47 ci-après donne une vue partielle du code d'allocation des différentes opérations aux robots.

```
message_for_missing_prop = "GetProp#" + eiid_j + "#" + eiid_o;
if (proposition[agent_id] == 0) {//chang
    proposition[agent_id] = cost;
    n_propo_recu++;
    op_allocated[agent_id] = nomber_op_all;
}
OperationProposition oo=new OperationProposition();
oo.cost[agent_id] = cost;
if (n_propo_recu == n_agents-3) {
    for (int u = 0; u < n_agents; u++) {
        proposition[u] = 0;
    }
    n_propo_recu = 1;
    an_timer = false;
    Allocation_operation(eiid_j, eiid_o);//préparer le plan local
```

Figure 47 :Allocation des différentes opérations aux agents robots

3.4.4. Ordonnancement en utilisant les algorithmes génétiques (AG)

Pour chaque opération allouée, l'agent robot correspondante des séquences de son plan local en utilisant l'algorithme génétique à base de règles de priorité.

```
public Operation[] ScheduleOperationLocalPlan(String sched_Rul, String benchmark) {

    Operation[] op1 = new Operation[nu_operation_priority1];
    Operation[] op2 = new Operation[nu_operation_priority2];
    Operation[] op3 = new Operation[nu_operation_priority3];
    Operation[] op4 = new Operation[nu_operation_priority4];
    Operation[] op5 = new Operation[nu_operation_priority5];
    Operation[] op6 = new Operation[nu_operation_priority6];
    Operation[] op7 = new Operation[nu_operation_priority7];
    Operation[] op8 = new Operation[nu_operation_priority8];
    Operation[] op9 = new Operation[nu_operation_priority9];
    Operation[] op10 = new Operation[nu_operation_priority10];
    Operation[] op11 = new Operation[nu_operation_priority11];
    Operation[] op12 = new Operation[nu_operation_priority12];
    Operation[] op13 = new Operation[nu_operation_priority13];
    Operation[] op14 = new Operation[nu_operation_priority14];
```

Figure 48 : Ordonnancement

5.Affichage du plan d'opérations

Pour l'affichage des résultats d'allocations et d'ordonnancement des opérations à exécuter par le système robotique, nous avons utilisé le diagramme de Gantt.

5.1. Configuration

Pour la configurer la classe Diagramme de Gantt sur Netbeans, nous avons procédé de la manière suivante :

- Télécharger le package "jfreechart.jar" et "jcommon-1.0.16.jar".
- Démarrer NetBeans et ouvrir le projet.
- Ajouter les fichiers jar dans : ProjectProperties -> Librairies -> Compile, en cliquant sur leboutonAdd JAR/Folder, puis sur parcourir les deux bibliothèques.

5.2. Description

On déclare un TaskSeries ou on va ajouter l'ensemble des tâches et des robots, les tâches seront ajoutées dans cette série grâce à la commande taskserie.add (idtâche).On déclare l'objet Task, on définit la tâche par son nom, début d'exécution et la fin d'exécution, les opérations seront ajoutées grâce à la commande task.addSubtask (nom de l'opération). On déclare un autre objet Task pour définir une opération, on passe en paramètre son nom, début d'exécution et la fin d'exécution.

5.3. Exécution

Après l'exécution de la classe Gantt la figure suivante (figure 49) montre les résultats d'allocations et d'ordonnancement obtenue.

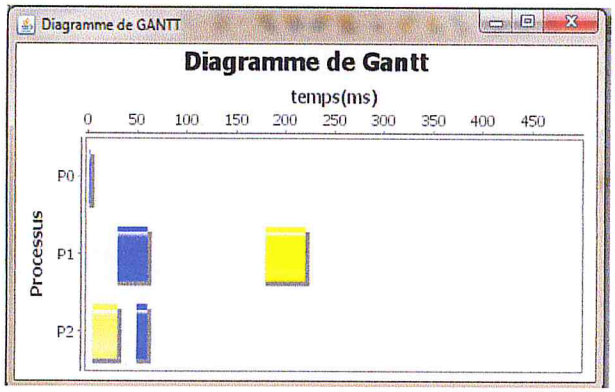


Figure 49 :Diagramme de Gantt.

6.Conclusion

Dans ce chapitre, nous nous sommes intéressés à l'implémentation de l'interface homme/robot de contrôle proposée. Nous avons d'abord présenté les points importants de notre implémentation, en nous décrivant les environnements de développements utilisés. Ensuite, nous avons décrit l'implémentation des différentes classes utilisées. Enfin, le processus d'allocation et d'ordonnancement des opérations sur les différents robots a été implémenté en utilisant les règles de priorité et les algorithmes génétiques.

La validation de la solution proposée et implémentée fera l'objet du cinquième et dernier chapitre.

Chapitre 05

Validation de la solution proposée

Chapitre 05

Validation de la solution proposée

1. Introduction

Le développement, la maîtrise, la cohérence et l'efficacité du fonctionnement d'une interface homme/robot de contrôle se montre par sa validation. Afin de pouvoir valider l'interface 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 de quatre différents scénarios de validation considérés. Il y a lieu de préciser que lors de la réalisation de ces trois premiers scénarios, l'opérateur humain effectue lui-même la répartition des opérations sur les robots. De plus, il se charge aussi de l'ordonnancement des opérations allouées aux robots. Les trois premiers scénarios sont considérés dans les deux cas d'acceptation et de refus de la tâche assignée au SRCP. Enfin, nous évoluons les principaux résultats obtenus.

2. Présentation de l'interface homme/robot de contrôle

Nous avons réalisé une application sécurisée qui se compose d'un ensemble de fenêtres graphiques afin de gérer toutes les entités du système.

2.1. Interface d'authentification et de contrôle

Lorsque l'opérateur lance l'exécution de l'application, l'interface de contrôle donnée par la figure 50 apparaît, portant l'interface d'authentification.

Figure 50 : Authentification.

Après avoir été authentifié, l'utilisateur est autorisé à poursuivre. Il doit initialiser une tâche, après il choisit le type de contrôle (manuel ou autonome). S'il choisit le contrôle manuel, il doit déclarer le(s) robot(s) qu'il veut contrôler ; puis, l'utilisateur doit remplir le formulaire donné par la figure 51 (en haut à droite). Dans le cas contraire (contrôle autonome), l'utilisateur sélectionne une des tâches prédéfinies (benchmark) et la(es) règles de priorité à considérer (SPT ou GPT). Enfin, l'utilisateur lance l'exécution de la tâche.

Figure 51 : Interface de contrôle.

2.2. Interface menu principal

Après authentification, l'utilisateur doit cliquer sur « initialiser une tâche » ; l'interface suivante représentée par la figure 52 apparaît. Dans cette interface, il peut sélectionner l'environnement de travail, gérer la liste des environnements, des personnes, des objets, des robots, etc.

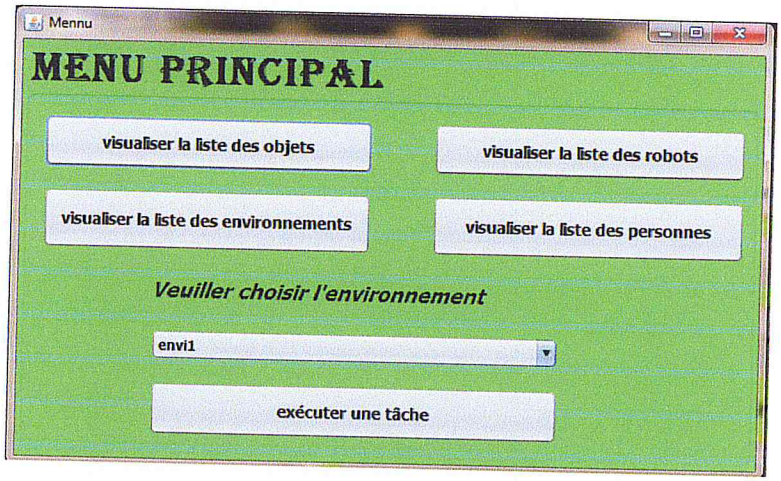


Figure 52 : Interface menu principale.

2.3. Interface gestion des environnements

Cette interface, donnée par la figure 53, permet de visualiser, ajouter, supprimer, rechercher des environnements dans la base de données. L'interface de gestion des robots, des objets, des personnes est implémentée de la même manière.

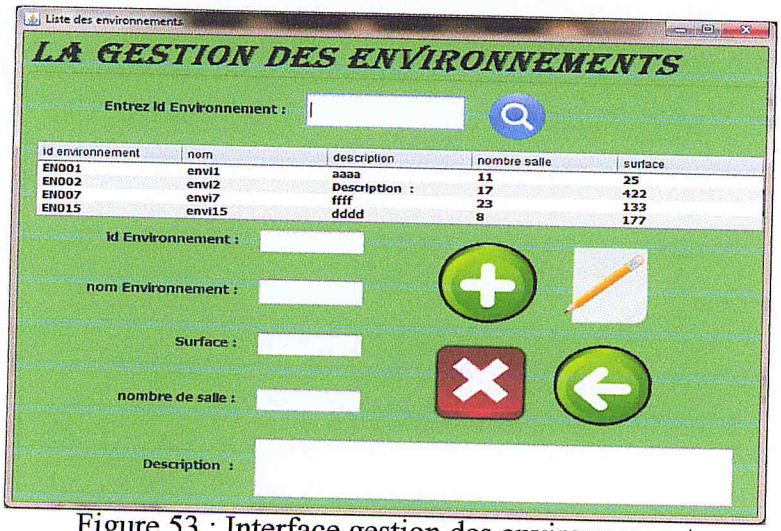


Figure 53 : Interface gestion des environnements.

2.4. Interface de configuration

Dans cette interface se trouve deux champs : le premier champ concerne le nombre de robots existants dans le système ; le deuxième champ concerne les objets. L'utilisateur doit remplir ces deux champs puis ajouter les robots et les objets en respectant les nombres saisis. Si le robot existe déjà dans la base de données, l'utilisateur le sélectionne puis modifie ses informations. Plus de détails sont données dans la figure 54.

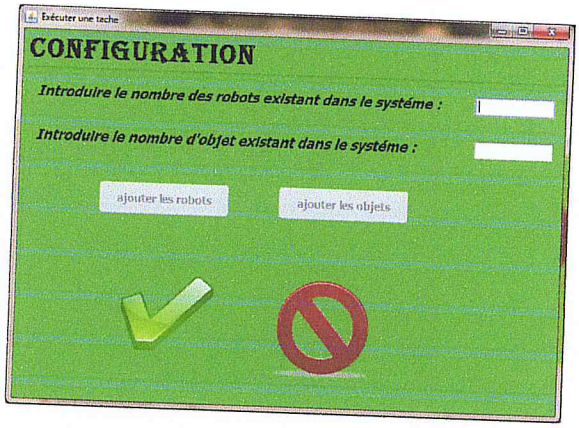


Figure 54 : Interface configuration.

2.5. Interface de l'historique

Cette interface permet aux utilisateurs de visualiser l'historique de l'exécution d'une tâche quelconque (figure 55).

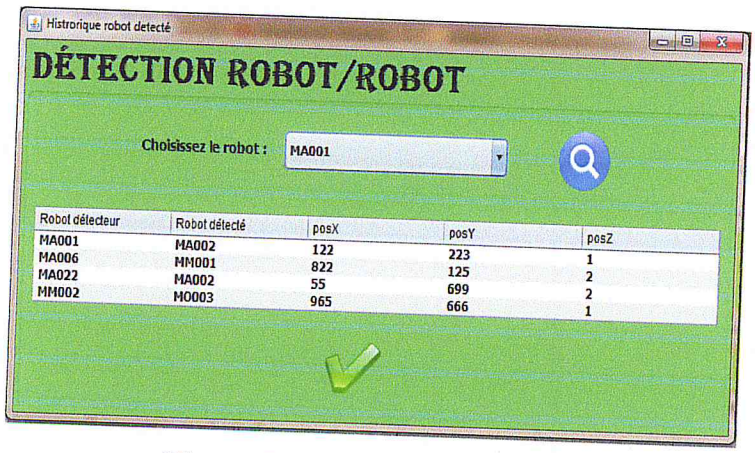


Figure 55 : Interface historique.

3. Déroulement de l'application développée

3.1. Interface utilisateur

L'interface utilisateur facilite la communication entre l'utilisateur et le système multi-agents de contrôle. Lorsque l'opérateur lance l'exécution de l'application, il doit s'authentifier telle que montrée dans la Figure 55.

Après avoir été authentifié, l'utilisateur est autorisé à poursuivre. Lorsqu'il veut réaliser une tâche, il doit suivre les étapes décrites ci-dessous :

- Initialiser une nouvelle tâche dans la barre du menu (Fichier→ Initialiser une tâche).
- Si l'environnement de travail n'existe pas encore, il faut l'insérer. Pour ce faire, l'utilisateur doit cliquer sur « liste des environnements », puis « ajouter environnement » en remplissant le formulaire correspondant.
- Sélectionner l'environnement et cliquer sur « Exécuter une tâche » (étape obligatoire).
- Introduire le nombre de robots existant dans l'environnement ; puis, ajouter les nouveaux robots ou modifier des robots existants (étape obligatoire).
- Introduire le nombre d'objets existants dans l'environnement ; puis, ajouter les nouveaux objets ou modifier les objets existants. Ce nombre doit être respecté lors de l'introduction des objets de l'environnement.
- Spécifier le type de contrôle (contrôler un seul robot, un sous-ensemble de robots hétérogène, tous les robots, etc.).
- Sélectionner le(s) robot(s) à contrôler dans le cas de contrôle d'un seul robot ou d'un sous-ensemble de robots.
- Enfin, remplir le formulaire.

3.2. Lancement de la plateforme JADE avec les agents

Nous avons suivi les étapes décrites ci-dessus pour introduire cinq robots hétérogènes (trois nouveaux robots et deux robots déjà existants) dans un scénario. Les nouveaux robots sont donnés comme suite (Tableau 5) :

Désignation du robot	Type du robot	Capacité de saisie/transport	Tag	Lecteur	Vitesse de déplacement	Nombre d'antennes
Manipulateur2	robot	1800g	TRFID5555	LRFID5555	–	08

	manipulateur					
Manipulateur3	robot manipulateur	2400g	TRFID5556	LRFID5556	–	04
Mobile5	robot mobile	1100g	TRFID5558	LRFID5558	02 m/s	07

Tableau 05 : Les nouveaux robots consistés dans le scénario.

Les anciens robots sont donnés comme suite (Tableau 6) :

Désignation du robot	**Type du robot**	**Capacité de saisie/transport**	**Tag**	**Lecteur**	**Vitesse de déplacement**	**Nombre d'antennes**
Manipulateur1	robot manipulateur	1155g	TRFID999	LRFID999	–	08
Mobile2	robot mobile	1789g	TRFID011	LRFID011	05 m/s	06

Tableau 06 : Les anciens robots consistés dans le scénario.

Pour lancer l'environnement JADE ainsi que les agents de contrôle, il faut lancer les classes suivantes : MainContainer, ContainerRobot et ContainerUser. Plus d'informations sont données sur la figure 56.

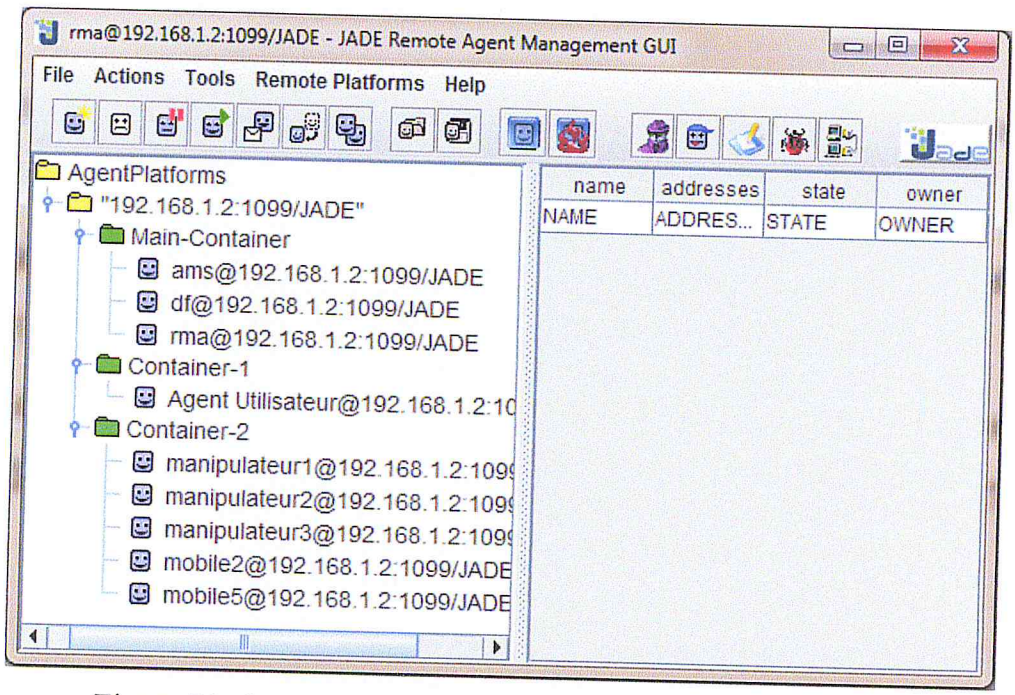


Figure 56 : Lancement de la plateforme JADE avec les agents.

4. Scénarios de validation

Pour la validation de la solution proposée, nous avons choisi quatre différents scénarios. L'objectif des trois premiers scénarios est de contrôler (allouer et ordonnancer les opérations) manuellement le système robotique. Quant au dernier scénario, sons objectif principal est d'automatiser la répartition des opérations sur les robots ainsi que l'ordonnancement de ces opérations sur chaque robot du SRCP tout en respectant les contraintes de précédence entre les opérations et en optimisant le temps total d'exécution.

4.1. Scénario 1 :contrôler un seul robot

Le premier scénario concerne le contrôle d'un seul robot dans le système cyber-physique.

4.1.1.Tâche acceptée

Nous considérons une tâche de « Pick-and-place » d'un objet par un robot manipulateur. L'objet doit se trouver dans son espace de travail accessible. Après la saisie, le robot doit déposer cet objet à une position cible donnée par les coordonnées (155, 333, 2) et qui doit se trouver toujours dans son espace de travail :

- Type de contrôle : un seul robot.
- Robot sélectionné : manipulateur1.
- Formulaire (à droite de la figure 57).
- Tâche à exécuter : « Pick-and-place » l'objet (OB004) par un seul robot (manipulateur1).

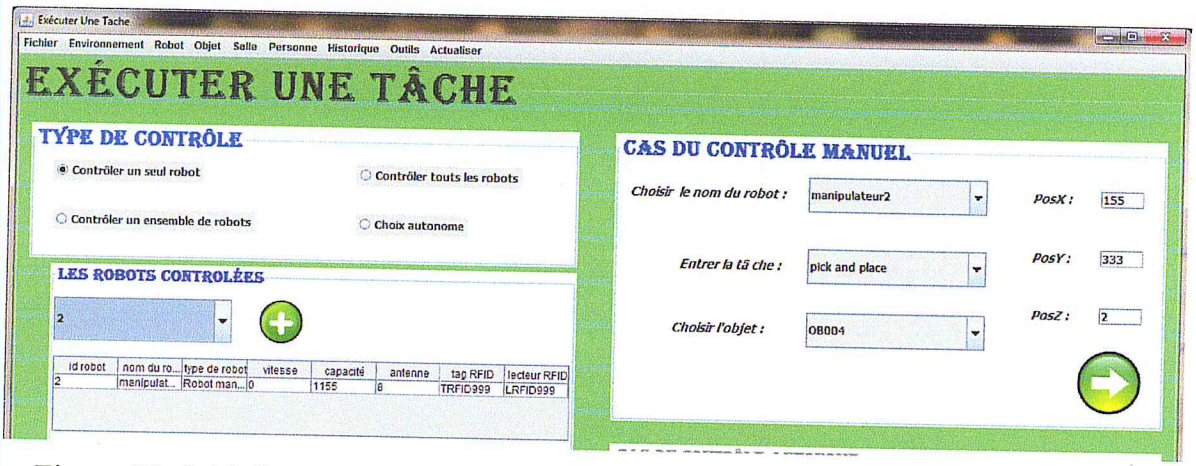
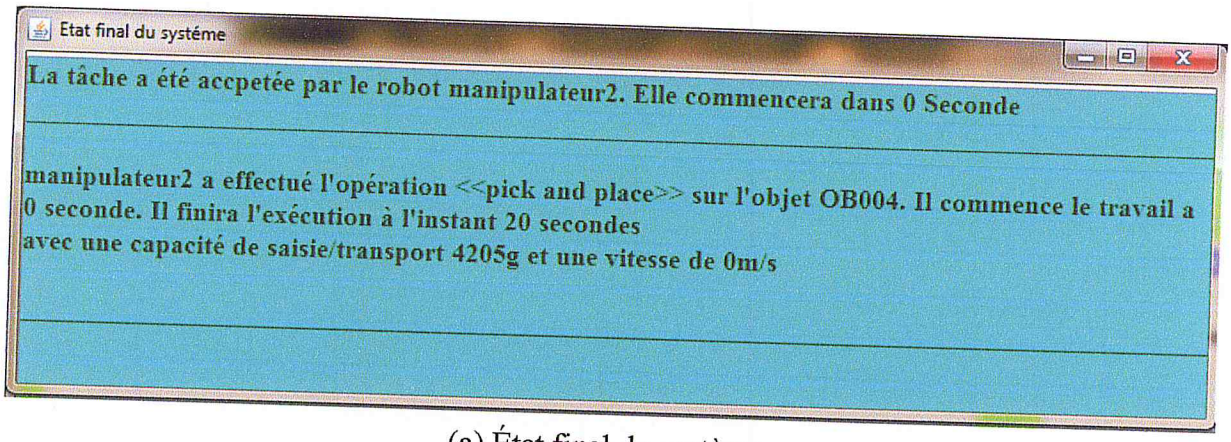
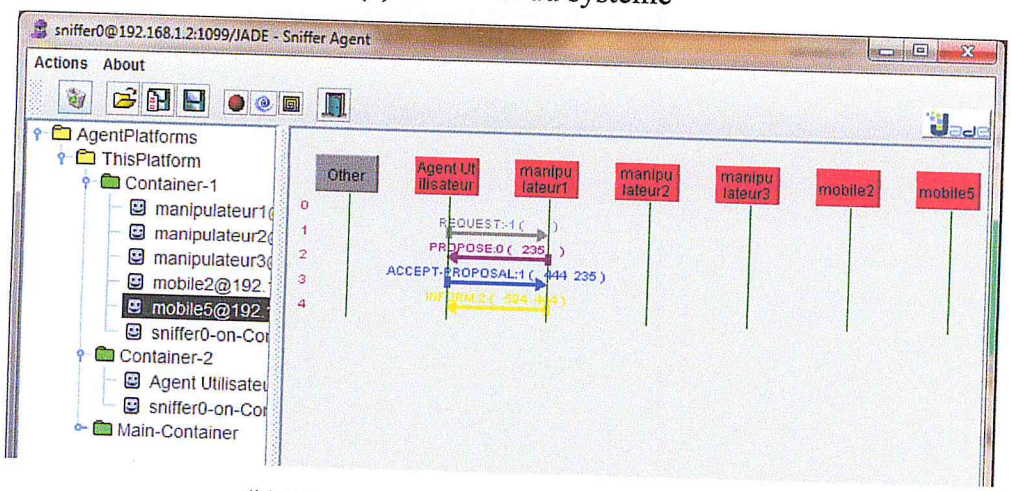


Figure 57 : Initialisation de la tâche « Pick-and-place » dans le cas de « contrôler un seul robot ».

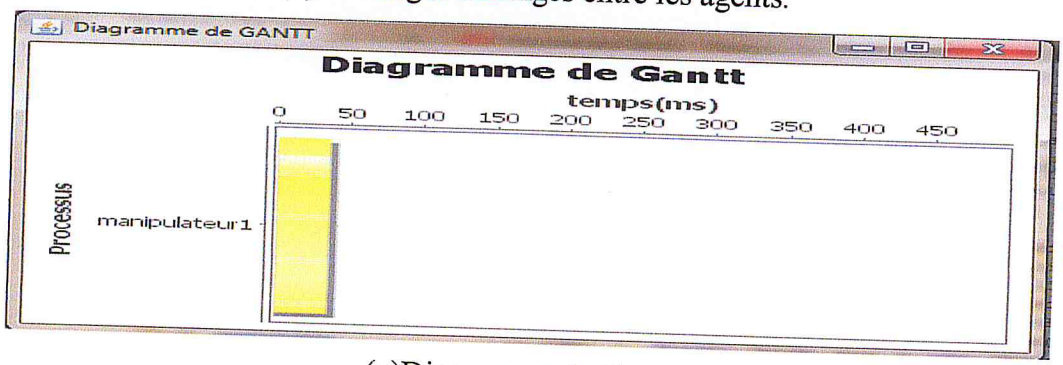
Après validation, le résultat obtenu est donné par les figures ci-dessous. La figure 58a donne l'état final du système ; les messages échangés entre l'agent utilisateur et l'agent manipulateur1 sont présentés dans la figure 58b ; le diagramme de Gantt de l'ordonnancement des opérations est donné par la figure 58c :



(a) État final du système



(b) Messages échangés entre les agents.

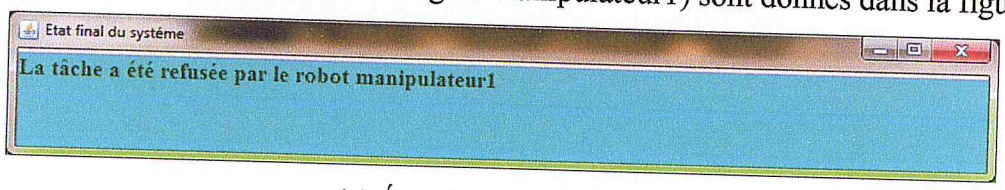


(c)Diagramme de Gantt.

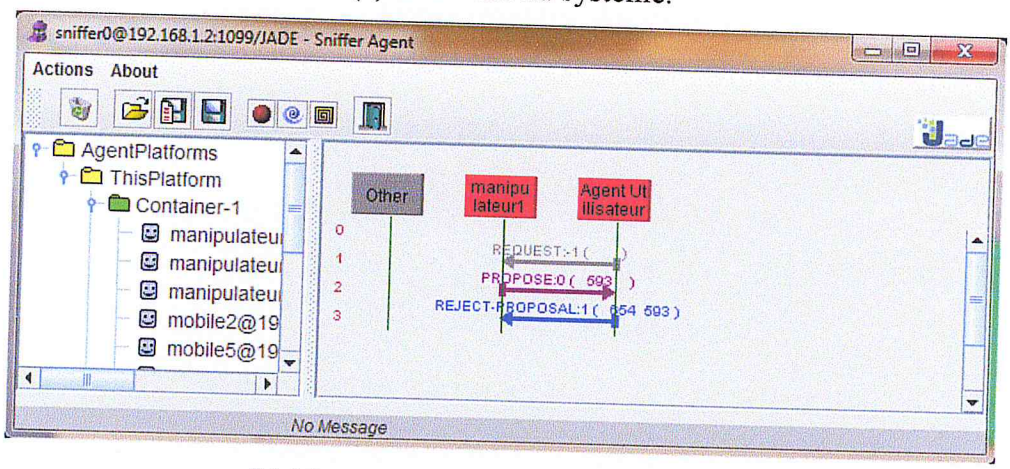
Figure 58 : Résultats obtenus pour la tâche « Pick-and-place » acceptée et exécutée par un seul robot

4.1.2. Tâche rejetée

L'initialisation de la tâche se fait comme pour le cas précédent ; sauf que cette fois-ci le robot doit effectuer la tâche « Déplacer-objet » au lieu de « Pick-and-place ». L'état du système est présenté dans la figure 59a ; les différents messages échangés entre les agents du système de contrôle (agent utilisateur et agent manipulateur1) sont donnés dans la figure 59b.



(a) État final du système.



(b) Messages échangés entre les agents

Figure 59 : Résultats obtenus pour la tâche « Déplacer-objet » rejetée par un seul robot.

4.2. Scénario 2 : Contrôler un sous-ensemble de robots hétérogènes

Dans le deuxième scénario, il s'agit de contrôler un ensemble robots.

4.2.1.Tâche acceptée

Nous avons considéré la tâche « Transporter-objet » suivante : un robot manipulateur (manipulateur1) saisit un objet (OB003) et le dépose sur un robot mobile (mobile2). Ce dernier transporte l'objet jusqu'à une position intermédiaire. Là, un autre robot manipulateur (manipulateur2) saisit l'objet et le dépose à la position cible donnée par les coordonnées (117, 243, 2) :

- Type de contrôle : un sous-ensemble de robots.
- Les robots sélectionnés : manipulateur1, manipulateur2, mobile2.

- Formulaire (à droite de la figure 60).
- Tâche à exécuter : « Transporter-objet » l'objet (OB003) par un sous-ensemble de robots hétérogènes (manipulateur1, manipulateur2, mobile2).

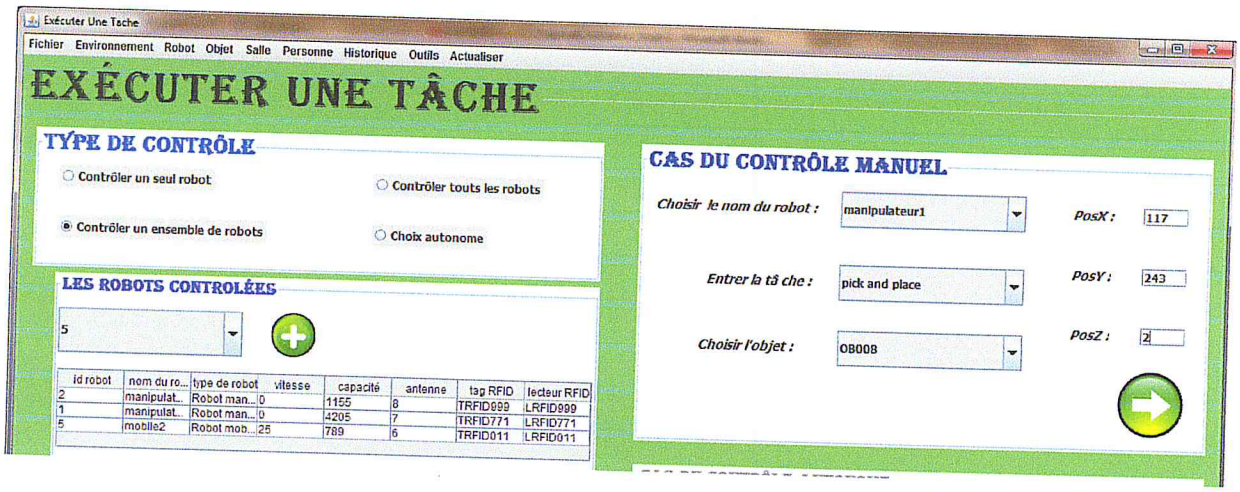
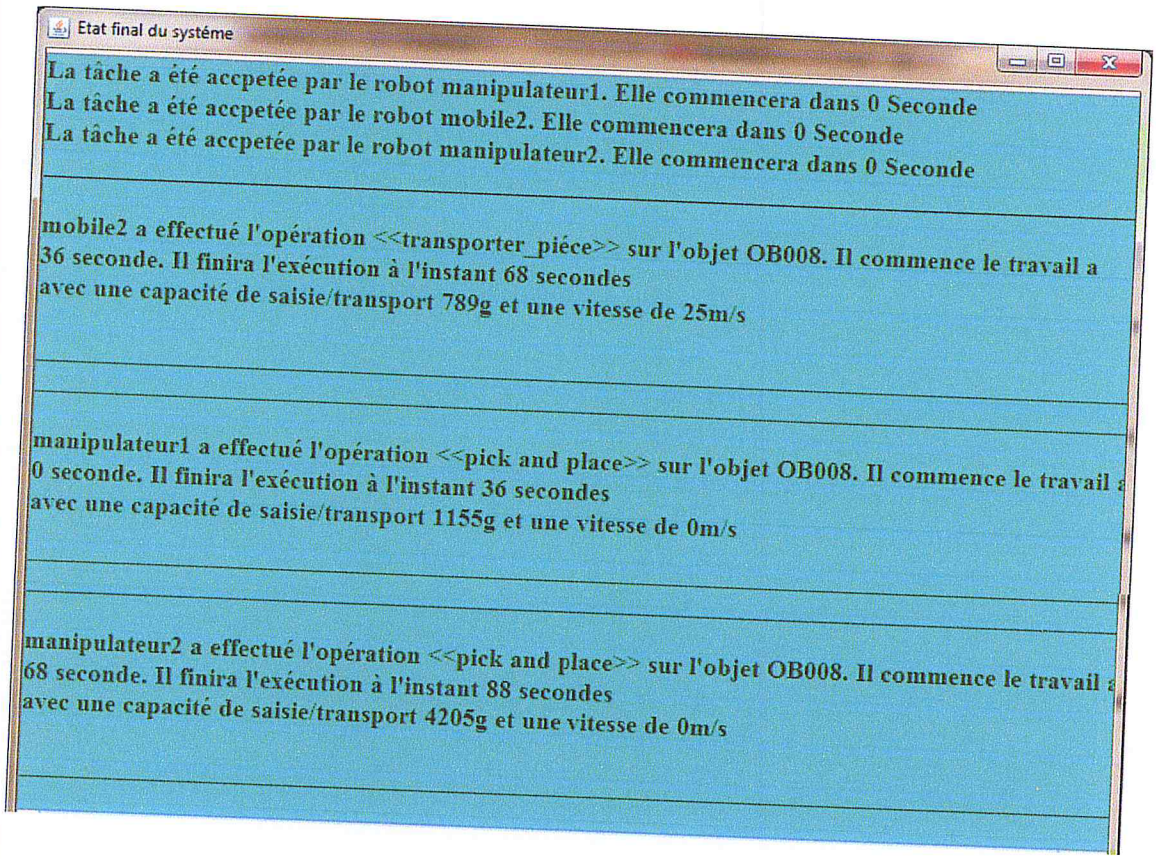
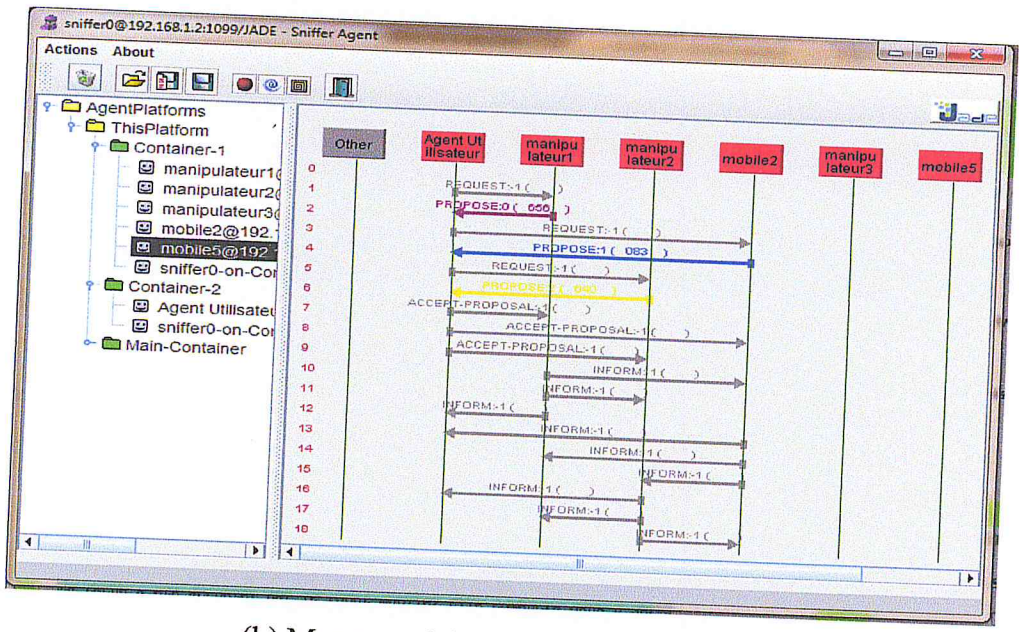


Figure 60 : Initialisation de la tâche « Transporter-objet »acceptée et exécutée par un sous-ensemble de robots hétérogènes.

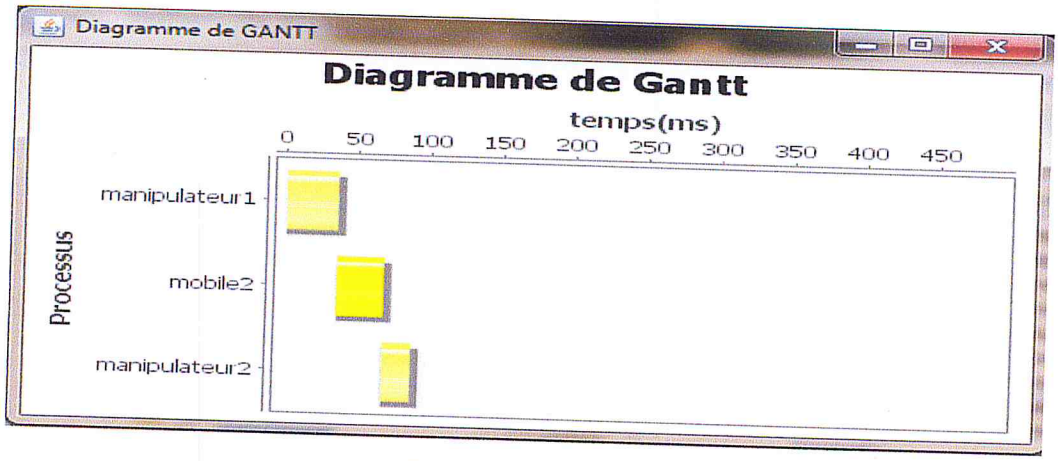
Après que l'utilisateur ait validé, les résultats obtenus. sont donnés dans la figure 61. L'état finale du système est donné par la figure 61a. Les messages échangés entre l'agent utilisateur et les agents manipulateur1, manipulateur2 et mobile2 sont présentés dans la figure 61b. Enfin, le diagramme de Gantt de l'exécution des opérations est donné par la figure 61c.



(a) État final du système.



(b) Messages échangés entre les agents.

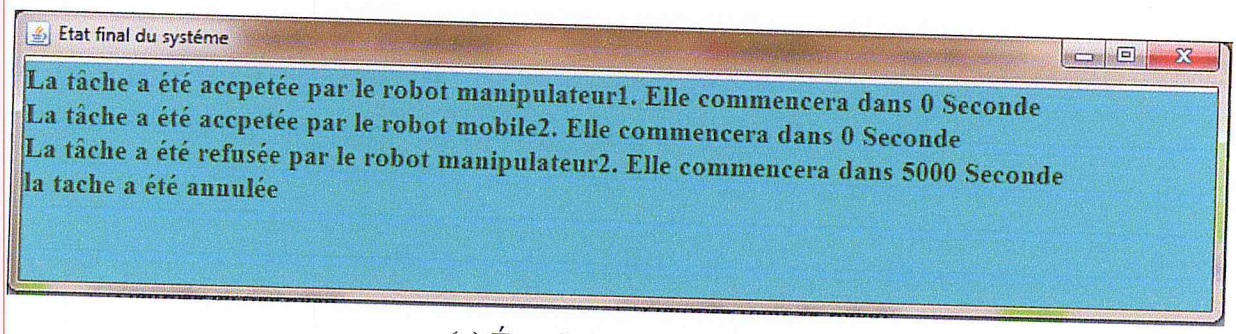


(c) Diagramme de Gantt

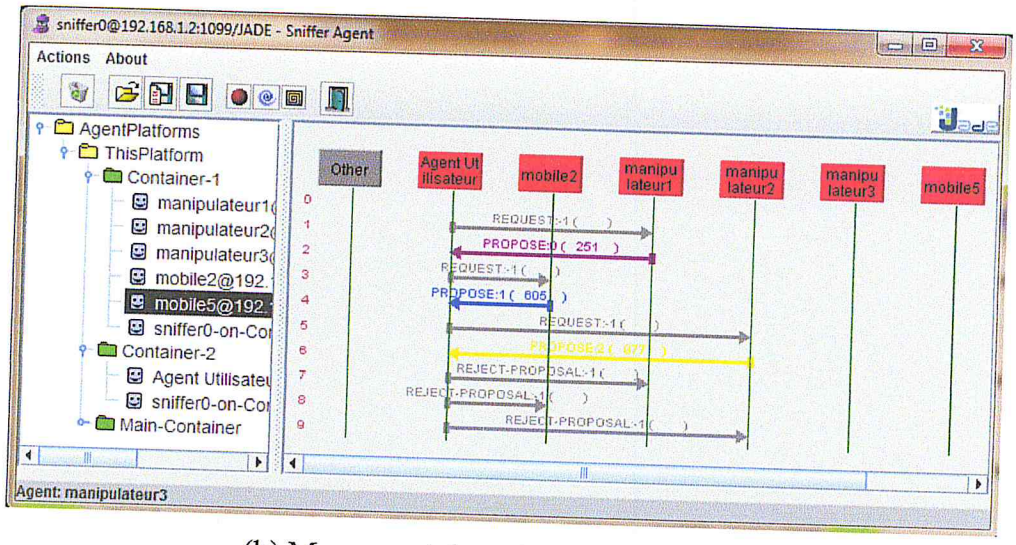
Figure 61 : Résultats obtenus pour la tâche « Transporter-objet » acceptée et exécutée par un sous-ensemble de robots hétérogènes

4.2.2. Tâche rejetée

Dans ce cas, la tâche « Transporter-objet » a été rejeté par un ou plusieurs robots (tâche rejetée par le robot manipulateur2), l'état final du système est représenté dans la figure 62a. Les messages échangés entre l'agent utilisateur et les agents robots sélectionnés sont présentés dans la figure 62b.



(a) État final du système.



(b) Messages échangés entre les agents.

Figure 62 : Résultats obtenus pour la tâche « Transporter-objet » rejetée par un ou plusieurs robots

4.3. Scénario 3 : Contrôler tous les robots

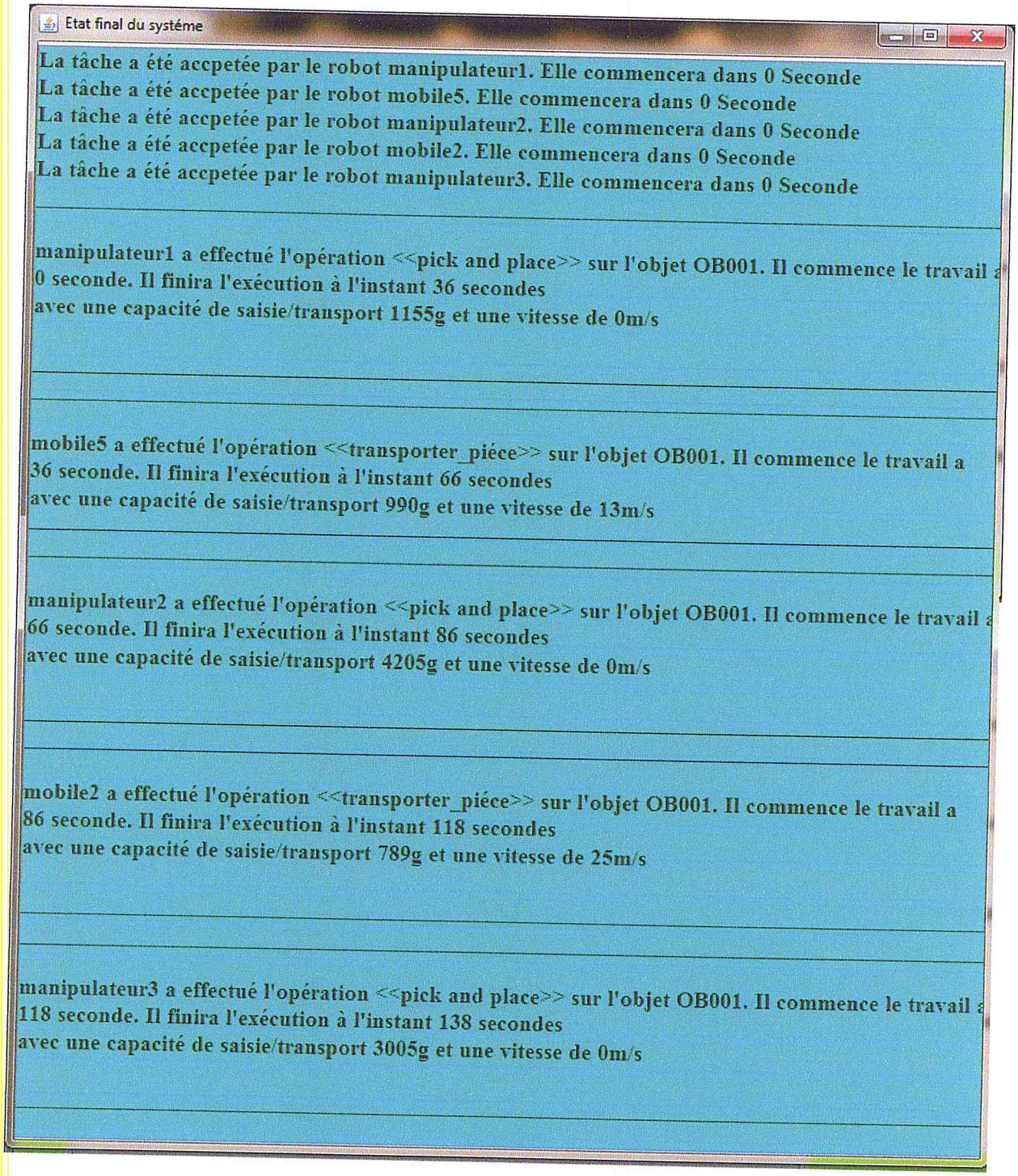
Le troisième scénario est plus complexe ;il concerne le contrôle de tous les robots existant dans le système.

Nous avons considéré la tâche « Transporter-objet » suivante : un robot manipulateur (manipulateur1) saisit un objet (OB003) et le dépose sur un robot mobile (mobile2). Ce dernier transporte l'objet jusqu'à une position intermédiaire. Là, un autre robot manipulateur (manipulateur2) saisit l'objet et le dépose sur un autre robot mobile (mobile5). Puis, ce dernier transport l'objet ou se trouve un autre robot manipulateur (manipulateur3) qui va saisir l'objet et le déposer à la position cible donnée par les coordonnées (117, 243, 2) :

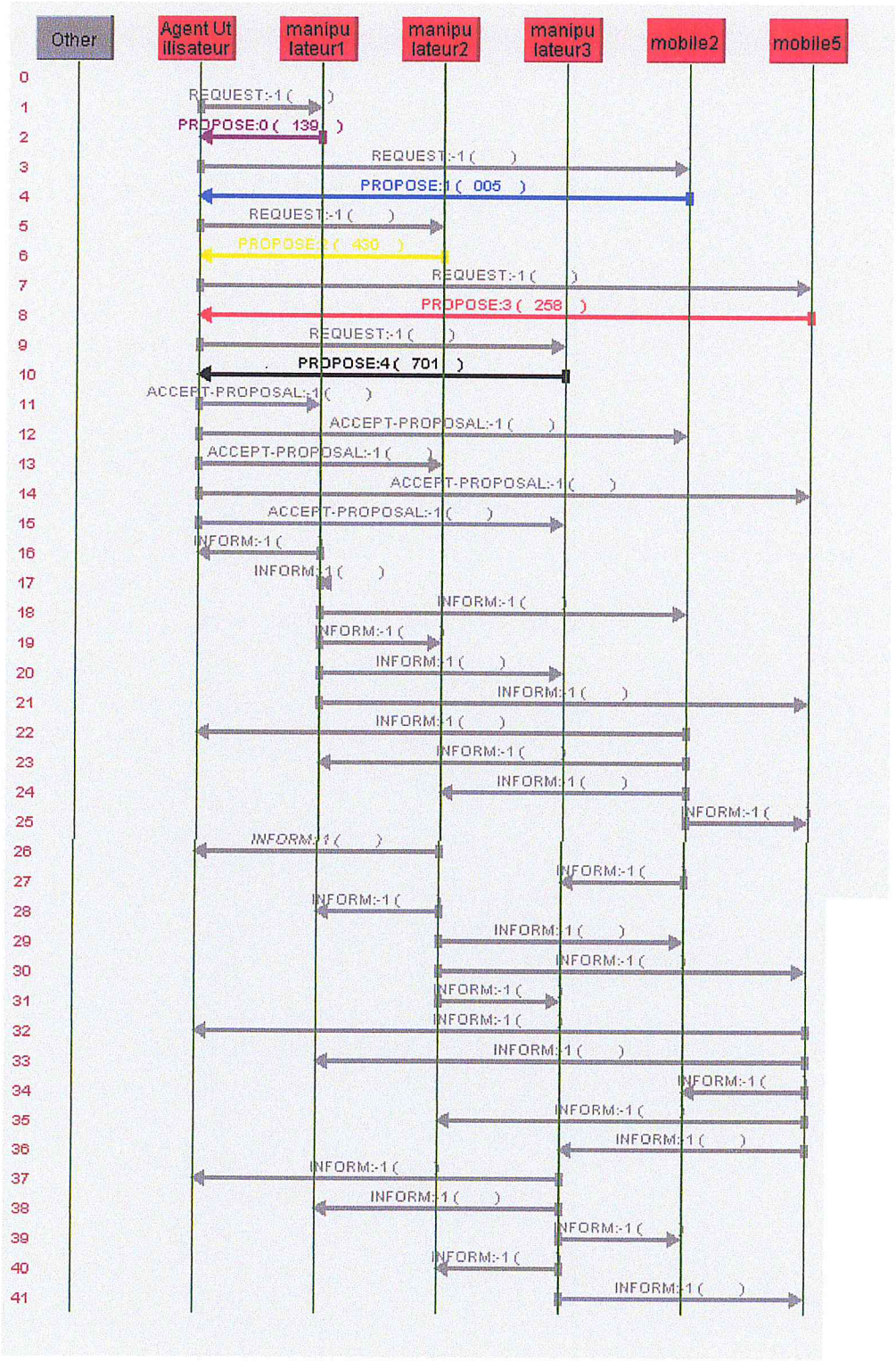
- Type de contrôle : un sous-ensemble de robots.
- Robots sélectionnés : manipulateur1, manipulateur2, mobile2, mobile5, manipulateur3.
- Tâche à exécuter : « Transporter-objet » l'objet (OB003) par un sous-ensemble de robots hétérogènes (manipulateur1, manipulateur2, mobile2, mobile5, manipulateur3).

Après que l'utilisateur ait validé, les résultats obtenus sont donnés dans la figure 63. L'état final est donné dans la figure 63a. Les messages échangés entre l'agent utilisateur et les agents manipulateur1, manipulateur2, mobile2, mobile5 et manipulateur3 sont présentés dans

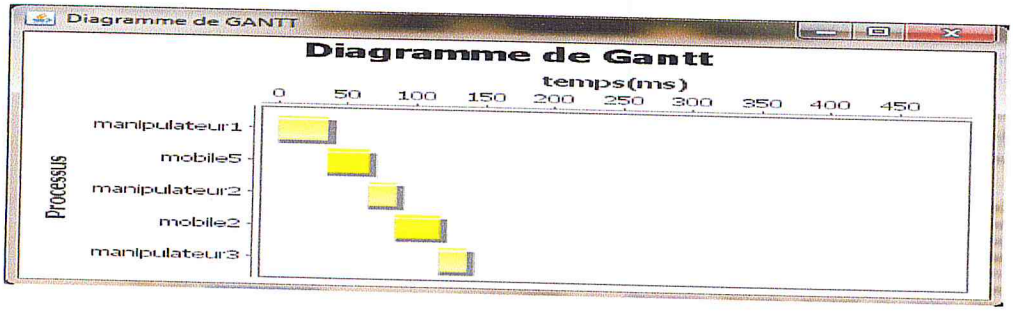
la figure 63b. Enfin, le diagramme de Gantt de l'exécution des opérations est donné par la figure 63c.



(a) Etat final du système.



(b) Messages échangés entre les agents.



(c) Diagramme de Gantt

Figure 63 : Résultats obtenus pour la tâche « Transporter-objet » acceptée et exécutée par tous les robots du système.

Dans le cas où la tâche est refusée, le traitement se fait de la même manière que dans le cas de contrôle d'un ensemble de robots.

4.2. Scénario 4 : Choix autonome

Dans le quatrième et dernier scénario, tous les robots ou un sous-ensemble de robots sont considérés. La différence réside dans les processus d'allocation et d'ordonnancement des différentes opérations de la tâche. Cette fois-ci, ils sont réalisés de façon autonome. En effet, les agents robots négocient, en utilisant le protocole décrit dans les chapitres précédents (basé sur les règles de priorité et les algorithmes génétiques), pour répartir les opérations sur les robots et ordonnancer le plan local de chaque agent robot. Ceci est réalisé dans le respect de contraintes de précédence et en minimisant le temps total d'exécution de la tâche.

5. Conclusion

Tout au long de ce dernier chapitre, divers scénarios de validation ont été réalisé. En examinant les résultats obtenus, nous pouvons affirmer de la validation de la solution que nous avons proposée pour le contrôle d'un système robotique cyber-physique.

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 et capables de coordonner leurs actions. En parallèle, une évolution similaire dans le domaine des systèmes temps-réel embarqués et répartis a justifié l'émergence du domaine des « systèmes robotiques cyber-physiques (SRCP) » reflétant une montée similaire en complexité.

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

L'objectif de notre travail est double. Le premier volet consiste en le développement d'une « Interface Homme/Robot (IHR) » conviviale et destinée au contrôle collaboratif multi-robots dans un SRCP. Chaque robot est considéré comme étant un agent autonome et coopérant dans le système. Cette interface homme/robot a comme rôles de :

- Transmettre les intentions de l'opérateur humain aux robots.
- Transmettre à l'opérateur les informations issues des différents capteurs équipant le système multi-robots hétérogènes.
- Transmettre à l'opérateur les informations issues des différents capteurs/tags implantés dans l'environnement cyber-physique où évoluent ces robots.
- Transmettre à l'opérateur les feedbacks sur l'exécution des différentes opérations et tâches confiées au SRCP.

Nous avons entamé ce travail par la présentation des SCP et des SRCP. Ces derniers consistent en un ensemble de robots autonomes fonctionnant dans un environnement communicant intelligent.

La solution que nous avons proposée utilise la technologie des systèmes multi-agents (SMA) dans le contexte robotique cyber physique.

Agent utilisateur : permet de prendre la décision sur l'acceptation ou le refus de l'exécution des tâches confiées au SRCP. Aussi, il transmet les consignes (les opérations) aux agents robots et affiches les résultats obtenus à l'opérateur après planification et exécution.

Agent robot : son rôle et de prendre des décisions (allocation et ordonnancement) en collaboration avec les autres agents robots du système. Cette décision collaborative respecte les contraintes imposées (précédence, temps total d'exécution, …) pour, par la suite, exécuter des opérations et des tâches par le SRPC.

Concernant le deuxième volet de ce travail, il concerne le processus de prise de décisions de façon autonome : allocation des opérations aux différents robots du SRCP et ordonnancement de ces opérations sur le même robot. Ici, nous avons considéré une approche basée sur les règles de priorité combinée aux algorithmes génétiques. L'objectif étant de minimiser le temps total d'exécution tout en respectant les contraintes de précédence entre les opérations. .

Cette approche a été implémentée en utilisant la plateforme JADE.

Les performances de la solution proposée ont été évaluées via plusieurs scénarios de validation :

- Contrôler un seul robot.
- Contrôler un ensemble de robots.
- Contrôler tous les robots du système.

Notre validation a donné de bons résultats. Les intentions de l'opérateur humain ont été transmises aux robots et les différents résultats affichés avec les messages échangés entre les agents et le diagramme de Gantt montrent le séquençage des opérations.

Nous espérons que la solution proposée pour contrôler l'équipe de robots hétérogènes gagnerait à susciter l'intérêt chez d'autres chercheurs pour des développements futurs dans le domaine de la robotique cyber-physique.

Les perspectives pour d'éventuels travaux futurs concernent essentiellement l'implémentation et la validation expérimentale de cette approche dans un environnement cyber-physique réel via d'autres types de tâches. En effet, il faut considérer des robots réels et des entités étiquetés (RFID ou autres) afin de tester l'efficacité et la robustesse de l'approche. Enfin, il faut considérer un grand nombre de robots hétérogènes dans un environnement complexe (plusieurs objets/obstacles).

Annexe 01

Algorithmes génétiques (AG)

Annexe 01

Algorithmes génétiques (AG)

1. Introduction

Les algorithmes génétiques sont constitués par une catégorie de programmes dont l'objectif est de résoudre un problème en reproduisant les mécanismes de la sélection naturelle. Ces algorithmes sont particulièrement adaptés à l'optimisation de problèmes combinatoires et notamment des problèmes dits NP-complets (dont le temps de calcul croit de façon non polynomiale avec la complexité du problème). Ces algorithmes constituent parfois une alternative intéressante aux réseaux de neurones mais sont le plus souvent complémentaires. Un exemple peut être donné dans le cas d'un système à N variables (dans le cas où N est grand) et où seulement certaines combinaisons de ces variables sont pertinentes dans quelques cas particuliers [44]. Un des exemples les plus intéressants dans ce cas concerne le problème du voyageur de commerce (« Traveling SalesmanProblem ») qui est très bien connu dans la communauté des chercheurs. L'énoncé est le suivant : un voyageur de commerce doit effectuer un déplacement dans 20 villes, quel est l'itinéraire le plus court ?

Résoudre ce problème par un algorithme génétique pour 20 villes équivaut à utiliser une centrale nucléaire pour alimenter une seule maison, mais qu'en est-il lorsque vous avez 100, 10000 ou même 1 milliard de villes ou de points ? Un algorithme traditionnel aura généralement un temps de calculs dissuasif et c'est justement là qu'interviennent les algorithmes génétiques, pour aider à réduire ce temps de calcul [44].

2. Principe général

Les algorithmes génétiques, comme les réseaux de neurones, ont calqué leur schéma d'optimisation sur l'observation de la sélection naturelle, et plus précisément sur les gènes et les chromosomes. À la différence des programmes génétiques, introduits par Holland [45], les

algorithmes génétiques travaillent sur des chaînes de caractères de taille fixe. Ces chaînes de taille fixe définissent chacune un individu et une solution potentielle au problème à résoudre.

On définit au départ une population d'individus, chacun disposant d'une chaîne de caractères particulière (codant son chromosome) généralement définie de façon aléatoire au départ. Les individus vont, ensuite, être évalués sur la base d'une fonction Objectif, être sélectionnés, se reproduire et subir des mutations. C'est un processus itératif qui prend généralement fin lorsque la population n'évolue plus entre deux périodes [44].

Nous disposons en première période d'une population P(0) d'individus ayant des caractéristiques diverses (aléatoires au départ) et dont les chaînes de caractères représentent des solutions potentielles au problème considéré. Les caractéristiques des individus sont ainsi codées par des chaînes de caractères de taille fixe et les individus n'ont aucune connaissance d'un modèle éventuel. À chaque période, nous sélectionnons les meilleurs individus selon une fonction « Objectif » et certains individus mutent ou se reproduisent. Nous itérons le processus jusqu'à la condition de terminaison (stabilité des caractéristiques de la population sur deux périodes par exemple). « Évaluation » utilise la fonction d'évaluation qui dépend du problème et qu'il faudra minimiser ou maximiser [44].

Le principe général des AG est donné par la figure 1 ci-dessous :

```
t = 0;
Initialisation de P[t];
Evaluation de P[t];
Tant Que Population Non Identique
   t = t + 1;
   Selection de P[t] dans P[t-1];
   Recombinaison de P[t];
   Evaluation de P[t];
Fin Tant Que
```

Figure 1 : Principe général des AG

3. Sélection

Comme son nom l'indique, la sélection vise à sélectionner une sous population à partir d'une population parent. La méthode la plus courante est celle initiée par Holland lui-même en 1975 : la « roulette sélection » (wheel selection). C'est une méthode de sélection proportionnelle au niveau de fitness des individus. Le nombre de fois qu'un individu sera sélectionné est égal à son fitness divisé par la moyenne des fitness de la population totale

(plus exactement, la partie entière représente le nombre de fois qu'il sera sélectionné, et la partie flottante la probabilité qu'il aura d'être sélectionné à nouveau). Cette fonction est déterminante dans un algorithme génétique. Aussi, de nombreuses méthodes de sélection, bien plus complexes sont disponibles : le sigma scaling, la sélection à la Boltzman, la sélection par rang, la sélection par tournois, etc.

4. Crossover

Le crossover, qui symbolise la reproduction sexuée (toujours par métaphore du mécanisme de sélection naturelle), est une des étapes importantes des AG. C'est l'instrument majeur des innovations au sein de l'algorithme, c'est lui qui insuffle le changement. Il peut être effectué de plusieurs manières mais la plus courante croise les chaînes de caractères de deux individus parents pour former des chaînes de caractères enfants.

La figure ci-dessous (Figure 2) illustre un « single-point » crossover avec deux parents (A,B) et deux enfants (C,D) qui échangent une partie de leur chaîne.

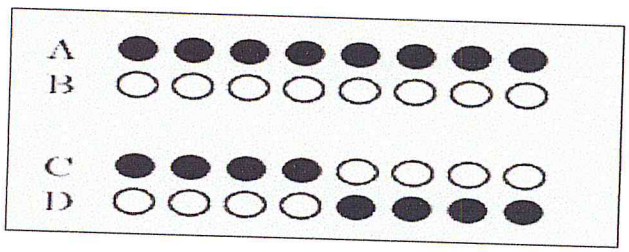


Figure 2 :Exemple de crossover.

Le taux de crossover est en général assez fort et se situe entre 70% et 95% de la population totale [44].

5. Mutation

Comme pour le crossover, la mutation vise à modifier de façon aléatoire une partie de la population. C'est le second mécanisme d'innovation d'un AG. Ici, le principe est de choisir une valeur de remplacement aléatoire pour l'un des gènes des individus de la population concernés. D'autres méthodes de mutation peuvent aussi être utilisées comme la mutation par soustraction numérique fixée sur un gène, ou remplacée par une valeur aléatoire choisie dans un sous-ensemble de valeurs (pour un cas réel par exemple).

À la différence du crossover le taux de mutation est généralement faible et se situe entre 0.5% et 1% de la population totale. Ce taux faible permet d'éviter une dispersion

aléatoire de la population et n'entraîner que quelques modifications sur un nombre limité d'individus. La mutation prend une place de plus en plus importante dans les AG, alors qu'il y a encore quelques années son rôle était encore considéré comme accessoire[44].

6. Évaluation

L'évaluation est la phase au sein de laquelle l'ensemble des individus devant être évalués (notamment ceux ayant subi une mutation ou un crossover) vont pouvoir quantifier leur degré d'élitisme. Le degré de fitness d'un individu sera calculé à l'aide de cette fonction d'évaluation qu'il faudra maximiser ou minimiser. L'opération se fait à l'aide d'une fonction fournie par l'auteur et qui dépend très étroitement du problème à résoudre via les AG[44].

Bibliographie

Bibliographie

[1] « Control Architectures of Robots 2015», *CAR'15*. [en ligne]. http://www.car-conference.fr/ [page consultée le 03 janvier 2016].

[2] E. A. Lee, "CPS Foundations", The 47th ACM Design Automation Confere (DAC'10), pp. 737-742, Anaheim, CA, USA, 13-18 June 2010.

[3] Krishna Venkatasubramanian, "Cyber-Physical Systems", [Partly adapted from slides by Prof. Insup Lee, Upenn], CS 525 C.

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

[5] C. Neuman, "Challenges in Security for Cyber-Physical Systems", DHS: S&T. Workshop on Future Directions in Cyber-physical Systems Security, Newark, NJ, USA, 22-24 July 2009.

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

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

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

[9] N. Wu, X. Li, "RFID Applications in Cyber-Physical System", Deploying RFID Challenges, Solutions, and Open Issues, Cristina Turcu (Ed.), InTech, pp. 291-302, 2011.

[10] http://www.journaldunet.com/solutions/systemes-reseaux/nfc/ **Dernière consultation : Avril 2016.**

[11] https://www.unitag.io/fr/nfc/what-is-nfc **Dernière consultation :Avril 2016.**

[12] C. Benavente-Peces, V. M Moracho-Oliva, A. Domínguez-García, M. Lugilde Rodríguez, "Global System for Location and Guidance of Disabled People: indoor and outdoor technologies integration", Fifth International Conference on Networking and

Services, 2009.

[13] B. Noubel El Houssine, "Le WIFI Technologies & Enjeu", TOPNETFournisseur DeService Internet, avril 2014.

[14] http ://www.gps.gov/french.php/ **Dernière consultation** : **Mars 2016.**

[15] http ://www.embedded360.com/industry-served/zigbee.htm/ **Dernière consultation : Mars 2016.**

[16] http ://www.centrenational-rfid.com/comment-fonctionne-le-nfc-article-133-fr-ruid-17.html/ **Dernière consultation : Avril 2016.**

[17] A. A. Mekonnen, F. Lerasle, A. Herbulot, A. Coustou, "Coopération entre un robot mobile et des caméras d'ambiance pour le suivi multi-personnes'', RFIA 2012 (Reconnaissance des Formes et Intelligence Artificielle), Lyon, France, Jan 2012.

[18] 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.

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

[20] 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.

[21] https://www.laas.fr/public/ **Dernière consultation : Mars 2016.**

[22] http://www.cnrs.fr/ **Dernière consultation : Mars 2016.**

[23] T. Deyle, M. S. Reynolds, C. C. Kemp, "Finding and Navigating to Household Objects with UHF RFID Tagsby Optimizing RF Signal Strength ", 2014.

[24] https://hal.archives-ouvertes.fr/hal-00761898/document **Dernière consultation : Mai 2016.**

[25] K.Desai, Y.Liu, et G.Liu,"A Graphical User Interface for Tele-operated Robotic Sample Acquisition",Aout 2012.

[26] A.Goodrich1 et Alan C.Schultz,"Human–Robot Interaction: A Survey",2007.

[27] M.E. "SMART PHONE BASED ROBOTIC CONTROL FOR SURVEILLANCE APPLICATIONS",Embedded Systems, Dept of ECE, Karpagam University, Coimbatore, Tamilnadu, India-641021

[28] https://openclassrooms.com/courses/debutez-l-analyse-logicielle-avec-uml/quelle-demarche-suivre **Dernière consultation : Mai 2016.**

[29] A.M.Uhrmacher et D.Weyns. "Multi-Agent Systems Simulation and Applications", CRCPress Taylor and Francis Group, 2009.

[30] J.Odell,H.V.D.Parunak et B.Bauer, "Extension de UML pour des système basés agents".

[31] http://jade.tilab.com/ **Dernière consultation : Mars 2016.**

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

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

[34] M.Wooldridge et N. R.Jennings, "Intelligent agents: Theory and practice", The Knowledge Engineering Review, 1995.

[35] Y.Demazea et J.Müller, "Decentalized Artificial Intelligence", the First European Workshop on Modelling Autnomous Agents in a Multi-Agent World-Cambridge, England 1990.

[36] Th.W.Maldone, "What is coordination theory In National Science Foundation Coordination TheoryWorkshop", The National Science Foundation Coordination TheoryWorkshop, 1988.

[37] http://www.auml.org/ **Dernière consultation : Décembre 2015.**

[38] J.Odel, H.V.D. Parunak et B.Bauer, "Extending UML for Agents", Article, 2001.

[39] http://cormas.cirad.fr/ **Dernière consultation : Décembre 2015.**

[40] http://aosgrp.com/products/jack/ **Dernière consultation : Décembre 2015.**

[41] E.Amouroux, C.Thanh-Quang, A.Boucher et A.Drogoul, "GAMA: an environment for implementing and running spatially explicit multi-agent simulations",,2009.

[42] http://www.madkit.net/madkit/ **Dernière consultation : Décembre 2015.**

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

[44] http://www.sylbarth.com/ag.php **Dernière consultation : Juin 2016.**

[45] J. H. Holland, Adaptation In Natural And Artificial Systems, University of Michigan Press (1975)

[46] A.Maoudj, B.Bouzouia2, A.Hentout, A.Kouider et R.Toumi,"Distributed multi-agent based approach for Products scheduling of multi-robot cells",CDTA.

[47] B.Thierno, "Le problème d'interopérabilité entre les plate-formes d'agents mobiles", Erricson, Canada, CAT 2000.

[48] M.Côté et N.Troudi, "NetSA : Une architecture multiagent pour la recherche sur Internet", Université Laval Département d'informatique Pavillon Pouliot Ste-Foy, G1K 7P4, Canada.

[49] G. Picard, "Méthodologie de développement de systèmes multi-agents adaptatifs et conception de logiciels à fonctionnalité émergente", Thèse de Doctorat, Université Paul Sabatier de Toulouse III, 2004.

[50] B. Chaib-draa, I. Jarras et B. Moulin, "Systèmes Multi-agents : Principes généraux et Applications", Article qui apparaît dans : J. P. Briot et Y. Demazeau « Agent et systèmes multiagents » chez Hermès en 2001.

[51] L.Overgaard, H. G. Petersen, J.Perram., Motion Planning for an Articulated Robot : a Multi-Agent Approach, in Distributed Software Agents and Applications (Maamaw'94), vol. LNAI 1069, J.-P. Müller, and J. Perram (Eds). Springer Verlag, 1994.

[52] C.Sohier, P.Bourdet "Real Time Scheduling of a Production Cell Based on a Multi-Agent System", Int. Conf. on Indust. Engin. and Product. Management. Mons, 1993 .