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

Université Saad Dahlab, Blida
USDB.

Faculté des sciences.
Département informatique.

Mémoire pour l'obtention
D'un diplôme d'ingénieur d'état en informatique.
Option : IA

Thème :

CONCEPTION ET IMPLEMENTATION D'UN SIMULATEUR GRAPHIQUE D'UN ROBOT MANIPULATEUR MOBILE

Présenté par : ABBASSI Mourad
HASNAOUI Younes

Promoteur: BOUKHELEF.DJ

Encadreur : A. HENTOUT
Z. TOUKAL

Organisme d'accueil : CDTA.

Soutenu le: 22/07/2006 **devant le jury composé de :**

AOUSSAT FADILA, M.A, BLIDA — **Président**

MAZARI REDHA, P.G, BLIDA — **Examinateur**

- Numéro/2005/2006-

DEDICACES

C'est avec un grand plaisir que je dédie ce modeste travail,
fruit de mes études en exprimant ma profonde
gratitude à tous mes proches particulièrement :

A mes précieux parents pour leur amour,
affection et compréhension.
Mon père pour sa patience, ses encouragements
et ses sacrifices.
Ma mère pour l'éducation qu'elle m'a inculquée
et toutes les peines qu'elle s'est donnée pour ma réussite.
Mes chers frères sans oublier mes très chères sœurs
A toute ma famille : mes grandes mères,
mes tantes et leurs époux .mes oncles et leur épouse ,
En particulier Mon Oncle Djamel
qui a cause lui Je suis la,

A la mémoire de mon oncle Brahim et ma tante Fatima,
que le Dieu l'accueille dans son vaste paradis,

A qui change ma vie, A qui sera ma femme un jour

A tous mes amis
A Mon binôme Younes ainsi que toute sa famille
A touts mes enseignants du primaires jusqu'aujourd'Hui,
A tous ceux et celles dont les noms n'ont pu être cités.

Mourad

الحمد لله

DEDICACES

Je dédie ce travail

À mes très chers parents

À mes frères et sœurs

À mes amis

Younes

REMERCIEMENTS

REMERCIEMENTS

Nous exprimons notre plus grande reconnaissance et nos vifs remerciements à notre promoteur **M HENTOUT Abdelfetah** et **M BOUKHELEF DJ** pour le temps et l'attention qu'ils ont bien voulu consacrer au bon déroulement de ce travail.

Nous remercions très vivement toute l'équipe Système robotisé de production de la division productique et robotique du centre de développement des technologies avancées pour les moyens qu'ils ont mis à notre disposition, tous les conseils et les encouragements qu'ils nous ont prodigués tout au long de ce projet.

Nos gratitudes les plus sincères à tous ceux qui ont voulu juger ce travail.

Nous exprimons notre profonde reconnaissance à Mme AOUSSAT Fadila et M MAZARI Rédha ainsi qu'aux membres du département informatique, pour leurs conseilles, leurs suggestions et pour l'aide qu'ils nous en fourni.

De même, nous adressons un grand merci au chef du département Informatique.

Enfin, nous remercions tous les enseignants qui ont contribué à notre formation ainsi que tous ceux qui ont participé de près ou de loin à la réalisation de ce travail.

SOMMAIRE

SOMMAIRE

INTRODUCTION GENERALE

INTRODUCTION GENERALE

La tendance est allée clairement vers les systèmes complexes qui sont utilisé pour la résolution de façon intelligente des problèmes liés à la production industrielle et robotique.

Donc les besoins de réaliser les systèmes complexes sont devenus évidents. Ils sont nécessaires pour exécuter des tâches dans des région ou la présence de l'être humain est difficile (site nucléaire, la lune...).

Le principal problème est souvent la réalisation d'une architecture du contrôle intelligente pour les systèmes complexes.
Jusqu'à maintenant, le contrôle et la structure de tels systèmes été conçu comme habituellement une hiérarchie centralisée, décentralisée ou bien distribuée.

La complexité de cette architecture nécessite un temps plus élevé pour une réaction rapide et une qualité considérable d'une action choisie. Pour la simplicité de système conçus la solution est de divisé les système en sous système appelé agent (qu'on va expliquer par la suite); pour cela il existe trois architectures qui sont ***l'architecture centralisé, décentralisé et distribué.***

Ce qui nous intéresse plus c'est le système distribué donc on va expliquer cette approche

Système distribué:

L'idée de la distribution de la connaissance a été à l'origine de la conception d'un ensemble de modèles d'architectures pour les systèmes d'Intelligence Artificielle. L'organisation de la connaissance du système dans des entités autonomes à pour objectif de lui garantir des meilleures performances.

[Brooks, 1991] : « l'intelligence humaine peut se construire sans une représentation unique et centralisée du monde ». [MINSKY, 1986] : « l'esprit est une société d'agents plus simples qui coopèrent ou se font concurrence»

Cette architecture contient un ensemble d'agents spécialisés dont chacun est une entité autonome ayant toutes les compétences requises pour communiquer, coopérer, agir et réagir. Des méthodes de communication adaptées sont offertes à chaque agent Dynamiquement, en fonction des communications, des accointances.

Les agents communiquent alors directement entre-eux et ne passent pas par le superviseur qu'en cas d'erreur. [Modèle MAGIQUE].

L'avantage de la distribution est inspiré en analysant les entités autonomes qui construisent le system robotique manipulateur cette analyse englobe:

La modularité: en possédant une structure prédéfinie pour le processus d'échange d'information on peut tester chaque partie du système (exemple le module de navigation pour la base mobile).

Base de connaissances décentralisée: le système total autorise chaque sous système d'utiliser une base de connaissances locale qui contient des informations pertinentes capable d'échanger l'information avec d'autres entités.

Forte tolérance et redondance: En cas d'un brisé de sous système ou une situation d'erreur signalée la redondance des systèmes inséparable -si elle existe- devrait être utilisable sans aucune erreur.

Dans le premier chapitre, nous allons présenter des notions et des définitions pour la simulation dans le domaine de la robotique distribué. Cette approche nécessite un système multi agent distribué parce que les taches sont distribuées. Pour cela nous allons expliquer superficiellement les différentes architectures de contrôle/commande et la notion SMA.

Dans le deuxième chapitre, nous allons présenter un peu de détail sur les simulateurs des robots qui existent.

L'objectif visé dans le troisième chapitre est de proposer une architecture multi-agents pour le contrôle d'un système robotique constitué d'une plateforme mobile surmontée d'un bras manipulateur (robot manipulateur mobile). Cette architecture doit être basée sur un modèle générique (qui est indépendant du type de robot utilisé et des différentes ressources existantes ou les ressources opérationnelles) pour permettre l'ajout de nouvelles fonctionnalités et de nouvelles ressources.

Dans le quatrième chapitre nous allons présenter l'architecture détaillée de contrôle/commande du robot.

Au cinquième chapitre nous présenterons quelques tests en vue de valider la modélisation proposée.

Nous terminerons par une conclusion générale et quelques perspectives pour notre travail.

CHAPITRE I

ETAT DE L'ART DES ARCHITECTURES DE CONTROLE/COMMANDE DES ROBOTS

CHAPITRE I

ETAT DE L'ART DES ARCHITECTURES DE CONTROLE/COMMANDE DES ROBOTS

I.1 Introduction:

Dans ce chapitre nous allons présenter des notions et des définitions pour la simulation dans le domaine de la robotique distribué cette approche nécessite un système multi agent distribué parce que les taches sont distribuées. Pour cela nous allons explique superficiellement les différent architecture de contrôle/commande et la notion SMA.

I.2 Les architectures de contrôle:

L'architecture de contrôle ou de commande est un concept qui donne une certaine intelligence à un système conçu, dans les robots mobiles ce concept devenu une méthode très importante a cause de l'accroissement immense de l'utilisation des robots dans diffèrent domaine.

Il existe plusieurs approches pour la conception des systèmes complexe depuis le début de la robotique pour offrir un seul objectif c'est bien muni le robot une intelligence pour puisse le robot comporte selon les exigence de certain situation plus ou moins complexe ; citons « l'environnement, la mission à accomplir .... ».

Parmi les architecture qui existe on peut citer trois méthodes qui sont importantes *une architecture délibérative, une architecture réactive, une architecture hybride.*

I.2 Architecture générale d'un robot

De manière générale un système robotique est composé de trois couches hiérarchiques Comme le montre la figure 1.1, la *couche logicielle* la couche *Robot Virtuel* et la *structure matérielle* du robot.

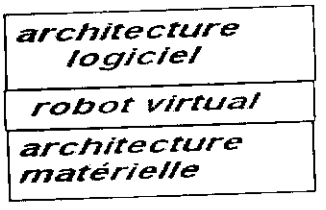


Figure 1.1 : Architecture générale d'un robot

La couche logicielle: ou on trouve une *interface utilisateur*, *le planificateur, un module d'exécution d'actions* et un *module fonctionnel.*

L'interface utilisateur : possède comme rôle l'interaction via une visualisation graphique avec l'utilisateur pour afficher le robot et leur environnement.

Le planificateur: si une mission est établie a partir de l'interface utilisateur le planificateur fait l'interprétation de mission en plusieurs taches et génère un plans d'action pour puisse exécuter les taches par le module d'exécution.

Module d'exécution:

Ce module est responsable au contrôle d'exécution de plan d'actions fournit par le planificateur.

Module fonctionnel : est constitué par un ensemble de modules fonctionnels qui coopèrent entre eux pour réaliser les tâches fournies par l'*Exécutif*. Chaque module est capable d'exécuter des services spécifiques (appelés *fonctionnalités*), au moyen d'actions sur le robot virtuel et/ou sur les autres modules.

Au dessus du niveau fonctionnel, on retrouve le « *Niveau Exécutif* » qui est chargée de l'exécution des *plans d'actions* fournis par le *Planificateur*, et qui doit:

- Vérifier l'évolution de l'état du robot et de son environnement;
- S'assurer de l'applicabilité des actions du plan;
- Déclencher l'exécution de ces actions, en activant ses fonctionnalités;
- Détecter d'éventuels échecs d'exécution par rapport au plan original et essayer de les corriger. [RRN05 HENTOUT]

I.3 Différentes architectures de contrôle/commande

I.3.1 Introduction:

Nous allons présenter quelques architectures basées sur la notion de système multi-agents pour cela l'étude est basée sur trois caractéristiques qui sont:

La réactivité, délibérative, hybride.

I.3.2 Architecture réactive

C'est une architecture basée sur la réaction du robot c.-à-d. donner au robot un certain taux de réflexion qui le permet de s'adapter aux différents changements de son entourage, cette réactivité permet donc au robot de réagir à un stimuli sans avoir à raisonner sur des variables d'état (l'état de robot) [MAE 93].

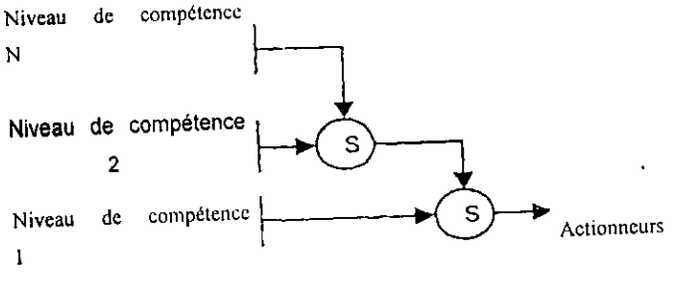
I.3.2.1 Architecture de BROOKS [86] :

Brooks à inspiré cette approche du comportement animal pour développer une architecture de contrôle; par exemple si tu est agressive devant un chien son comportement sera de baisse la tête et la queue ce comportement est compréhensible par tous les chiens sans avoir une signification à ce geste animal.

Cette architecture présente une approche de quatre modules perception, modélisation, planification, exécution.

Brooks conçu un system de comportement hiérarchique ou les actions sont classées sous un ordre de priorité bien défini donc on trouve des actions de haut niveau qui commande les actions de bas niveau. Des nouveaux comportement plus complexes sont ajoutés de telle sorte que les anciens englobent le system précédent (subsume).

Le schéma suivant résume l'architecture de Brooks :



I.3.2.2 L'architecture de MATARIC :

C'est une version de Brooks amélioré. Il a ajouté la notion d'apprentissage,

I.3.2.3 L'architecture de robot footballeur [KIM et AL] :

Cette architecture de contrôle distribuée basée sur un système multi-agents coopératifs et temps réel pour la conduite d'une équipe de robots footballeurs [KIM97].

Chaque robot possède une stratégie ou un comportement spécifique au rôle qu'il a au sein de l'équipe à un instant donné (gardien, attaquant, défenseur) [these.Z].

I.3.2.4 L'architecture de robot marcheur autonome :

Sur le même concept que Brooks ; l'équipe de MIME a développé un concept coopération entre les défirent robot pour puisse les robot résolvent le problème de manière collective.

Chaque robot possède un ensemble d'organe dont l'activation est commandée par un module central.

Des capteurs spécifiques pour déterminer toute réaction du robot via l'environnement

I.3.3 Architecture Délibérative

La délibération offre au robot mobile une autonomie lui permettant de planifier ses opérations à partir d'une mission décrite à un haut niveau d'abstraction. Plusieurs architectures de contrôle/commande de robot mobile ont été développées sur la base de l'architecture du robot. L'objectif de cette architecture était de doter le robot de capacités de raisonnement.

Dans ce qui suit, nous présentons quelques exemples d'architectures délibératives.

I.3.3.1 Architecture du robot SHAKEY

L'architecture de contrôle/commande du robot mobile SHAKEY [NİL 69] est basée sur une approche hiérarchique à deux niveaux, soit:

-Le niveau planificateur

-Le niveau de contrôle d'exécution

En résumé, la commande du robot SHAKEY est hiérarchisée, le contrôle est centralisé au niveau planificateur, le degré de raisonnement est limité et l'interfaçage entre les deux niveaux est complexe du fait de la grande différence des degrés d'abstractions des actions.

I.3.3.2 Architecture de Meystel

MEYSTEL a proposé une architecture hiérarchisée construite à partir du modèle de l'architecture du robot SHAKEY par décomposition de l'application en trois niveaux de base dans la planification et l'exécution des trajectoires d'un robot: le planificateur, le navigateur et le pilote [MEY 82][CHA 84]. Le planificateur établit des chemins en se basant sur des connaissances a priori, le navigateur établit des trajectoires plus précises à partir du plan fourni par le planificateur et le pilote exécute les fonctions de déplacement du robot mobile. En plus des caractéristiques du modèle SHAKEY, cette architecture présente l'avantage d'avoir un niveau supplémentaire d'abstraction sur le modèle de l'environnement et sur la tâche de planification: le niveau navigateur.

I.3.3.3 Architecture de Albus et Shneir

Albus et Shneir ont proposé une architecture composée de plusieurs niveaux hiérarchiques dont le nombre peut varier de six à huit [ALB 85]. Chacun de ces niveaux comporte trois modules concurrents dont le premier est dédié au traitement des informations de retour, le second à la modélisation du monde et le troisième à la décomposition d'un but en sous buts et par conséquent d'une tâche en sous-tâches.
Cette architecture de contrôle/commande pour robot mobile, quel que soit le nombre de niveaux hiérarchiques présente un inconvénient majeur qui est celui de la centralisation de la décision dans le module de plus haut niveau.

I.3.3.4 Architecture du robot CMU Rover

L'architecture de contrôle/commande du robot CMU Rover est basée sur un modèle de tableau noir parallèle [ELF 83]. Ce tableau noir contient des informations sur l'état du robot, des capteurs et de l'évolution de la mission. Le contrôle est distribué sur plusieurs modules où chacun d'eux gère un type particulier de sous

tâches comme la gestion des capteurs ou celle des effecteurs. Grâce au tableau noir, ces modules sont dotés d'un certain degré d'expertise qui donne au système une certaine flexibilité pour mettre en oeuvre différents types d'applications. Cette architecture présente l'avantage de comporter un module original de supervision de plan qui est chargé d'extraire des informations sur l'ordonnancement des sous tâches du plan de contrôle.

I.3.3.5 Architecture du robot NAVLAB

Une architecture de contrôle/commande CODGER (Communication Database with GEometric Reasoning) a été développée en vue de piloter [SHA 86]:

1. Le CMU-NAVLAB [GOT 87], véhicule automobile traditionnel qui intègre plusieurs capteurs, une caméra et un système informatique embarqué.
2. Le NAVLAB, robot mobile destiné à naviguer sur un réseau routier qui, reconnaît les marquages au sol et détecte les obstacles.

Le contrôle est distribué sur cinq modules qui se partagent les tâches de contrôle. Le système utilise aussi un tableau noir central qui contient la carte du secteur parcouru (Local Map Database). Les modules fonctionnent de façon asynchrone et se synchronisent par l'accès au tableau noir.

I.3.3.6 Architecture NASREM

Cette architecture est fortement hiérarchique et structurée sur. Un modèle en six niveaux de commande. Chaque niveau est structuré autour de trois modules concurrents similaires à l'approche proposée par Meystel: module de traitement du retour d'informations, module pour la modélisation de l'environnement et un module pour la décomposition en sous tâches. Pour une mission donnée, l'échange d'informations se fait horizontalement au même niveau d'abstraction et verticalement entre les modules de même nature pour atteindre les différents capteurs et actionneurs. Par ailleurs, un système de tableau noir qui gère des informations diverses telles que les variables d'état du robot, les listes d'objets, l'évaluation des fonctions, etc, offre à cette architecture un aspect cognitif. Cette architecture présente tous les avantages de l'approche hiérarchique d'un point vue délibératif ainsi que l'avantage de distribution grâce aux modules concurrents présents à chaque niveau d'abstraction. Cependant,

elle présente un inconvénient majeur qui est dû à sa faible réactivité étant donné son caractère centralisé au niveau commande.

I.3.3.7 Architecture du robot KAMRO

Une architecture de contrôle/commande a été développée à l'Université de Karlsruhe pour piloter le robot mobile KAMRO. Ce robot est muni de deux bras manipulateurs et d'une base mobile [RAC 89]. Cette architecture est dédiée à des applications d'assemblage où la base du robot se déplace entre les postes de travail pour la réalisation de tâches d'assemblage à l'aide des deux bras. Cette architecture comporte les niveaux hiérarchiques suivants :

- Trois niveaux de commande embarqués ;
- Trois niveaux de contrôle niveau tâche sur l'hôte (superviseur).

L'originalité du système réside dans le fait que chacun des niveaux est construit autour d'une structure de tableau noir. La structure contient une représentation du monde. Le niveau d'abstraction de la représentation va de la représentation d'objets 3D au niveau 6, à des coordonnées cartésiennes et articulaires au niveau 1. Chaque tableau noir possède une unité de contrôle indépendante.

I.3.4 Architectures hybrides

Dans les architectures hybrides, on distingue en général trois niveaux de réactivité :

-La réactivité réflexe qui met en œuvre des boucles directes capteurs actionneurs : Ces boucles comprennent les actions réflexes (exceptions) et les processus asservis (boucles sensori-motrices).

-La réactivité stratégique qui met en œuvre une planification basée sur de nouvelles hypothèses dues aux modifications récentes de l'environnement.

-La réactivité tactique, qui est intermédiaire entre les deux niveaux précédents. Il s'agit de processus décisionnels fermés qui agissent en fonction du contexte avec un temps de réponse borné.

I.3.4.1 Architecture du robot HILARE

Cette architecture est hiérarchisé et comporte plusieurs niveaux :

Niveau planificateur " pour la collaboration, la planification locale et la coordination".

Niveau contrôle pour l'exécution d'une mission, et comporte :

- une unité fonctionnelle.
- Un module capteur
- Servo-processus
- Un module effecteur.

En résumé, cette architecture présente une structure de contrôle à la fois distribuée et centralisée qui permet une réactivité tactique du fait qu'elle utilise un modèle local de représentation de l'environnement. Ce type de réactivité offre au robot la possibilité de réagir instantanément à toutes les situations possibles par des actions susceptibles de le rapprocher de son objectif. Il s'agit dans ce cas d'une réactivité plus élaborée que la réactivité utilisée dans l'approche de Brooks.

I.3.4.2 Architecture TCA

L'architecture TCA (Task Control Architecture)' [S1M 92] [SIM 94] est structurée autour de modules spécifiques concurrents qui communiquent par messages. La gestion des ressources, des tâches et du routage des communications est assurée par un contrôleur central appelé contrôleur de tâche. Ce contrôleur est chargé aussi de la coordination des modules concurrents (perception, planification, et exécution de plans).

Ce mécanisme de réponse aux événements confère au robot une certaine réactivité temps réel, lui permettant de réagir en un temps borné aux événements asynchrones de l'application. La réactivité est donc ici considérée comme la gestion des comportements exceptionnels par opposition aux comportements nominaux correspondants à l'exécution délibérée du plan.

En résumé, l'architecture TCA est délibérative relativement aux comportements nominaux et réactive à deux niveaux : tactique relativement aux comportements d'exception et stratégique (raisonnement au niveau tâche) en cas de replanification.

I.4 Systèmes multi-agents

I.4.1 Le degré d'intelligence d'un agent

Cela mène a parler au diffèrent type d'agent pour cela on distingue trois types d'agent:

I.4.1.1 Agent cognitif:

Un agent cognitif possède des connaissances et des notions logiques dites des *notions mentales*.

Les agents cognitifs possèdent une représentation partielle de l'environnement, des buts explicites, ils sont capables de planifier leur comportement, mémoriser leurs actions passées, communiquer par envoi de messages, négocier, etc. c-à-d une capacité de raisonnement. Un SMA constitué d'agents cognitifs possède communément peu d'agents.

I.4.1.2 Agent réactive:

Contrairement aux agents cognitif l'agent réactif possède une capacité de réaction sur l'environnement à l'aide des composants qui perçoivent l'environnement mais pas une représentation de l'environnement et des notions mentales ni de mécanisme de message il possède uniquement une capacité de rétroaction (si condition alors action) selon le développeur.

I.4.1.3 Agent hybride

C'est une combinaison de l'intelligence réactive est cognitive afin d'éliminer les limitation de ces deux dernières.

L'architecture hybride est un ensemble de modules organisés dans une hiérarchie qui constitue un agent, les modules sont soit des entités qui possèdent des caractéristiques cognitifs soit des modules réactifs.

I.5 Conclusion:

Comme conclusion nous allons choisir la meilleure approche parmi les trois cités précédemment c'est bien l'architecture hybride car elle est la plus qualifier bien qu'elle donne au robot une capacité de raisonnement qui permette de réalisé une tâche complexe par décomposition de ces dernières en sous taches.

Aussi peut être la combinaison des deux autres architecture fait éléminéer les limites qui possède les deux autres architectures.

Cependant la réactivité présente un aspect très important au niveau robustesse mais l'absence de la planification rendre l'approche inutile devant un environnement complexe; hors que l'IA se trouve dans les domaines complexes. A cet effet l'architecture délibérative apparaisse pour doter au robot une capacité de flexibilité et accomplisse des taches complexes qui nécessitent une planification.

L'inconvénient majeur c'est quelle ne possède pas une réactivité assez important malgré une certain capacité de perception d'environnement c-à-d ne peut pas suivre l'évolution d'environnement.

CHAPITRE II

ENVIRONNEMENTS DE SIMULATION DANS LE CONTEXTE ROBOTIQUE

CHAPITRE II

ENVIRONNEMENTS DE SIMULATION DANS LE CONTEXTE ROBOTIQUE

II.1 Introduction

Nous allons présenter dans ce qui suit un peu de détail sur les simulateurs des robots qui existent.

La simulation du robot est un des aspects principaux dans le champ de recherche de la robotique.

Elle offre l'avantage de pouvoir attaquer la résolution d'un problème avant même d'avoir construit le mécanisme qui fait l'objet de l'étude, et même avant de fixer tous les choix technologiques. Cela doit permettre, au moment de la réalisation concrète, d'obtenir rapidement un robot fonctionnel, et surtout de ne pas faire courir de risque grave à un matériel Coûteux.

Les programmes du robot peuvent être testés et analysé sur le modèle simulé avant d'être exécuté sur le robot réel lui-même.

II.2 Définition de la simulation :

La simulation est un science qui sert a visualiser le phénomène étudier sur ordinateur avant de le réaliser, dans notre cas c'est bien pour valider et visualiser les mouvement de robot au sein d'un environnement crée dans le but de vérifier les erreur et les anomalie.

II.3 Avantage de la simulation :

La simulation informatique offre cet avantage de pouvoir attaquer la résolution d'un problème avant même d'avoir construit le mécanisme qui fait l'objet de l'étude, et même avant de fixer tous les choix technologiques. Cela doit permettre, au moment de la réalisation concrète, d'obtenir rapidement un robot fonctionnel, et surtout de ne pas faire courir de risque grave à un matériel Coûteux.

II.4 Objectif de simulation :

Le but majorant de simulation c'est bien de raccourcit le temps et diminué le coût à vrai dire crée un monde virtuelle « réalité virtuelle » pour validé les défirent algorithme associer au robot c-à-d on va visualiser le robot sur ordinateur pour faire une analyse Pertinente afin de crée et implémenté des algorithmes dans le robot.

II.5 Différents types de simulation :

Dans ce cadre on peut distinguer deux grand types de simulation une simulation *commandé par ordinateur* et une *simulation relative aux actions de robot.*

II.5.1 Simulation commandée par ordinateur :

C'est un algorithme qui sera exécuter directement par ordinateur c.-à-d. il n'y a pas une interaction avec le robot.

II.5.2 Simulation par des mouvements émis par le robot :

C'est bien clair. Dés fois on trouve un environnement ou l'être humain ne peut pas intervenir C'est le cas ou le robot fait son travail d'où on simule les défirent action émis par le robot.

II.6 C'est quoi simulé :

Cette réalité virtuelle de manière générale s'agit de construire un monde virtuel qui contient les objets qui intéressent le concepteur ; qui vont se comporte comme la réalité. Faisons un peut d'analyse ; on réalité on peut jamais atteint une simulation parfaite mais par hypothèse le but est atteint car on commander et maîtriser le simulateur.

II.7 Le simulateur de robot PUMA 560:

Le simulateur graphique est réalisé en programmation (off-line).

D'après sa configuration standard, le robot PUMA peut être utilise pour effectuer des taches simples seulement. il est un manipulateur à six degré de liberté avec un organe terminal qui peut atteindre un point dans son espace de travail de toute direction.

Ce robot est très répandu dans l'industrie pour effectuer des taches d'assemblage, dans lesquelles les pièces a assembler ne sont pas fragiles, et ou les tolérances d'assemblage ne sont pas très précises. Le robot PUMA est aussi souvent utilise pour des opérations de soudure et de stockage.

Donc, en règle générale, le champ d'utilisation de ce robot comprend les taches répétitives, sans interaction de l'environnement extérieur.

Le simulateur du PUMA est capable de simuler efficacement les mouvements du vrai robot dans haute fidélité.

II.7.1 Langage de programmation de robot:

Tous les robots utilisent un langage de commande pour que le robot se déplace ou bien précisément exécute une tache.

Le langage de programmation définit les limites de sa capacité du traitement et flexibilité [puma].

Citons quelque langage de programmation autonomes, tel que VAL, AML/2, ARMBASIC, etc.

Le simulateur de robot puma560 utilise le langage VAL[1], VAL est un langage de programmation du robot développé par Unimation, Inc. usages la langue VAL contrôler leur série du PUMA de robots sophistiqués. VAL utilise des ordres tels que MOUVEMENT, OUVREZ, FIN, DEMI-TOUR, DÉLAI, et APPROCHE

II.7.2 La simulation graphique de robot puma560:

Le simulateur du PUMA est capable de produire la Cinématique Avancée pour le robot, aussi bien que rendre effectif la fonction de la Cinématique Inverse. Et la Cinématique peut être exécutée pas à pas.

Le système comprend les unités suivantes:

-un module interpréteur de langage de robot qui interprété le code source de programme de robot.

[1] VAL II est le système de commande et le langage conçus spécialement pour être utilise par les robots industriels Unimation Inc.
Le format de la langue VAL est semblable à cela du langage informatique du but général, DE BASE. Chaque ligne des VAL code imitateur cela d'élément ESSENTIEL avec une instruction par ligne, a suivi par un espace et une instruction, suivies encore par un espace et toute autre discussion. La langue VAL fournit de grands avantages. Un avantage est qu'il tient compte de vraies données du temps être utilisé pour modifier le chemin du robot. En conséquence, le programme du contrôle peut s'ajuster à ce qui se passe dans le vrai monde. VAL a aussi la capacité unique d'être édité pendant l'étape de l'exécution du programme. Cela permet au contrôleur d'ajuster le robot pendant le programme. À cause de ceux-ci et autres avantages, beaucoup de compagnies, tel que Robotique Habile" usage cette langue puissante.

- Un environnement virtuel: un module de l'environnement de la simulation graphique pour simuler et exécuter graphiquement les opérations du robot sur l'écran afin de tester et corriger.

-Un module de la découverte de la collision pour détecter la collision potentielle dans le mouvement de bras du robot.

-Un module du dépannage du programme pour fournir à programmeur du robot un outil du dépannage dans programmation de la tâche.

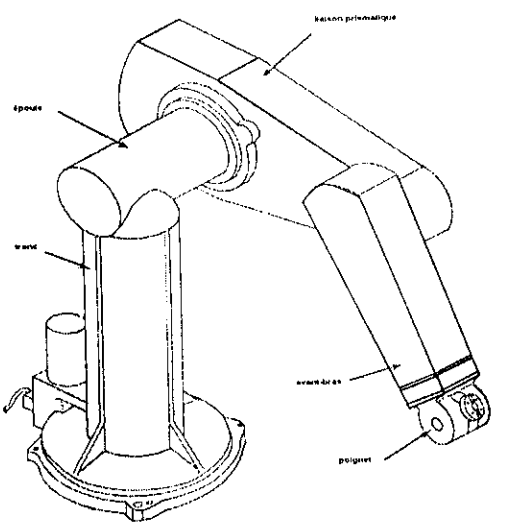
-Un module de la communication entre le robot langue interpréteur module et les environnements virtuels module de l'environnement de la simulation graphique.

II.7.3 L'architecture de simulateur PUMA560:

Ce simulateur comprend plusieurs modules, y compris un éditeur de la langue du robot et module de l'interpréteur; un 3-D environnements virtuels simulateur graphique pour modeler le robot et son module de l'environnement du travail; un module de la détection de collision; un module du dépannage du programme; un module de la communication entre l'éditeur de langage et l'environnement.

II.7.4 Le robot:

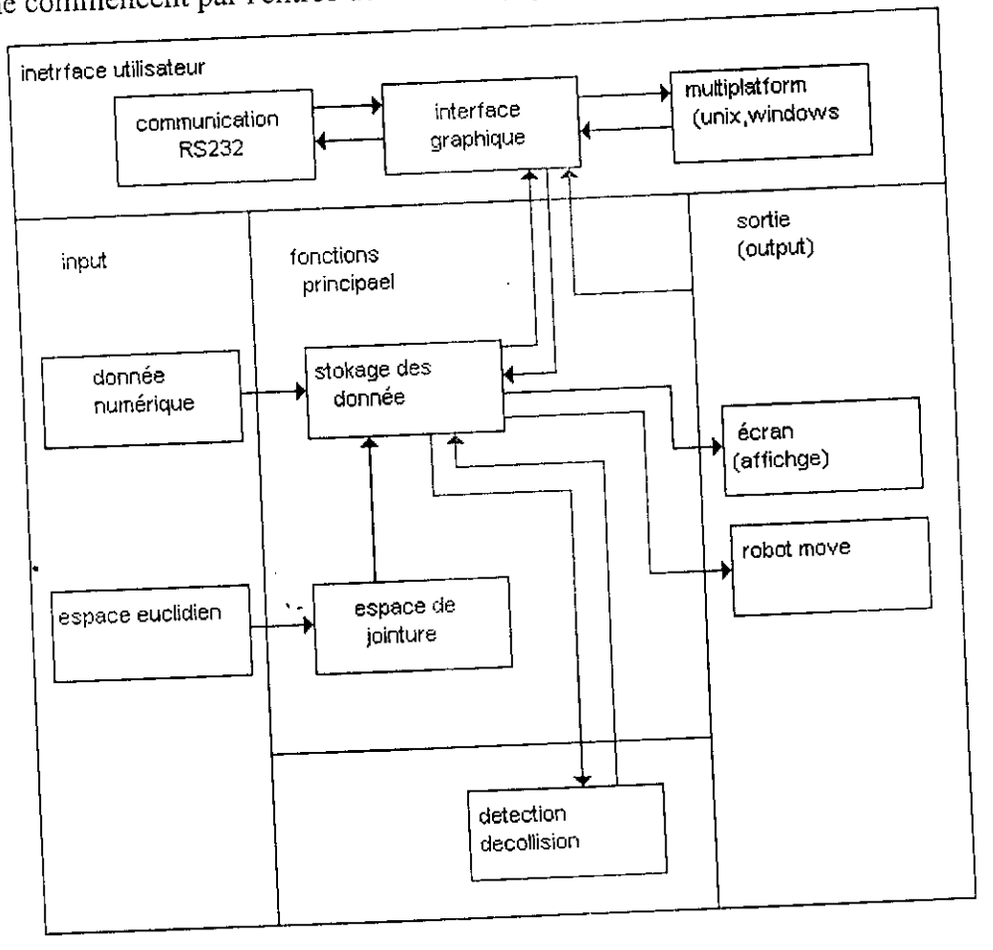
Le bras manipulateur est le composant mécanique du système et comporte six axes de rotation, chacun contrôlé par un servomoteur a courant continu. Chaque partie du bras manipulateur est connectée aux autres par une articulation. Le bras manipulateur, qui contient plusieurs servomoteurs et réducteurs est détaillé sur la figure ci-dessous.



II.7.8 Les diagrammes

II.7.8.1 Diagramme de contexte:

Le diagramme de contexte représente un schéma générale de fonctionnement de système commencent par l'entrée des donnée (input) jusqu'au l'affichage des résultats.



Le diagramme de contexte

Dans cette architecture logiciel on visualise trois partie (*input, fonction principale, output)*

Input: des données numériques en espace articulé (numérique).

Fonction principale: on trouve un module de stockage de donnée (base de connaissance) ou toutes les traces de simulateur sont enregistrées afin de donner au simulateur une certaine connaissance (système cognitif) donc une intelligence au système.

Un module de détection de collision pour assuré une navigation sans aucun risque de collision.

Output: plus une interface graphique cela assure une très bonne affichage d'information (résultats) et une interaction avec l'utilisateur.

II.7.8.2 Le diagramme d'état:

Ce diagramme est très important il décrit touts les taches qui empreint l'utilisateur de l'interface graphique.

Ces branches montrent chaque chemin que l'ordinateur peut prendre pour atteint le résultat final

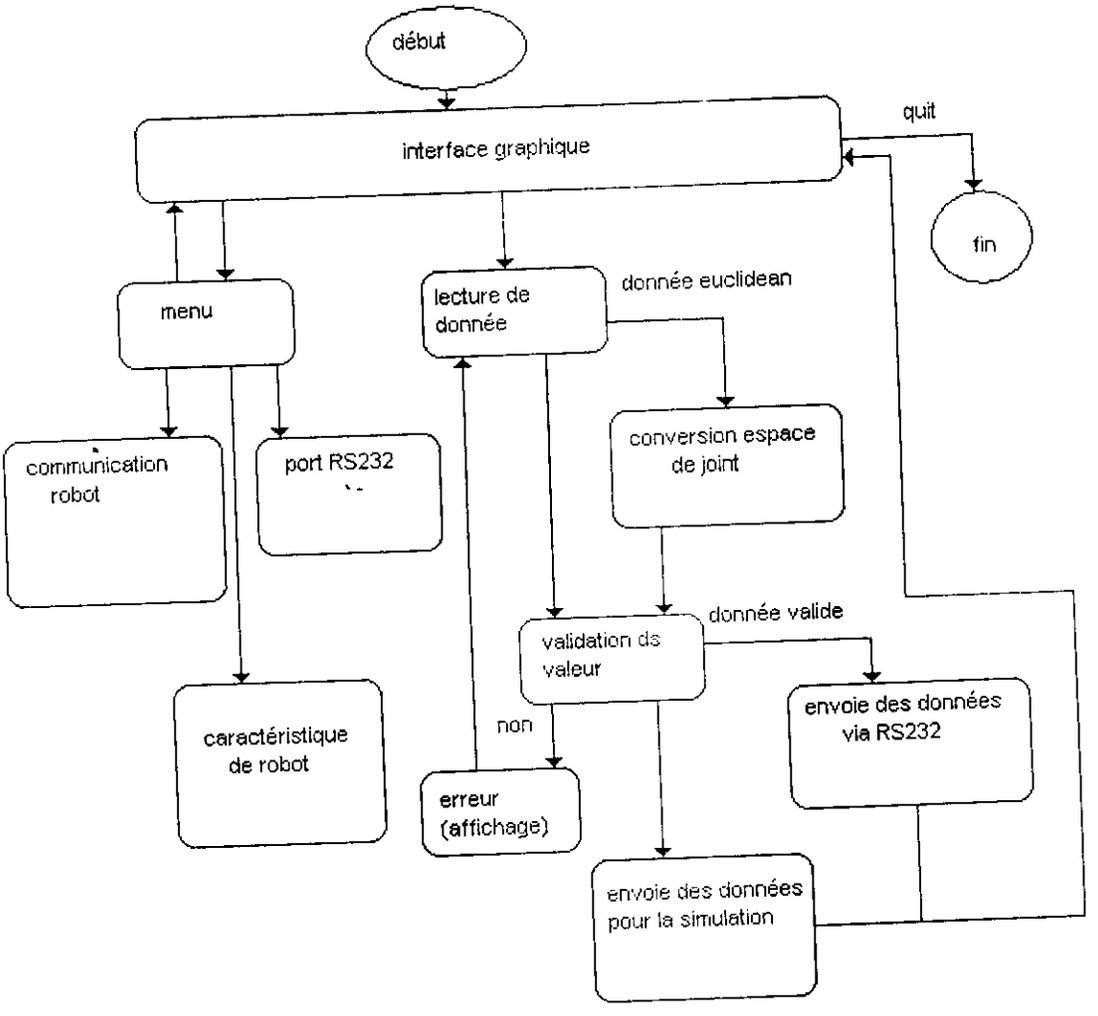


Diagramme d'état

II.8 Le simulateur RCELLSIM

C'est un simulateur graphique implémenté en C++ et utilise la notion orienté objet sous Windows. Ce simulateur est capable de simulé la cinématique du robot, exécuté les algorithmes de manipulation et permis la synchronisation et la communication entre plusieurs robots et machines de cellules de travail.

II.8.1 Les différents objets qui composent le simulateur:

On distingue plusieurs objets citons les :

II.8.1.1 Station de travail:

Contient des objets fixes. Ces derniers interagissent avec les objets qui les supportent et ne changent leurs état ni leurs orientations tout seul.

II.8.1.2 Les pièces de travail:

C'est des objets manipulés par des machines ou bien des robots c-à-d ces états peuvent être modifié que par les objets machine.

II.8.1.3 Les objets machines:

C'est les objets qui peut manipule et modifier les pièces de travail ils sont contrôlé par un programme spécial ce programme doit être écrit avec un langage machine générique; les sous objets de machines sont:

-ROBOT: c'est un bras manipulateur qui contient plusieurs objets (corps, liaisons, l'organe terminal.....)

Plusieurs robots peuvent être simulés en même temps

-CNC: c'est le langage des machines qui est générique ce langage possède uniquement des instructions de synchronisation avec d'autres machines et les processus de manipulation du pièces de travail (changement d'état, changement de position, changement de d'orientation.....).

Ce composant est simulé uniquement pour autorisé l'exécution des programmes machines (la synchronisation entre plusieurs machines).

II.8.2 L'architecture de simulateur RECLLSIM

II.8.2.1 Modèle géométrique: qu'utilise des polygones pour la description géométrique des objets.

II.8.2.2 Modèle cinématique: modélisé par une arbre dynamique qui est une structure de donnée pour modéliser la relation cinématique entre deux objets.

II.8.2.3 Système de contrôle: c'est un module de contrôle machine, il existe que pour les objets de type machines il émule la machine CNC et le module de contrôle machine afin d'exécuté les programme machines pour modifier le modèle cinématique.

Ce système contient trios sous module qui sont: *module d'interprétation, module de génération de trajectoire, modèle inverse.*

II.8.3 Système de contrôle de simulation: contient trois fonction principale:

1) TIMING: pour déterminé un cycle de simulation
2) SYNCHRONISATION: pour synchroniser plusieurs machine
3) CREATION DU ROBOT: initialisation d'un modèle du robot désiré simulé.

II.8.4 Système graphique: à partir de modèle géométrique, modèle cinématique et les paramètres de visualisation et après un cycle de simulation le système graphique génère une image coloré pour visualisé la scène.

II.8.5 Interface système : le système possède deux interface partiellement disjoint qui sont:

Une interface utilisateur: elle permet l'interaction de l'utilisateur avec le système via le Windows (sourie clavier……).

Une interface d'utilisateur machine: autorise deux voie de communication

II.9 Le simulateur simrobot

SIMROBOT est un simulateur graphique permis de similé n'importe quel robot programmé par l'utilisateur, il couvre le modèle physique qui est basé sur la dynamique des corps rigides.

Simrobot est différent des autres simulateurs car simrobot n'est pas spécialisé a une classe du robot précise comme la plupart des simulateurs cette technique est réalisé en utilisant comme plateforme de base le langage XML qui permis aux utilisateurs de

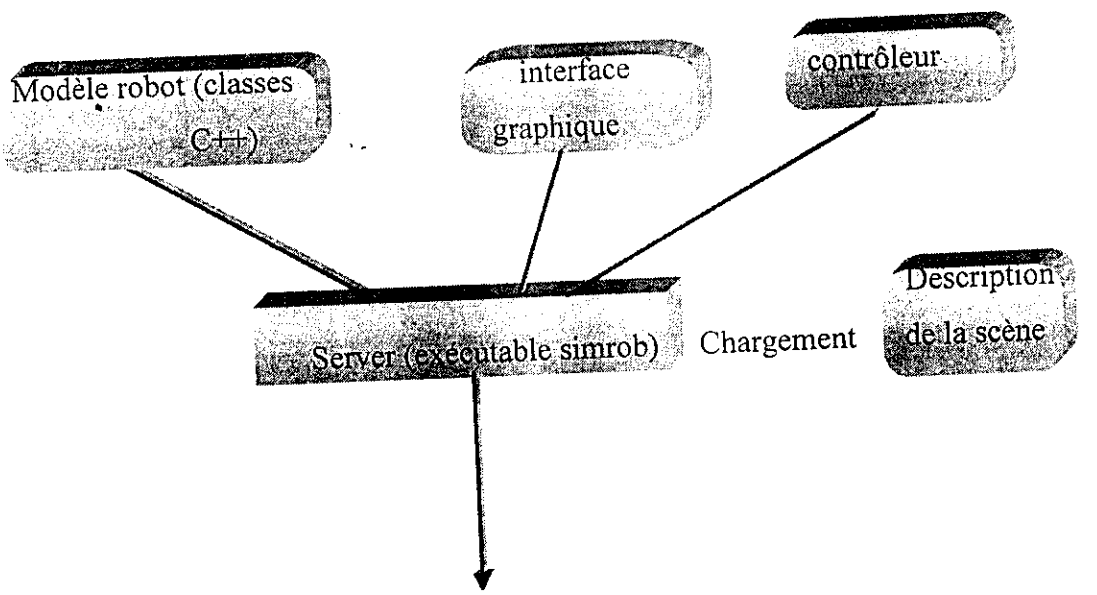
spécifier leurs propre robot et environnement sans aucun extension d'autres langage de programmation.

SIMROBOT offre la possibilité de visualiser les corps rigide de façon arbitraire via ODE[2] (moteur graphique dynamique).

La visualisation des différents événements est basée sur OpenGL.

II.9.1 L'architecture de simulateur SIMROBOT:

L'architecture de ce simulateur est basé sur l'approche Client/Server cette architecture est choisit parce qu'elle offre un avantage très important c'est qu'elle peut arrêté l'exécution de la simulation en cas de panne et autorise un module de dépannage pour l'exécution de l'algorithme du robot.



Architecture de simrobot

II.9.2 Les modules qui composent SIMROBOT

II.9.2.1 modèle robot(corps simrobot): c'est le plus important des modules il est considérer comme un moteur graphique c'est bien le noyau de système ces modules sont le robot et l'environnement. Il modélise le robot et l'environnement simulé et exécute des commendes fournit par le contrôleur.

[2] ODE (Open Dynamic Engine) est un moteur graphique qui possède des bibliothèques graphiques à base de OpenGL. qui permettant de dessiner différentes formes géométriques

On trouve aussi que une grande partie de visualisation est intégré dans ce module (inconv).

Ce module (Server) est une plateforme indépendante connecté aux autres modules (client) via une interface de communication et un protocole bien définit, cela permis un défilement facile aux autres plateformes.

II.9.2.2 Module interface graphique: le rôle de l'interface graphique est la visualisation et l'affichage des informations sur écran et permettre l'interaction avec l'utilisateur.

Le contrôleur: ce module rend effectif un cycle d'action dont il pense (comme il veut).

Un scénario est un ensemble d'action exécuté dans un ordre donc chaque scénario appelé par une telle simulation fait :

La lecture des ressources disponibles.

Planifier en suite l'action.

Finalement mettre l'actuateur aux états désiré.

Scène : c'est l'ensemble de deux modules robot et environnement spécifiés est modélisés a base de langage XML, ce fichier sera charger a partir d'un emplacement connu.

L'utilisation de ce fichier externe permet la modélisation de la scène sans aucune modification ou bien une extension de code source de simulateur, donc le processus de modélisation devient simple pour tous les utilisateurs sans aucune compétence de programmation

II.10 Le simulateur UBERSIM

Nous allons présenter un simulateur graphique qui possède plus de caractéristique que les précédents; ce simulateur destiné pour construire un environnement un robot basé sur la vision dans le contexte de la perception par exemple le robot *SEGWAY RMP*ou bien *AIBO SONY*.

II.10.1 L'approche comportementale et spécification de simulateur:

Dans ce qui suit nous allons expliquer la motivation de simulateur UBERSIM.

II.10.2 L'architecture:

Ce simulateur est basé sur une architecture client/Server ou le client communique avec le serveur via un protocole de communication TCP/IP.

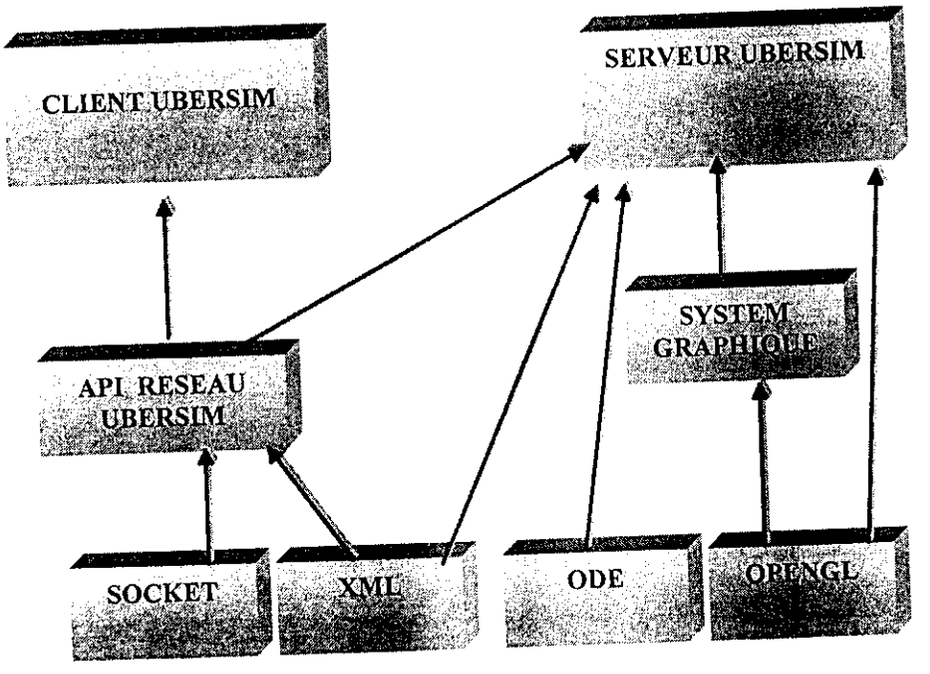
UBERSIM est aussi implémenté sous Windows et ces classes en C, il utilise l'ODE comme un moteur graphique.

D'autre part et pour un raison de souplesse UBERSIM fournit un langage de description XML pour la spécification des composants virtuel de robot donc pour modifier le robot il suffi un simple changement dans la définition XML de robot et nous n'avons pas besoin de recompiler le programme au niveau serveur.

Pour une très bonne implémentation de simulateur il faut respecter les contraintes suivantes:

1) une dynamique réaliste pour le robot et une exactitude de simulation
2) un support de haute fréquence pour les boucles de contrôle
3) synthèse de la vision exacte avec une bonne perception d'objets.
4) spécification du robot flexible et extensible.

II.10.3 L'architecture client/Server:



ARCHITECTURE DE UBERSIM

Cette architecture est traditionnelle car elle est utilisée par plusieurs simulateurs mais chaque simulateur possède ces propres clients et leur propre serveur.

UBERSIM possède un client/serveur qui fournit une intelligence pour augmenter l'interchange entre le code de système de contrôle de simulateur et le code de système de contrôle du robot physique.

II.10.3.1 Le serveur:

Le code source de serveur contient des primitifs (c'est des fonctions implémentées en langage bas niveau) pour les corps rigides et pour les senseurs, les actionneurs et touts les composants du robot ces derniers possèdent des paramètres aident a maximisés la réutilisation de simulateur selon le chois de l'utilisateur par exemple les primitifs qui incluent les corps rigides tel que les prismes rectangulaire, sphères et tous les outils graphique pour particulier un robot sont intégrés dans le serveur donc il suffit de compiler un code en XML crée par l'utilisateur(client) par le serveur.

II.10.3.2 Le client:

La responsabilité de client est exprimée en deux processus principaux:

Le premier processus est établir une socket TCP et transmet au serveur un fichier XML qui contient les primitifs qui définissent le robot.

Le second implique le traitement message qui détermine le code source de robot, nouveau message, envoyer message...etc.

A ce moment le serveur intervient, il convertie le fichier client dans les composants simulés et utilise l'ODE pour établit la prochaine étape qui est la construction de l'environnement, dans cette étape envoie au client un document sous une forme prédéfinit, ce document contient une information obtenus a partir d'un senseur (image d'une appareil photo qui envoyer comprimé au client).

II.10.4 Système graphique:

C'est un open source graphique qui est responsable a la représentation graphique de la scène.

II.11 CONCLUSION:

La simulation en robotique de manière générale est un outil qui accélère le développement du comportement du robot, la question qui se pose: ces simulateurs répond t-il au ces exigences? D'autre manière est-ce que les simulateurs simulent fidèlement le monde réal? (Robot + environnement), la réponse certainement non mais l'essentiel est de réalisé un simulateur qui capture convenablement les interaction entre le robot et son environnement simulé.

RECELLSIM est un simulateur capable de synchronisé plusieurs robots et machines d'une cellule flexible et possède un moteur graphique dynamique mais malgré ces qualité il rencontre des problème tel que:
La complexité: pour la génération d'un mouvement d'un seule robot on trouve souvent des situation critique sans l'aide du robot réal imaginants si on y devant plusieurs machines et robots, même problème pour la synchronisation de plusieurs robots et chaque robot possède plusieurs composant (senseurs, actionneurs....).
Autre problème c'est lorsqu'on y dans la phase de teste et validation si les résultats ne sont pas souhaitable alors la correction devient difficile et parfois coûteux.

PUMA560 est implémenté en architecture cognitif il possède une base connaissance (base de stockage) [voir architecture puma560] alors on trouve non seulement les limites qui possède cette approche mais aussi autre problème c'est que les données d'entrés de système sont des valeurs numérique(dans l'espace articulé) calculés auparavant donc forcement ces calcules possèdent un intervalle d'erreur ou bien un précision de calcule d'où les résultats ne sont pas optimale exemple si on veut déplacer l'organe terminale du robot au point (x,y,z), tout les points du corps du bras se déplace pour que l'organe terminale touche la destination mais avec une erreur de calcule considérable l'organe ne touche jamais ce point.

Finalement pour atteindre un simulateur proche a la perfection il faut implémenté une architecture robuste et convenable de tel sorte les robots naviguent et réagissent de façons logique et proche a la réalité dans leurs environnement.

CHAPITRE III

APPROCHE DE CONTROLE/COMMANDE PROPOSEE

CHAPITRE III

APPROCHE DE CONTROLE/COMMANDE PROPOSEE

III.1. Introduction

Le système de pilotage d'un robot manipulateur mobile doit être perçu comme une entité unique et globale même si elle présente un ensemble de sous-systèmes autonomes (plateforme mobile + bras manipulateur). Cette globalisation introduit des besoins de communication, d'interaction, de partage de connaissances et de partage de ressources.

L'objectif visé dans ce chapitre est de proposer une architecture multi-agents pour le contrôle d'un système robotique constitué d'une plateforme mobile surmontée d'un bras manipulateur (robot manipulateur mobile). Cette architecture doit être basée sur un modèle générique (qui est indépendant du type de robot utilisé et des différentes ressources existantes ou les ressources opérationnelles) pour permettre l'ajout de nouvelles fonctionnalités et de nouvelles ressources.

Avant de présenter en détail l'architecture de contrôle proposée pour le du robot (Robot *RobuTER*), on va procéder d'abord à une description plus ou moins détaillée de l'architecture matérielle du robot.

III.2. Description du robot *RobuTER/ULM*

Le robot qui fait l'objet de cette approche (*RobuTER/ULM*) est un ensemble d'une plateforme surmontée par un bras manipulateur. La plateforme comporte deux roues motrices permettant une grande mobilité sur sols lisses. Elle comporte aussi deux roues folles à l'avant de la plateforme assurant la stabilité de l'ensemble. Piloté en différentiel de vitesse, le *RobuTER* peut tourner sur lui-même.

Cette plateforme mobile est équipée d'un bras manipulateur Ultra-Léger à six d.d.l avec une pince. Ce bras est adapté pour le transport ou la manipulation de petites pièces, que le ramassage d'échantillon au sol.

L'ensemble de la plateforme munie du bras est contrôlé par un PC embarqué (Pentium MMX 233 MHz sous Linux) et 4 contrôleur MPC555. Le premier MPC555 contrôle la base mobile, Le deuxième contrôle les trois premières liaisons du bras, le troisième MPC555 contrôle les trois dernières liaisons. Le dernier MPC555 contrôle le capteur d'efforts qui est placé sur le sixième axe du bras. Ce robot peut être contrôlé aussi bien par un PC HOST sous Linux (Pentium MMX 600 MHz) via une connexion TCP/IP sans fil.

La figure III.1 montre le robot manipulateur mobile *RobuTER* décrit ci-dessus :

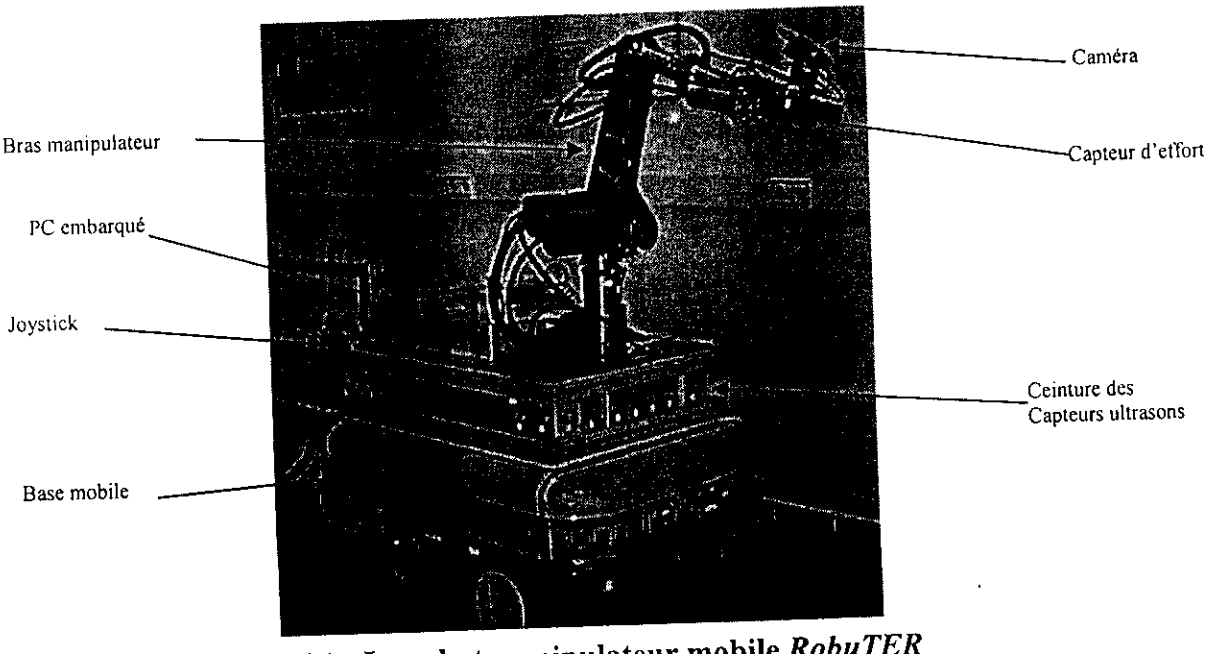


Figure III.1 : Le robot manipulateur mobile *RobuTER*

L'architecture décrite ci-dessus peut être schématisée de la façon suivante (voir la figure III.2). Elle présente la conception du robot de haut en bas, c.-à-d. en allant du niveau missions (le haut niveau) jusqu'au niveau des actions élémentaires (niveau physique) en passant bien sûr par le niveau tâche.

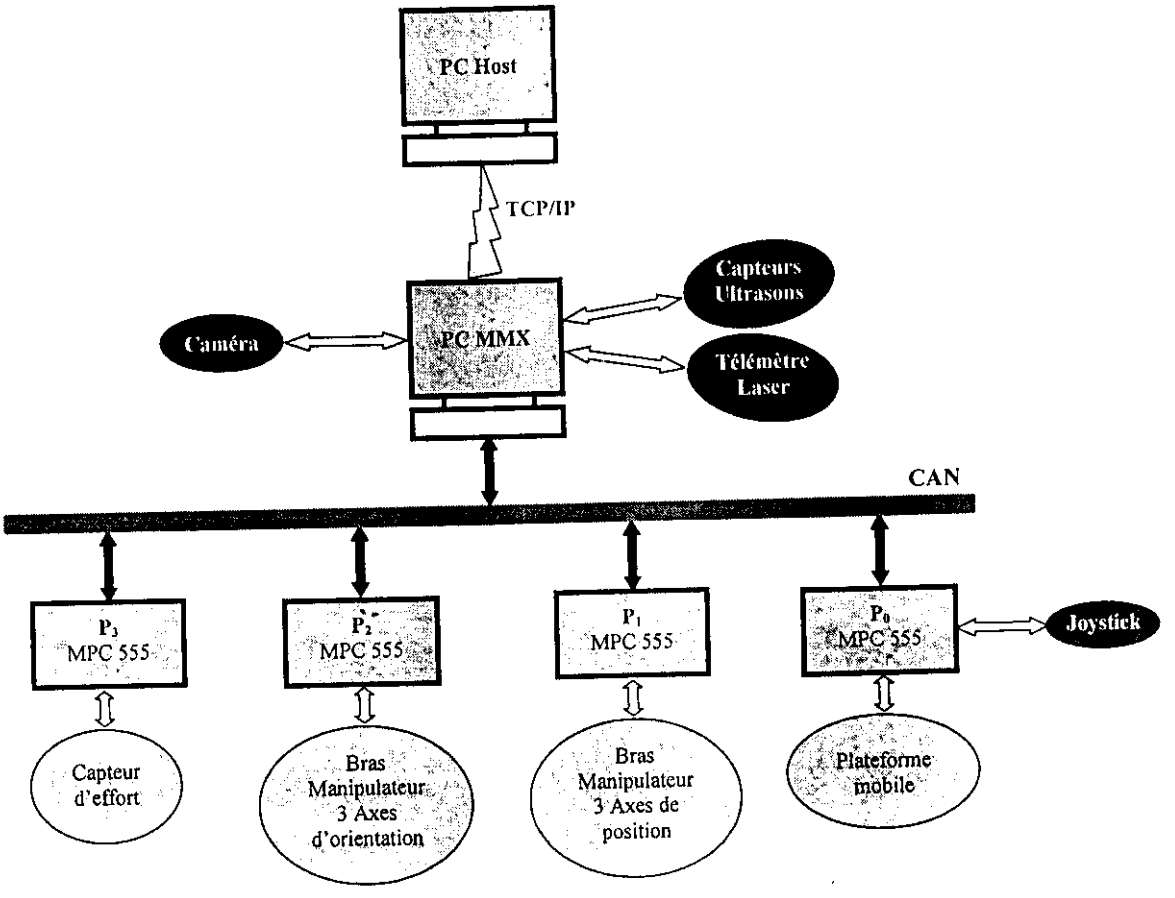


Figure III.2 : Architecture matérielle du robot RobuTER

Les moyens de perception du robot *RobuTER* (la plateforme et le bras manipulateur) sont de deux types proprioceptif et extéroceptifs, soit :

III.2.1. Proprioceptif

Deux capteurs incrémentaux permettant l'extraction des données odométriques, un sur chacune des roues motrices. Grâce à ce système, le robot connaît sa position à tout moment par rapport à un repère initial. Un deuxième groupe comprenant six capteurs incrémentaux installé sur chaque moteur des articulations du bras manipulateur. La résolution de tous les capteurs est de 500 points/tour.

III.2.2. Extéroceptif

Le robot dispose d'une ceinture de 24 capteurs ultrasonores pour détecter les obstacles. Chaque capteur est un émetteur-récepteur qui mesure le temps de vol d'une onde ultrasonore. Ce temps est alors associé à une distance robot-obstacle. La base dispose aussi d'un télémètre laser mis en rotation afin de balayer un plan horizontal, et qui permet de mesurer la distance des objets qui coupent ce plan. La résolution varie de 0.25, 0.5, 0.75 et 1 degré.

Le bras manipulateur ultraléger comporte un capteur d'effort six axe permettant de mesurer l'effort appliqué au niveau de l'organe terminal. Aussi, on trouve une caméra CCD noir et blanc positionnée sur la pince du robot pour l'acquisition des images en temps réel.

III.3. Pourquoi une architecture Multi Agents ?

Le contrôle multi-agent-est nécessaire quand plus d'un robot est utilisé pour effectuer des tâches, quand un seul robot doit coordonner l'utilisation de ses propres ressources (par exemple pour un robot manipulateur mobile, le bras et la plateforme), ou quand un ensemble de robots fonctionnent indépendamment sur des tâches multiples dans un environnement partagé [Bonasso95].

On peut expliquer la nécessité d'une distribution de l'activité et de l'intelligence pour les raisons suivantes [Ferber95] :

- *Les problèmes sont souvent physiquement distribués.*
- *Les problèmes sont fonctionnellement très distribués et hétérogènes.*
- *Les réseaux imposent une vision distribuée* : À l'heure des réseaux, où toute l'information et la puissance de traitement sont réparties sur un certain nombre de nœuds, il est nécessaire de penser en termes de systèmes distribués.
- *La complexité des problèmes impose une vision locale* : Lorsque les problèmes sont trop vastes pour être analysés globalement, les solutions fondées sur des approches locales permettent souvent de les résoudre plus rapidement.

D'autre part :

- *Les systèmes de contrôle doivent pouvoir s'adapter de plus en plus à des modifications de structure et d'environnement*. Devant la complexité de plus en plus croissante des applications, il est nécessaire de doter les systèmes de commande de capacités dévolution (ajouts de nouvelles fonctionnalités, modifications de l'utilisation, etc.).
- *Le Génie Logiciel va dans le sens d'une conception en termes d'unités autonomes en interactions*. L'histoire du développement des logiciels montre que la réalisation de programmes informatiques suit une démarche visant à la réalisation de systèmes conçus comme des ensembles d'entités de plus en plus distribuées, mettant en jeu des composants autonomes.
- *La conception de systèmes complexes émerge vers une nouvelle discipline de recherche* : celle des systèmes multi-agents. Cette nouvelle discipline repose sur des concepts propres très puissants qui permettent de proposer des solutions réactives et robustes pour la conception de systèmes complexes.

III.4. Architecture globale du système multi-agents

L'architecture multi-agents de contrôle proposée (Figure III.4) consiste en un méta-agent *ManipulateurMobile* constituée d'un ensemble de trois agents hybrides qui pilotent l'ensemble de ressources hétérogènes et un quatrième agent qui sert comme vérificateur graphique de l'exécution des tâches en temps réel (*Agent Simulateur graphique*). Deux agents, *Agent RobotMobile* et *Agent RobotManipulateur*, contrôlent le robot mobile et le bras manipulateur respectivement. Le troisième agent (*Agent Superviseur*) a pour rôle la gestion des missions et la distribution des tâches sur les différents agents ainsi que la communication et la négociation avec les autres systèmes via une passerelle dans le cas où on veut utiliser le robot avec d'autres systèmes existants (par exemple, une intégration de ce robot dans une cellule de production ou dans un système à plusieurs robots).

Les quatre agents possèdent un niveau substantiel d'autonomie à travers des capacités de perception de l'environnement qui est considéré ici comme dynamique, de communication avec d'autres agents, de navigation autonome (évitement d'obstacles imprévus), etc. Ils possèdent aussi des connaissances suffisantes pour une prise de décision autonome.

III.4.1. Caractéristiques de l'architecture des agents

On va analyser les caractéristiques que doit posséder l'architecture des trois agents dans ce contexte. L'objectif est de développer un système de contrôle qui possède les propriétés suivantes :

- *Modularité* : La modularité de l'architecture de contrôle des agents doit s'articuler autour de la décomposition en modules qui peuvent être développés et mis en œuvre. L'ouverture et la reconfigurabilité sont des caractéristiques permettant à tout système de contrôle d'évoluer en lui rajoutant de nouvelles fonctionnalités.
- *Réactivité à l'environnement* : Les agents doivent être capable de gérer en temps réel des événements extérieurs asynchrones afin de respecter la dynamique de l'environnement et celle de l'architecture. Un évènement extérieur peut avoir plusieurs origines : présence d'un obstacle imprévu, panne inopinée, requête en provenance d'un autre agent, etc. La réactivité implique généralement un traitement en temps réel de ces événements. Ainsi, le système de contrôle doit intégrer la notion de priorité et d'urgence de traitement des événements.
- *Comportements intelligents* : L'intelligence des agents se traduit par des capacités de perception, de communication, de raisonnement et d'action. L'intelligence peut être située à plusieurs niveaux :

    - Le premier niveau est associé à l'environnement qui est dynamique. Il est indispensable de doter les agents de comportements réflexes lui permettant d'éviter des obstacles rencontrés sur une trajectoire nominale, d'aller vers un point cible, etc. Ces comportements reposent sur un contrôle on–line de la trajectoire.
    - Le deuxième niveau d'intelligence gère les comportements du premier niveau. Il est donc nécessaire de disposer d'un mécanisme qui met en œuvre des changements de comportements de bas niveau pour s'adapter aux événements.

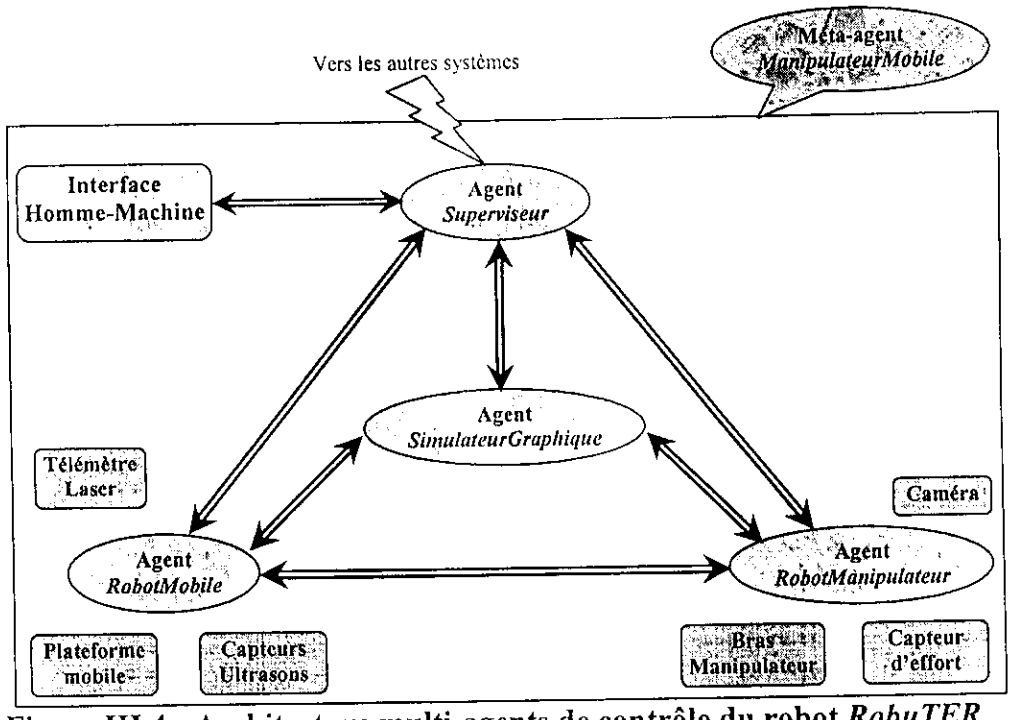


Figure III.4 : Architecture multi-agents de contrôle du robot *RobuTER*

III.4.2. Concept d'agent comportemental

L'architecture met en œuvre des entités comportementales concurrentes et des mécanismes d'activation/désactivation des comportements. Elle est inspirée de l'approche comportementaliste en général, puisque le système de contrôle du robot repose sur des entités informatiques indépendantes et expertes appelée agents comportementaux. Ces entités ont la faculté de doter le robot d'un comportement particulier.

III.4.2.1. Un comportement

Il est défini comme une évolution particulière observée du robot dans son monde réel. Il correspond à un mécanisme reliant *Perception/Communication–Raisonnement/Décision–Action* du robot sur et dans son monde réel. Sous l'influence de ce mécanisme, le robot agit sur le monde réel en percevant (couple *Perception/Communication*) ses modifications et en s'y adaptant.

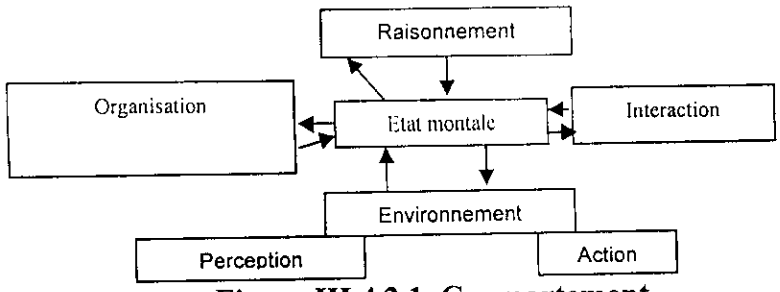


Figure III.4.2.1. Comportement

III.4.2.2. Un agent comportemental

Il peut être défini comme étant l'entité virtuelle, autonome et experte qui confère à l'agent robot un certain comportement face à son environnement. Chaque agent comportemental doit être doté d'une certaine autonomie permettant au robot d'évoluer sous sa seule influence et posséder une fonction de satisfaction selon le type de comportement (éviter les obstacles, ordonnancer, négocier, etc.). La mise en œuvre d'un agent comportemental se traduit par l'activation d'une ou de plusieurs capacités. Un exemple est celui de la navigation réflexe pour l'évitement d'obstacles qui agrège les capacités de perception, de raisonnement et d'action.

III.4.3. Environnement

L'environnement est constitué d'obstacles statiques (connus ou inconnus) et d'obstacles mobiles (autres robots mobiles ou manipulateurs, les opérateurs humains, etc.).

III.4.4. Modèle global de l'architecture du méta-agent *ManipulateurMobile*

Compte tenu de l'analyse effectuée dans les paragraphes précédents, on propose un modèle où le méta-agent *Manipulateur Mobile* est doté des capacités suivantes :

- *Perception* : Des scrutateurs évaluent en permanence le modèle du monde réel. Le contrôle continu de la situation de l'agent assure sa réaction à l'occurrence d'événements imprévus à tout moment. La fonction de perception permet d'une part, de percevoir l'état de l'environnement et d'autre part, de percevoir les états internes du méta-agent. Dans le premier cas, la fonction de perception permet de collecter les informations issues des différents capteurs qui équipent le robot. Dans le deuxième cas, la fonction perception est une estimation des états internes de l'agent (état d'exécution d'une opération, du plan, comportements déclenchés, etc.)
- *Communication* : Cette capacité assure la communication inter-agents qui repose sur des modules de réception, d'émission et d'interprétation de messages suivant un protocole d'interaction donné.

- *Actions sur l'environnement et sur les autres agents* : Dans les capacités d'action d'un agent, on distingue deux types, celles qui agissent sur l'environnement et celles qui agissent sur les autres agents. Parmi les actions qui agissent sur l'environnement réel de l'agent, on peut citer, le démarrage d'un plan, l'activation ou la désactivation d'une action. Dans le mécanisme d'interaction entre agents, un agent peut être sollicité par un autre pour lui fournir un service (partage de connaissances, examen d'un plan proposé par un autre agent lors d'une négociation). Dans ce cas, les réponses (messages) émises par l'agent sollicité constituent des actions sur l'agent solliciteur.
- *Raisonnements et contrôle de l'exécution* : Les capacités de raisonnement constituent le centre de décision des différentes compétences du méta-agent. En fonction du comportement déclenché, un module de raisonnement associé à un comportement donné détermine une réponse adaptée à la situation dans laquelle se trouve le méta-agent.

III.5. Structure des modèles des agents

Pour répondre aux caractéristiques décrites précédemment, on propose dans ce qui suit un modèle global d'architecture des trois agents.

III.5.1. Modèle d'architecture de l'agent *Superviseur*

Il est nécessaire d'utiliser un agent *Superviseur* dont la tâche est d'interpréter la mission introduite par l'opérateur en plan de tâches et de les distribuer sur tous les autres agents constituant l'organisation. Le rôle de l'agent *Superviseur* prend fin dès que cette phase d'interprétation et de distribution de tâches se termine. Chaque plan de tâches est ensuite transmis à un l'agent correspondant pour être exécuté.

Les fonctions de l'agent *Superviseur* peuvent être résumées dans les points suivants :

- Prise de décision de la faisabilité de la mission selon la disponibilité et l'état des ressources du méta-agent *Manipulateur Mobile* ;
- Interprétation des missions en plan de tâches ;
- Distribution des tâches sur les autres agents ;

- Gestion de la configuration matérielle du robot ;
- Communication et négociation avec les autres systèmes multi-agents dans le cas par exemple d'une intégration de ce robot dans une cellule de production ou dans un système à plusieurs robots. Cela implique bien sûr l'implémentation d'une passerelle entre ce système et les autres systèmes.

II.5.1.1. Base de connaissances

Elle décrit la configuration matérielle du robot (plateforme + bras manipulateur), elle regroupe :

- Une liste de toutes les composantes et les ressources du robot, leurs types et leurs états (opérationnelle ou en panne) ;
- Une liste de toutes les actions réalisables par les ressources du robot.

Ces connaissances sont obtenues dynamiquement grâce aux messages envoyés par les autres agents lors des modifications de la configuration matérielle du robot.

Cette base de connaissances contient, à tout instant, l'ensemble ordonnancé des tâches d'une mission à exécuter. Elle décrit aussi les engagements de l'agent sur le lancement des tâches à exécuter. A chaque début et fin d'exécution d'une tâche cette base de connaissances est mise à jour :

- identificateur de la tâche ;
- identificateur de la mission ;
- date de début et de fin d'exécution ;
- état de la tâche (exécution, terminée, bloquée).

III.5.1.2. Compétences

Les compétences de l'agent sont ses capacités décisionnelles sur la résolution des problèmes qui lui sont confié :

- *Module Gestion de missions* : si la mission à effectuer est faisable, il l'interprète en plan de tâches ordonnancé et distribue les différentes tâches sur les agents concernés.
- *Module Gestion de tâches et Contrôle d'exécution* : lancer l'exécution effective des tâches. Le lancement est une annonce de l'arrivée d'une nouvelle tâche aux autres agents. Il gère aussi les activités internes et externes de l'agent :

    - *Internes* : concernant la planification et le suivi des tâches internes de l'agent tels que l'analyse des messages, l'activation des compétences, la mise à jour des connaissances, etc.
    - *Externes* : concernant le suivi des tâches externes avec les autres agents tels que la demande d'une information à un agent, l'envoi d'informations, les requêtes de coopération (appel d'offres, proposition, contrat).

III.5.1.3. Communication

Les connaissances de communication permettent à l'agent de communiquer avec les autres agents du système. Cette communication se fait par échange de messages. Le module de communication regroupe les fonctions suivantes :

a) *Envoi de messages* ;
b) *Réception de messages* ;
c) *Interprétation des messages* : gérer et interpréter les messages reçus. Les différents types de messages que le module interprète sont :

    - *Configuration* : Quand on configure un autre agent que le *superviseur*, il l'envoi un message de configuration pour la mise à jour de sa base de connaissances.
    - *Proposition* : Lorsque l'agent *superviseur* envoie des appels d'offre sur une tâche aux agents, il obtient en réponse des propositions. A la réception d'une proposition, le module met à jour la table d'ordonnancement. Si le module reçoit des propositions relatives à une tâche, il sélectionne la meilleure proposition et envoie une demande de contrat à l'agent concerné.
    - *Acceptation* : c'est un message d'acceptation du contrat, l'agent *superviseur* stocke la tâche concernée dans le plan de tâches pour être exécutée à la date de début spécifiée dans le contrat.

- *Refus* : c'est un message de refus de contrat. A la réception d'un tel message, l'agent *superviseur* évalue de nouveau toutes les propositions et envoie une autre demande de contrat à l'agent qui a offert la meilleure proposition.
- *Annulation* : c'est un message d'annulation de contrat. Le module supprime les informations relatives au contrat annulé dans la table d'ordonnancement et informe le module *Gestion des tâches* de l'annulation.

III.5.1.4. Interface opérateur

Elle permet à l'opérateur :

- la saisie et la mise à jour des missions à exécuter ;
- la saisie et la mise à jour des tâches ;
- lancement de l'exécution des tâches ;
- Interruption de la mission ou des tâches.

La structure de l'agent *Superviseur* est schématisée dans la figure III.5.

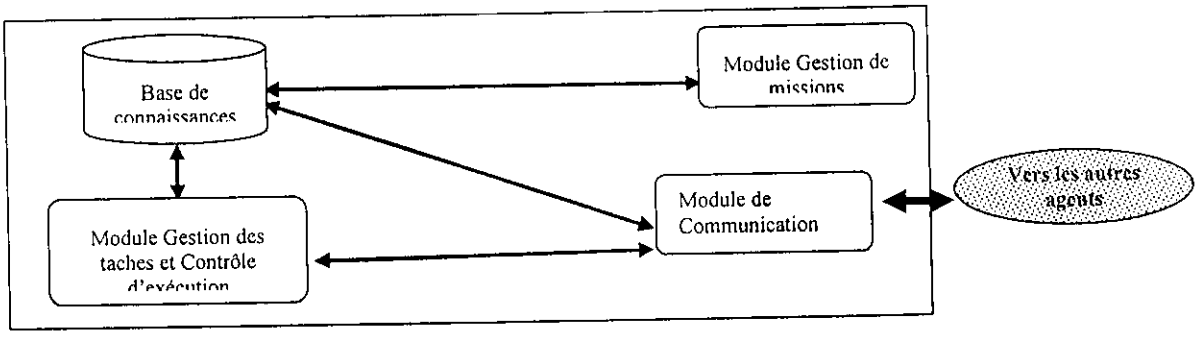


Figure III.5 : Modèle de l'agent *Superviseur*

III.5.2. Modèle d'architecture de l'agent *RobotMobile*

L'architecture globale de l'agent *RobotMobile* est similaire à celle de l'agent *Superviseur* à l'exception de quelques modules. Cet agent assure la gestion locale de la base mobile du robot, en intégrant les fonctions de contrôle en temps réel, à savoir : la planification, l'exécution, la surveillance, la communication, et le dialogue opérateur.

L'architecture proposée pour l'agent *RobotMobile* confère à la plateforme mobile à la fois des capacités réactives et des capacités délibératives pour raisonner sur des situations complexes :

- l'agent doit avoir, suivant sa situation, plusieurs comportements. Il doit mettre en œuvre des mécanismes de raisonnement intelligents sur ses actions et sur son plan de tâches.
- l'agent doit contrôler de manière autonome l'ensemble de ses actions en fonction de son état interne et de son environnement. Il doit donc connaître à tout moment son état d'exécution qui est lié au contexte de son environnement (perturbations dans le plan de tâches, présence d'obstacles), s'y adapter en élaborant un autre plan de tâches.

III.5.2.1. Base de connaissances

Cette base contient des informations décrivant la plateforme (identificateur, liste des actions, durées des actions, etc.).

L'agent dispose de l'expertise des pannes pouvant affecter la ressource (cette expertise est constituée d'un ensemble de règles de diagnostic qui servent à identifier les pannes) et de l'expertise de traitement (représentée par des règles de production). Il dispose également d'un agenda représentant les engagements de l'agent (à chaque début et fin d'exécution d'une opération cet agenda est mis à jour). Les informations qui caractérisent un engagement sont :

- identificateur de la tâche ;
- identificateur de l'opération ;
- date début et date fin d'exécution d'une opération ;
- état opération (début, terminée, bloquée).

Cette base de connaissances contient aussi les informations statiques (c.-à-d. les paramètres géométriques, cinématiques, et dynamiques de la base mobile) et dynamiques (la configuration courante des composantes).

La base contient finalement toutes les informations recueillies par les capteurs de la ressource (capteurs ultrasons, odométriques, ...) pour la modélisation de l'environnement (construction et maintenance du modèle de l'environnement) du robot.

III.5.2.2. Compétences

Les compétences de cet agent sont les suivantes :

- *Module Gestion des capteurs* : Il s'agit de générer et maintenir une meilleure estimation de l'état de l'environnement qui peut être utilisé pour contrôler les actions en cours et planifier les comportements futurs. Cette estimation est stockée dans la base de connaissances.
- *Module de Configuration* : permet d'affecter l'agent au contrôle d'une ressource bien précise. Il s'agit de saisir les différentes actions que la ressource peut exécuter.
- *Module Planification de trajectoires et de Navigation* : permet la planification de trajectoires sans collisions.
- *Module Gestion des actions et Contrôle d'exécution* : lance l'exécution des actions sous contrat. Pour cela, il consulte la base de connaissances (les informations relatives aux actions pour lesquelles l'agent *RobotMobile* est contractant) et met à jour son agenda local. Il collecte aussi toutes les informations de suivi de l'état de la plateforme pour le contrôle d'exécution.
- *Module Détection des anomalies, Diagnostic et Traitement* : analyse continuellement l'état de la ressource, et de l'action en cours d'exécution. La détection d'anomalie est effectuée avant (pré-contrôle) et après (post-contrôle) chaque exécution d'action. Si aucune erreur n'a été détectée alors il met à jour de l'état de l'action dans la base de connaissances, sinon il établit le diagnostic relatif à la panne détectée et exécute le traitement approprié.

III.5.2.3. Communication

Cette communication se fait par échange de message. Il s'agit d'un ensemble de modules utilisés pour coopérer avec les autres agents (ordonnancement et réordonnancement dynamique des tâches et des actions, formuler des requêtes d'information, transmettre des informations utiles ou des réponses à des requêtes) :

a) *Envoi de messages* : envoyer les messages aux autres agents ;
b) *Réception de messages* : recevoir les messages en provenance des autres agents ;
c) *Interprétation des messages* : interpréter les messages reçus. Les différents types de messages sont :

- *Configuration* : Le message de configuration est une requête que l'agent *RobotMobile* doit exécuter sur sa base de connaissances.
- *Appel d'Offre* : L'agent *RobotMobile* peut envoyer et recevoir des appels d'offre. A la réception d'un appel d'offre, il répond de deux manières :
  - Si l'appel d'offre concerne une tâche qui contient plusieurs actions, l'agent construit une nouvelle sous-tâche (sans la dernière action) et envoie un appel d'offre à la dernière action de la nouvelle sous-tâche (l'avant dernière de la tâche) aux agents capables d'exécuter cette action.
  - Si l'appel d'offre concerne une tâche qui contient une seule opération, il envoie une proposition à l'expéditeur de l'appel d'offre en se basant sur la disponibilité de la ressource qu'il contrôle.
- *Contrat* : C'est un message de demande de contrat, l'agent répond à ce type de message de deux façons :
  - Si la tâche contient une seule opération, l'agent répond par une acceptation de contrat dans le cas où il n y a pas de conflit, sinon il refuse le contrat.
  - Si la tâche contient plusieurs actions, l'agent transmis une demande de contrat à l'agent suivant (celui qui a offert la meilleure proposition concernant l'avant dernière action).
- *Acceptation* : C'est un message d'acceptation de contrat. S'il n y a pas de conflit avec une autre tâche, l'agent *RobotMobile* envoie une acceptation de contrat. Sinon, il envoie un refus.
- *Refus* : C'est un message de refus de contrat.
- *Annulation* : C'est un message d'annulation de contrat.

La structure de l'agent *RobotMobile* est montrée par la figure III.6.

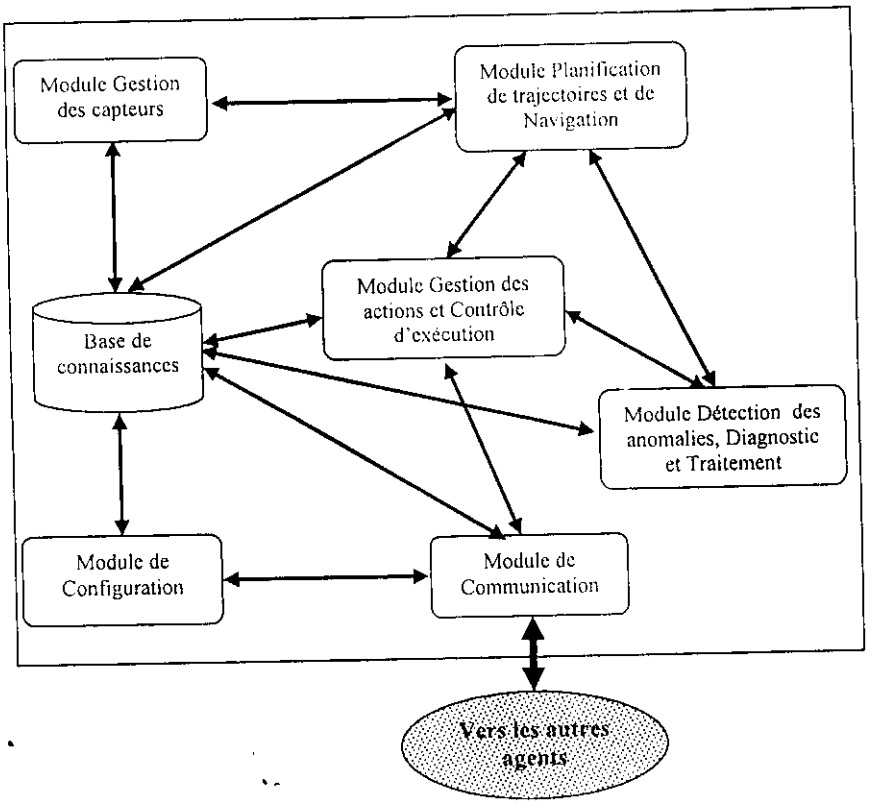


Figure III.6 : Modèle de l'agent *RobotMobile*

III.5.3. Modèle d'architecture de l'agent *RobotManipulateur*

L'architecture de l'agent *RobotManipulateur* est similaire à celle de l'agent *RobotMobile*. Cet agent assure la gestion locale du bras manipulateur, en intégrant aussi les fonctions de contrôle temps réel : la planification, l'exécution, la surveillance, la communication, et le dialogue opérateur.

La structure de l'agent *RobotManipulateur* est montrée par la figure III.7

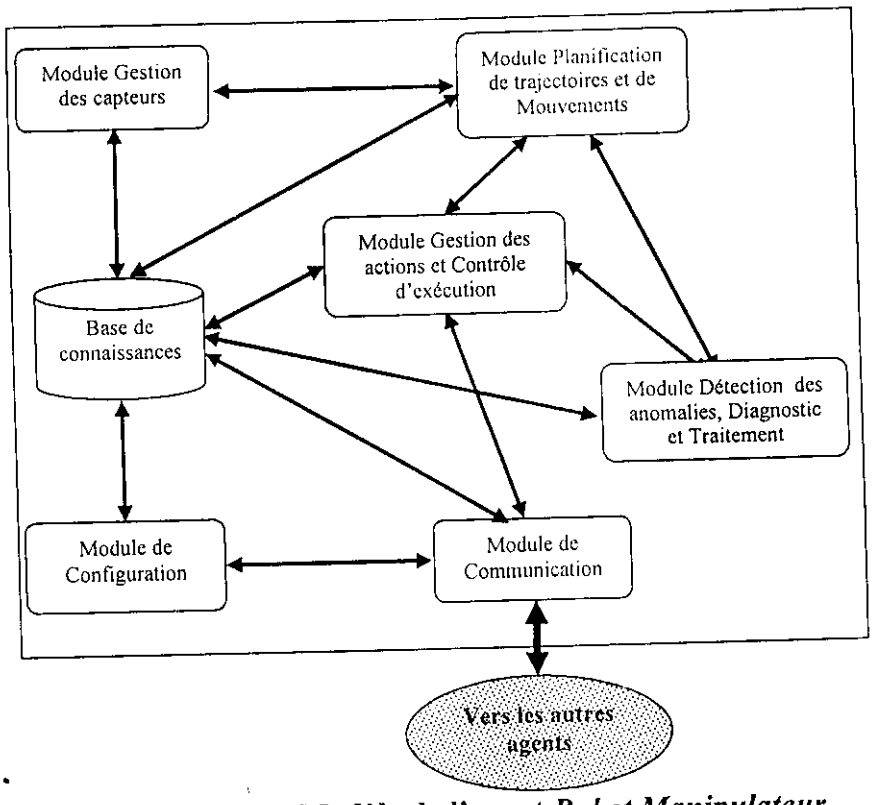


Figure III.7 : Modèle de l'agent *Robot Manipulateur*

III.6. Conclusion

On a présenté dans ce chapitre une architecture de contrôle distribuée basée sur le concept multi-agents dédiée au pilotage du robot manipulateur mobile *RobuTER* du laboratoire. Cette architecture utilise l'agent de type hybride qui permet d'obtenir des comportements adaptés aux différentes situations. Elle privilège la distribution du contrôle et des connaissances et la gestion coordonnée des situations imprévues (pannes inopinées).

Le choix de ce modèle de système multi-agents est justifié d'une part par le caractère générique du modèle de l'agent proposé (indépendant de la nature du robot) et d'autre part, par la possibilité d'intégration de l'ensemble dans un système robotique distribué (cellule flexible du laboratoire).

CHAPITRE IV

ARCHITECTURE DETAILLEE DE CONTROLE/COMMANDE

CHAPITRE IV

ARCHITECTURE DETAILLEE DE CONTROLE/COMMANDE

IV.1. Introduction:

Pour développer une application, il ne faut pas se lancer dans l'écriture du code : il faut d'abord organiser les idées, les document, puis organiser la réalisation en définissant les modules et les étapes de la réalisation. C'est cette démarche antérieure à l'écriture que l'on appelle "modélisation" ; son produit est un "modèle".

Pour modéliser et concevoir notre système on va présenter tous les modules qui le composent.

IV.2. Vue statique du système :

Représenter la structure statique d'un système à l'aide de diagrammes de classes.

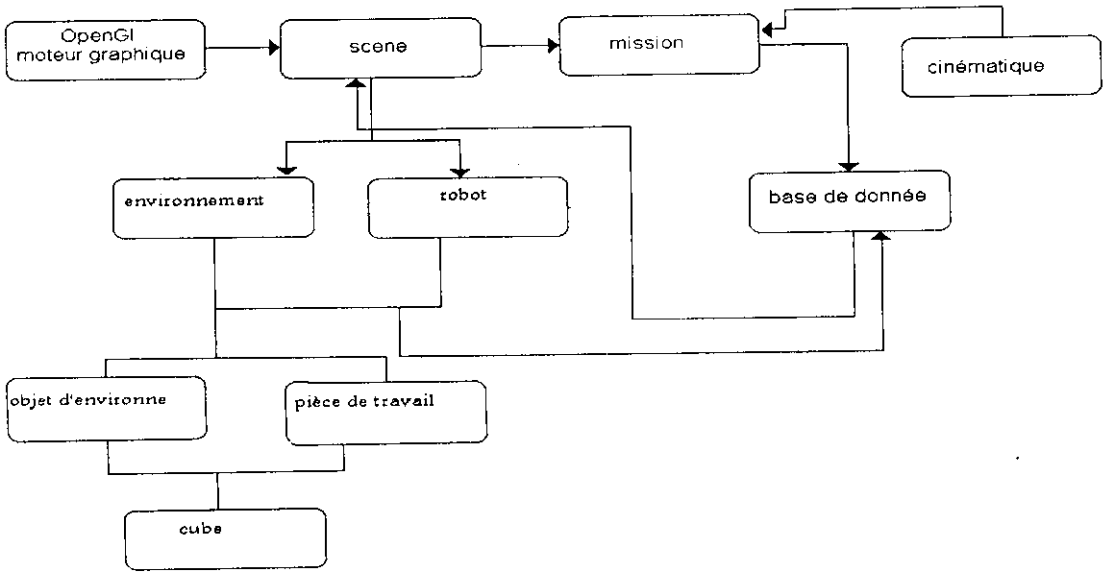
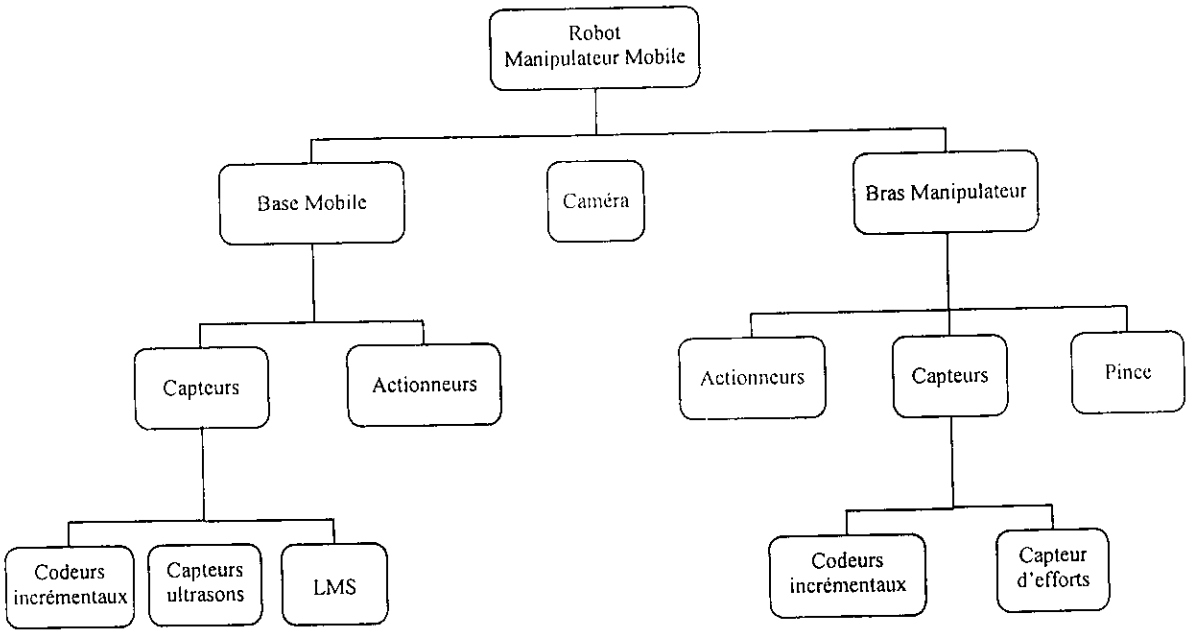


Diagramme d'objet

IV.3 Diagramme de l'architecture matérielle du robot manipulateur mobile



SPECIFICATION DES MESSAGES

IV.4. Protocole de communication

Les agents doivent disposer d'un langage commun pour pouvoir échanger des informations et coopérer pour la résolution d'un problème. Ce langage est un ensemble de primitives connues par chaque entité. Il constitue un protocole de communication qui permet de structurer et d'uniformiser les interactions inter-agents.

Les primitives de base d'un protocole de communication sont les suivants :

- Elaboration d'une connexion entre deux agents
- Identification du destinataire
- Envoi de données
- Réception de données

La communication entre les agents est du type asynchrone et se fait par envoi de messages. Les agents émettent des messages sans se soucier du devenir de ces messages. Ce type de communication permet d'éviter le blocage des agents en attente d'un accusé de réception dans la communication synchrone.

La structure d'un message contient deux types de champs :

- **Entête** : C'est l'ensemble des informations utilisées pour la formalité du message, on distingue :

    - **L'identificateur** : identificateur désignant le message
    - **La priorité:** urgent, normal, ...
    - **L'expéditeur** : identificateur de l'expéditeur
    - **Le destinataire** : identificateur du destinataire
    - **La date d'envoi** : sous le format suivant : *jj mm aaaa hh:mm:ss* ou bien une date locale entre 0 et *n*, ($n \in N$).
    - **La nature** : nature du message (offre, demande, information, ...)
    - **Le nombre de paquets** : indique le nombre total de paquets si l'information à envoyer ne tient pas dans un seul paquet.
    - **Le numéro du paquet** : numéro du paquet courant.

- **Information** : le contenu du message.

IV.5 Spécification des messages

Différents types de messages sont échangés à travers le système. Le message peut être une requête, une information, une suggestion, etc.

IV.5.1 Messages requête

Le message de type requête permet d'avoir des informations sur les agents accointances. On peut citer :

- *Les messages de coopération* : appel d'offre, proposition, contrat, acquittement, fin d'une tâche, échec d'ordonnancement d'une tâche, etc.
- *Les messages de demande d'informations* : ces messages sont utilisés par les agents pour obtenir des informations. Par exemple : opérations réalisables par ses ressources, outils et ressources disponibles, état d'exécution d'une opération ou d'une tâches, etc.

IV.5.2. Messages d'information

Les messages d'information sont utilisés pour être diffusées sur les autres agents. Un agent envoie un message d'information à ses accointances pour répondre à une requête ou soit parce qu'il le juge nécessaire. Par exemple, un agent informe ses accointances que la ressource qu'il contrôle est en panne ou en état dégradé.

IV.6 Mécanisme d'arbitrage

Un agent recevant deux messages de même nature donc de même priorité en provenance de deux agents distincts, la priorité est attribuée selon le contexte dans lequel se trouve chacun de ces deux agents.

Exemple

- **Agent1** : échec d'une opération d'assemblage de deux pièces
- **Agent2** : échec d'une opération de transport d'une troisième pièce pour qu'elle soit assemblée avec les deux autres.

Dans ce cas, la priorité est attribuée à l'agent *Agent1* parce qu'il doit assembler les deux pièces avant de les assembler avec la troisième.

IV.7. Modèle détaillé de l'architecture logicielle

IV.7.1. Agent *Superviseur*

Il est nécessaire d'utiliser un agent *Superviseur* dont le rôle est d'interpréter la mission introduite par l'opérateur ou par le *Superviseur global* de la cellule en un plan de tâches et de distribuer ces dernières sur tous les deux agents constituant le système. Chaque plan est ensuite transmis à l'agent correspondant pour être exécuté.

La base de connaissances de l'agent décrit la configuration matérielle du robot et contient l'ensemble ordonnancé des tâches d'une mission à exécuter. Par ailleurs, elle décrit les engagements de cet agent pour le lancement des tâches à exécuter.

Comme nous pouvons le constater aussi, les compétences de cet agent (ses capacités comportementales et décisionnelles sur la résolution des problèmes auxquels est confronté) sont les suivantes :

- *Module de Configuration*
- *Module Gestion des missions*
- *Module Gestion des tâches et Contrôle d'exécution*
- *Module de communication haut niveau*
- *Module de communication bas niveau*

Le modèle d'architecture de cet agent est donné par la figure 1.

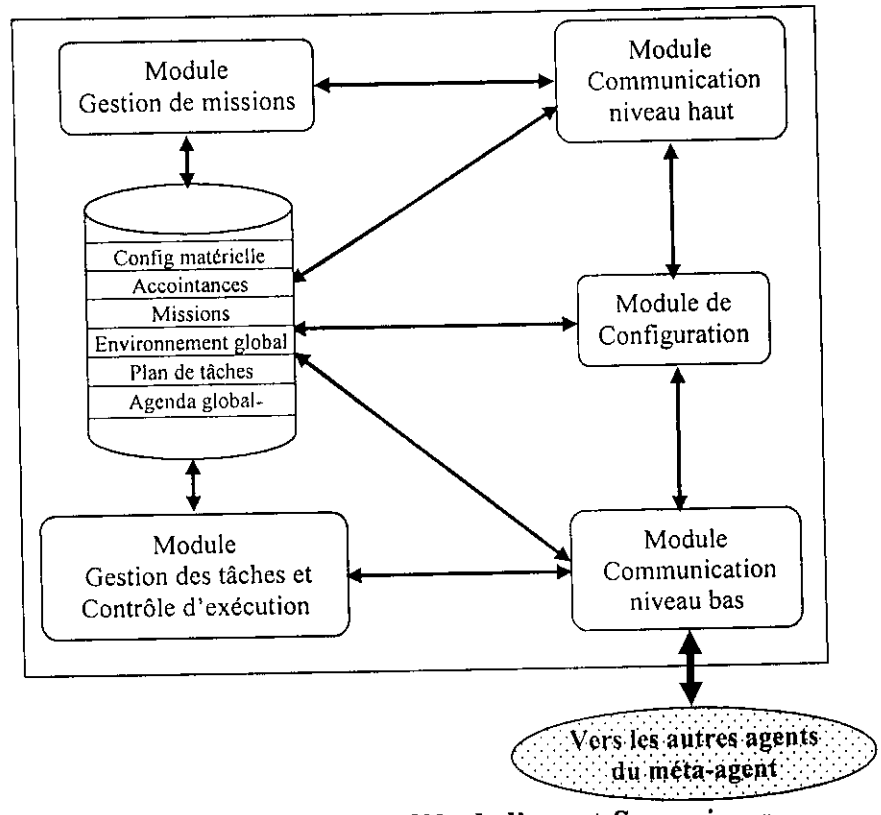


Figure IV.4 : Modèle de l'agent *Superviseur*

Dans les paragraphes suivants nous allons décrire en détail les différents modules constituants cet agent.

IV.7.1.1. Module Gestion des missions

Après avoir reçu la mission confiée au robot manipulateur mobile à travers le module *Communication haut niveau*, elle sera envoyée au module *Gestion des missions*..Ce dernier doit statuer sur l'acceptation ou le refus de la mission. Pour cela, il doit vérifier la disponibilité et l'état de toutes les ressources que dispose le robot. Si c'est le cas, il l'interprète en plan de

tâches ordonnancées et les distribue sur les agents concernés. Il peut aussi générer plusieurs plans de tâches et les stocker dans sa base de connaissances pour, éventuellement, faire un choix par la suite. Dans le cas contraire, il informe l'agent *Superviseur global* de la cellule ou l'opérateur de son incapacité à accomplir la mission.

Le module *Gestion des missions*, donné par la figure 2, est constitué des tâches suivantes :

- *Décision* : Cette tâche décide de la faisabilité de la mission confiée au robot manipulateur mobile. Si la mission ne peut pas être exécutée, la tâche *Décision* en informe le *Superviseur global* de la cellule ou l'opérateur. Sinon, elle active la tâche *Ordonnancement*.
- *Ordonnancement* : Cette tâche permet de choisir un des plans d'exécution stockés dans la base de connaissances ou de générer un nouveau plan et de le distribuer sur les agents concernés.
- *Négociation* : Dans le cas où la génération du plan de tâches nécessite une négociation avec les autres agents du système multi-agents, la tâche *Ordonnancement* fait appel à la tâche *Négociation*.
- *Echange d'informations* : Elle a en charge l'élaboration des messages à envoyer et la vérification de l'intégrité des messages reçus. L'échange des messages s'effectue au niveau de la même entité, c.-à-d. le même méta-agent.

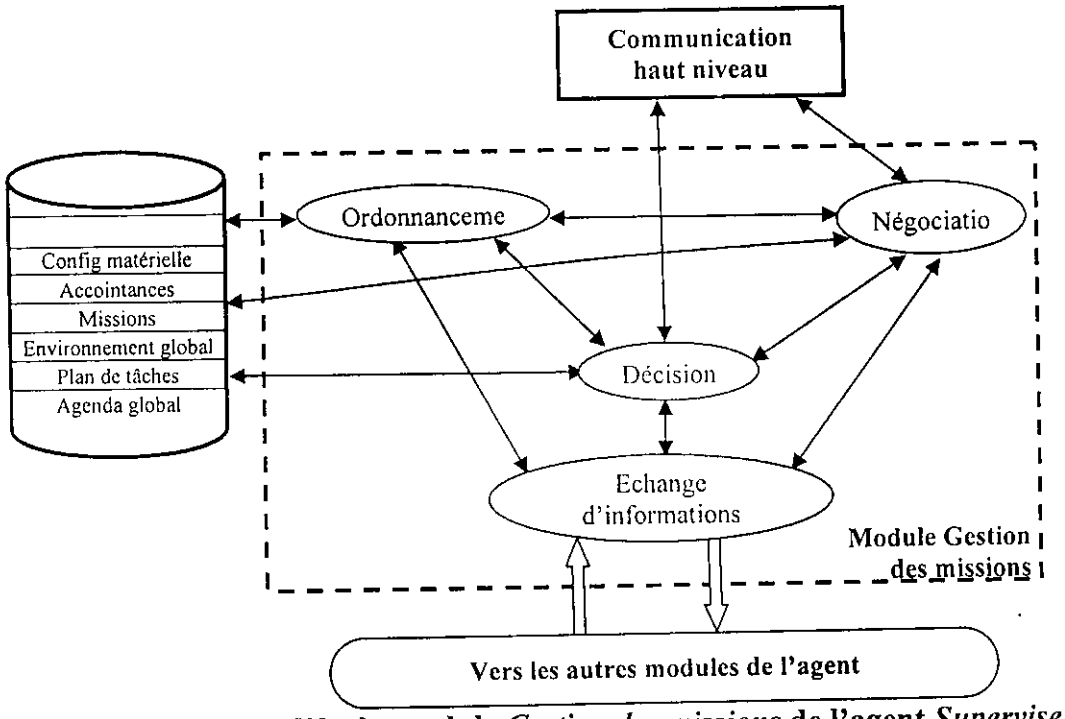


Figure IV.5 : Modèle du module *Gestion des missions* de l'agent *Superviseur*

IV.7.1.2. Module Communication bas niveau

C'est un ensemble de tâches concurrentes (temps réel) dont la fonction globale est la capture des messages et des événements issus de l'extérieur (autres agents) et de l'intérieur du méta-agent, leurs interprétation, leurs traitements et la prise de décision d'une part, et d'autre part la mise en forme, le codage et la transmission des messages à envoyer.
La décision peut s'obtenir de deux façons :

- **Au niveau du module Communication bas niveau** : dans ce cas c'est la tâche *Constructeur de messages* constituant ce module qui doit gérer l'émission du message de retour (réponse au message). La transmission effective de ce message de retour est assurée par la tâche *Emission.*
- **Au niveau du module Gestion des tâches et contrôle d'exécution** : ce dernier, selon le contexte, se charge de la prise de décision qu'il faut prendre et en avertir le module *Communication bas niveau* pour s'occuper de la transmission de la partie du message destinée aux autres agents.
  Ce type de situation s'impose pour une catégorie bien déterminée de messages et événements dont le traitement nécessite un degré d'intelligence élevé et, par conséquent, la prise de décision est confiée au module *Gestion des tâches et contrôle d'exécution.*

Le module *Communication bas niveau* est constitué des tâches suivantes :

- *Réception* : À chaque fois qu'un message (ou un événement) issu des autres agents du méta-agent est disponible, la tâche lance un *thread* qui se charge de lire le message et le mettre dans la table des messages reçus (la *BALR*). Cette table est accessible par une seule tâche de l'agent qui est la tâche *Interprétation.*
- *Interprétation* : C'est la tâche qui effectue le décodage et l'interprétation des messages (ou des événements) contenus dans la *BALR.* A chaque type de message, cette tâche effectue un traitement qui correspond à la nature du message. Les différents types de messages possibles sont :

  - *Configuration* : Lors de la configuration d'un autre agent du méta-agent, un message de configuration est envoyé au *Superviseur* pour la mise à jour de sa

base de connaissances. La tâche *Interprétation* l'envoie ensuite à la tâche *Constructeur de messages*.

- *Suite à une proposition* : Avant que l'agent *Superviseur* envoie une proposition à un méta-agent ou au *superviseur global* de la cellule, il doit consulter tout d'abord les deux agents qu'il contrôle (l'agent *RobotMobile* et l'agent *RobotManipulateur*). La tâche *Interprétation* l'envoie à la tâche *Constructeur de messages* par la suite.
- *Acceptation* : C'est un message d'acceptation de contrat. La tâche *Interprétation* l'envoie à la tâche *Constructeur de messages* pour la mise à jour de la base de connaissances.
- *Refus* : C'est un message de refus de contrat. La tâche *Interprétation* l'envoie à la tâche *Constructeur de messages* pour l'envoi d'une autre demande de contrat.
- *Annulation* : C'est un message d'annulation de contrat. La tâche *Interprétation* l'envoie à la tâche *Constructeur de messages* pour supprimer les informations relatives au contrat annulé.

- *Constructeur de messages* : Le module *Communication bas niveau* de l'agent *Superviseur* est doté d'une intelligence qui réside au niveau de la tâche *Constructeur de messages*. Cette tâche est une interface entre la tâche *Emission* et les différents modules et tâches impliqués dans l'émission des messages. Elle joue le double rôle en émission et en réception. Pour l'émission, elle élabore les informations constituants le message à envoyer à l'agent émetteur. Quand à la réception, elle procède à la répartition et la diffusion des informations partielles ou complètes des messages reçus. A chaque type de messages ou événement reçu est associé un traitement dont nous expliquerons le principe dans ce qui suit :

  - *Configuration* : La tâche envoie une requête de mise à jour de la base de connaissances (table configuration matérielle de l'agent correspondant) et au module *Configuration* pour la mise à jour de la base de connaissances de l'agent *Superviseur global* de la cellule.
  - *Proposition* : La tâche envoie une requête de mise à jour de la base de connaissances (table d'ordonnancement : une table allouée en mémoire qui contient toutes les informations relatives aux tâches lancées) de l'agent

Superviseur. Elle pointe aussi le même message vers le module *Gestion des tâches et Contrôle d'exécution* pour collecter toutes les propositions relatives à la tâche à exécuter, sélectionner la meilleure proposition et envoyer une demande de contrat à l'agent concerné.

- *Acceptation* : La tâche *Constructeur de messages* effectue une requête de stockage de la tâche concernée dans le plan de production, pour être exécuté à la date de début spécifiée dans le contrat. Cette requête est envoyée au module *Gestion des tâches et Contrôle d'exécution.*
- *Refus* : La tâche *Constructeur de messages* envoie une requête au module *Gestion des tâches et Contrôle d'exécution* pour une nouvelle évaluation de toutes les propositions.
- *Annulation* : La tâche *Constructeur de messages* envoie au module *Gestion des tâches et contrôle d'exécution* une requête de suppression des informations relatives au contrat annulé.

- *Emission* : Quand un module de l'agent *Superviseur* veut envoyer un message (ou un événement) aux agents du méta-agent, il le dépose dans la table des messages à envoyer. L'accès à cette table est synchronisé entre les différents modules de l'agent. Dés la présence d'un message dans la *BALE*, la tâche lance un *thread* qui se charge d'effectuer la mise en forme, le codage et la transmission effective des messages et des événements.

Le schéma d'interaction entre les différentes tâches constituant le module *Communication bas niveau* est montré ci-dessous :

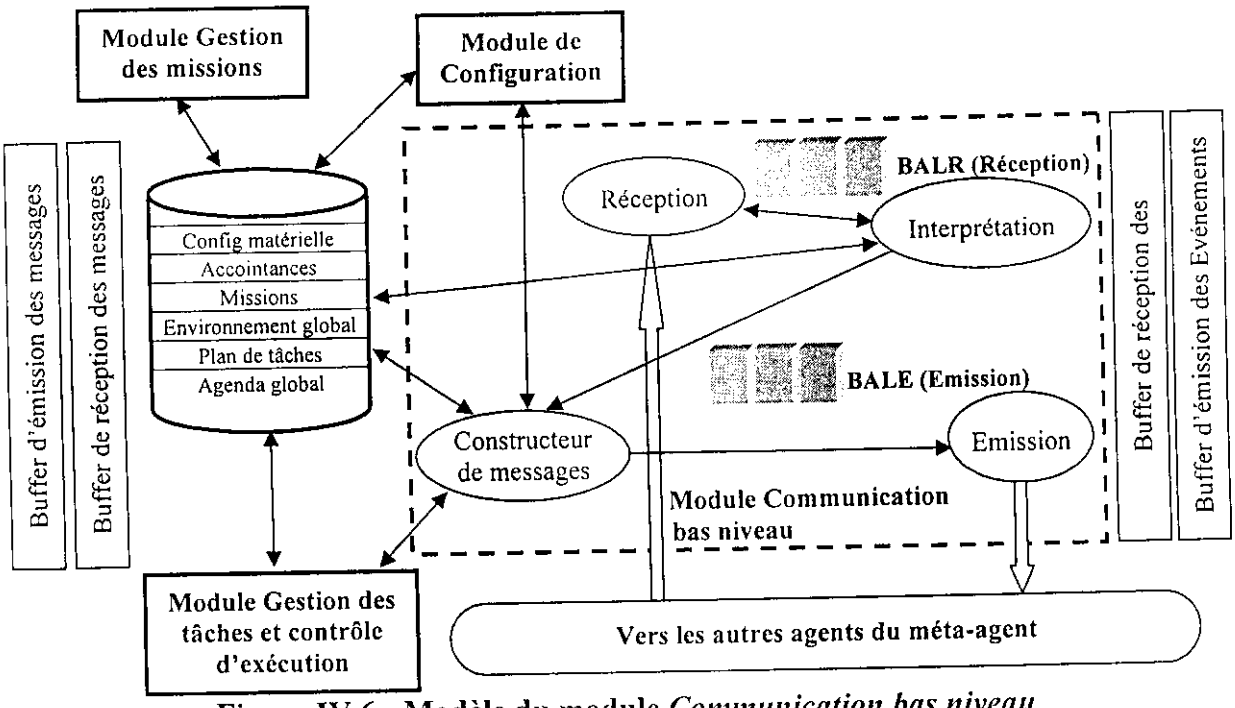


Figure IV.6 : Modèle du module *Communication bas niveau*

IV.7.1.3. Inventaire des messages et des événements

Lors du pilotage du robot manipulateur mobile, différents messages et événements peuvent être échangés entre le module *Communication bas niveau* de l'agent *Superviseur* et les autres agents du méta-agent. On distingue des messages de configuration et reconfiguration, des messages de négociation pour l'ordonnancement des tâches (offre, proposition, contrat, acquittement), des messages de suivi de mission, des messages pour la surveillance des différentes ressources du robot, etc. Parmi ces messages et événements, on peut citer :

IV.7.1.3.1 Messages

- Echec opération
- Succès d'une opération
- Envoi d'une tâche
- Proposition
- Contre proposition
- Etablissement de contrat
- Configuration des agents

- Mise à jour de la base de connaissances
- Archivage de missions
- Demande de la liste des ressources du robot
- Demande de la liste des opérations des ressources du robot
- Demande de la liste des outils des ressources du robot
- Demande de la liste des opérations d'une ressource
- Demande de la liste des outils d'une ressource
- Echec d'ordonnancement d'une tâche
- Fin d'une tâche

IV.7.1.3.2 Evénements

- Ajout d'une ressource
- Suppression d'une ressource
- Changement d'état d'opération
- Acquittement
- Appel d'offre
- Echec d'une opération
- Succès d'une opération
- Expiration de délai
- Début d'exécution d'une opération
- Début d'exécution d'une tâche
- Fin d'exécution d'une opération
- Fin d'exécution d'une tâche
- Changement d'un plan d'opérations
- Acceptation de proposition
- Refus de proposition
- Contrat
- Arrêt d'urgence

IV.7.1.3.3. Messages liés aux événements

- Appel d'offre
    - Appel d'offre
- Echec d'opération
    - Echec d'opération
- Changement de plan d'une tâche
    - Nouveau plan
- Acceptation de proposition

- Contrat
    - Contrat
- Demande de la liste des ressources du robot
    - Liste des ressources du robot
- Demande de la liste des opérations des ressources du robot
    - Liste des opérations des ressources du robot
- Demande de la liste des outils des ressources du robot
    - Liste des outils des ressources du robot
- Demande de la liste des opérations d'une ressource
    - Liste des opérations d'une ressource
- Demande de la liste des outils d'une ressource
    - Liste des outils d'une ressource
- Echec d'ordonnancement
    - Echec d'ordonnancement
- Refus de proposition
    - Contre proposition
- Fin d'une tâche

- Ressource en panne
    - Ressource en panne

IV.7.1.3.4. Messages non liés aux événements

- Configuration des agents

IV.7.1.3.5. Evénements non liés aux messages

- Succès d'une opération
- Suppression d'une ressource
- Changement d'état d'une opération
- Acquittement
- Expiration de délai
- Début d'exécution d'une opération
- Début d'exécution d'une tâche
- Fin d'exécution d'une opération
- Fin d'exécution d'une tâche
- Arrêt d'urgence

IV.7.1.3.5. Evénements liés aux événements

- **Contrat** : soit
  - Acceptation de contrat, ou
  - Refus de contrat

IV.7.1.4 Gestion des priorités

IV.7.1.4.1. Priorités des événements et des messages

Messages et événements	Très haute priorité	Haute priorité	Priorité normale	Priorité basse
Arrêt d'urgence	X			
Echec opération		X		
Succès d'une opération			X	
Envoi d'une tâche				
Proposition		X		
Contre proposition		X		
Etablissement de contrat				
Configuration des agents		X		
Mise à jour de la base de connaissances			X	

Archivage de missions			X	
Demande de la liste des ressources du robot				X
Demande de la liste des opérations des ressources du robot				X
Demande de la liste des outils des ressources du robot				X
Demande de la liste des opérations d'une ressource				X
Demande de la liste des outils d'une ressource				X
Echec d'ordonnancement d'une tâche		X		
Fin d'une tâche			X	
Ajout d'une ressource			X	
Suppression d'une ressource		X		
Changement d'état d'opération		X		
Acquittement		X		
Appel d'offre		X		
Echec d'une opération		X		
Succès d'une opération			X	
Expiration de délai			X	
Début d'exécution d'une opération			X	
Début d'exécution d'une tâche			X	
Fin d'exécution d'une opération		X		
Fin d'exécution d'une tâche		X		
Changement d'un plan d'opération		X		
Acceptation de proposition		X		
Refus de proposition			X	
Evénement de contrat		X		

IV.7.1.5. Module Communication haut niveau

Ce module assure l'échange d'informations entre l'agent *Superviseur* du méta-agent et les autres méta-agents ainsi que l'agent *Superviseur* global de la cellule (qui ne fait pas partie de cette étude).
Le module *Communication haut niveau* est constitué des tâches suivantes :

- *Réception* : À chaque fois qu'un message ou un événement issu d'un méta-agent ou de l'agent *Superviseur* global de la cellule est présent sur le réseau, la tâche lance un *thread* qui se charge de lire le message et le mettre dans la table des messages reçus (la BALR) qui n'est accessible que par une seule tâche : la tâche *Interprétation.*
- *Interprétation* : C'est la tâche qui effectue le décodage et l'interprétation des messages ou des événements contenus dans la *BALR*. Les différents types de messages en réception, possibles sont :
    - *Appel d'Offre* : A la réception d'un appel d'offre, l'agent *Superviseur* du méta-agent envoie une proposition à l'agent émetteur selon la disponibilité des ressources du robot manipulateur mobile qu'il contrôle.
    - *Contrat* : C'est un message de demande de contrat. L'agent *Superviseur* répond par une acceptation en cas où il n y a pas de conflit. Sinon, il refuse le contrat.
    - *Annulation* : C'est un message d'annulation de contrat pour l'exécution de la mission. La tâche *Interprétation* l'envoie à la tâche *Constructeur de messages* pour supprimer les informations relatives au contrat annulé.

- *Constructeur de messages* : Le module *Communication haut niveau*, comme celui du bas niveau, est doté d'une intelligence au niveau de la tâche *Constructeur de messages*. A chaque type de messages ou événement reçu est associé un traitement dont nous expliquons le principe dans ce qui suit :

    - *Configuration* : Lors de la configuration d'un agent quelconque du méta-agent, un message de configuration est envoyé au *Superviseur* pour la mise à jour de sa base de connaissances. Ces agents doivent être aussi configurés au niveau de l'agent *Superviseur* global de la cellule.

- *Proposition* : La tâche envoie la proposition à l'agent émetteur pour sélectionner la meilleure offre.
- *Acceptation* : La tâche *Constructeur de messages* de l'agent *Superviseur* active le module *Gestion des missions* pour le stockage de la mission.
- *Refus* : La tâche *Constructeur de messages* de l'agent *Superviseur* informe le module *Gestion des missions* du message de refus de contrat envoyé.
- *Annulation* : C'est un message d'annulation de contrat. La tâche *Constructeur de messages* lance une requête de suppression des informations relatives au contrat annulé.

- *Emission* : Quand un module de l'agent *Superviseur* veut envoyer un message ou un événement à un agent de la cellule, il le dépose dans la table des messages à envoyer. Dés la présence d'un message dans la *BALE*, la tâche lance un *thread* qui se charge d'effectuer la mise en trame, le codage et la transmission effective du message à l'agent *Superviseur global*.

Le schéma d'interaction entre les différentes tâches constituant le module *Communication haut niveau* est montré par la figure ci-dessous :

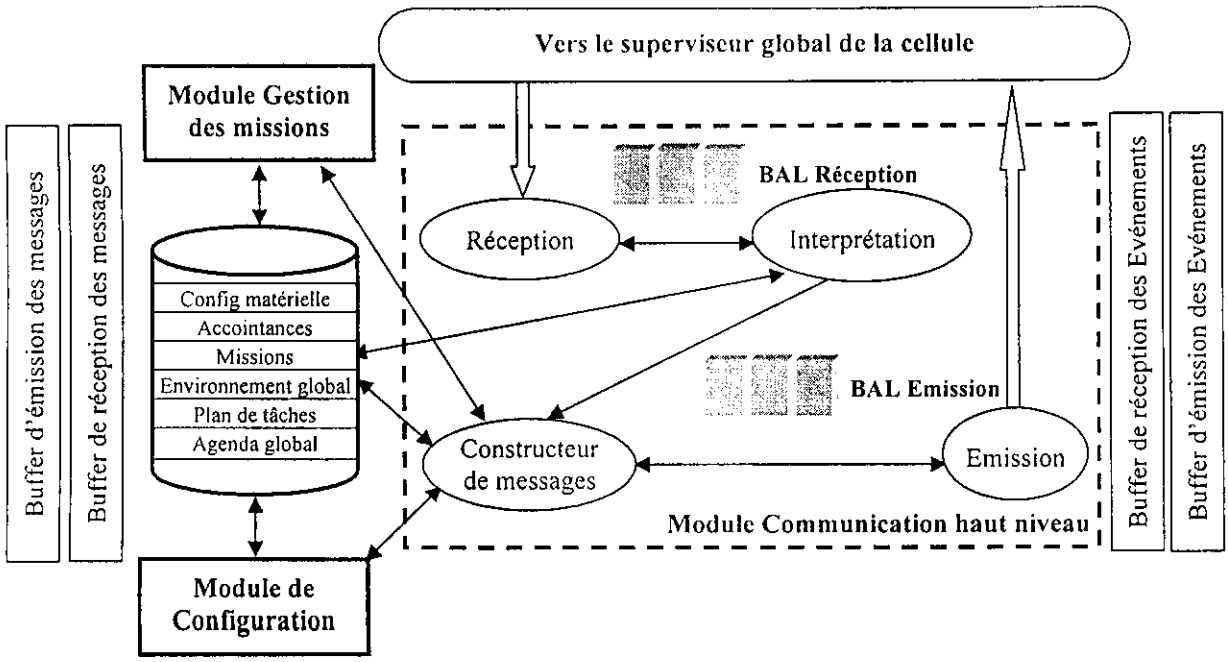


Figure IV.7 : Modèle du module *Communication haut niveau* de l'agent *Superviseur*

IV.7.1.6 Inventaire des messages et des événements

Lors du pilotage du robot manipulateur mobile, différents messages et événements peuvent être échangés entre le module *Communication haut niveau* de l'agent *Superviseur* et les autres agents (les autres méta-agents et le superviseur global de la cellule). Les messages et les événements envoyés et reçus par ce module sont les suivants :

- Mission à exécuter
- Plan de mission
- Contrat
- Acceptation
- Refus
- Annulation
- Arrêt d'urgence
- Demande du plan de mission
- Nouveau plan
- Fin d'exécution d'une mission avec succès
- Fin avec échec
- Proposition
- Configuration

Messages et événements	**Très haute priorité**	**Haute priorité**	**Priorité normale**	**Priorité basse**
Arrêt d'urgence	X			
Mission à exécuter		X		
Plan de mission		X		
Contrat		X		
Acceptation			X	
Refus			X	
Annulation		X		
Proposition			X	
Configuration		X		

Demande du plan de mission		X		
Nouveau plan		X		
Fin d'exécution d'une mission avec succès			X	
Fin avec échec			X	

IV.7.1.7 Module Gestion des tâches et Contrôle d'exécution

En plus de la tâche *Echange d'informations*, le module *Gestion des tâches et Contrôle d'exécution*, montré par la figure 5, est composé des tâches suivantes :

- *Exécution* : Cette tâche sert principalement au lancement des tâches sous contrat. En scrutant continuellement son *Horloge*, la tâche *Exécution* lance la tâche figurant dans le calendrier et ayant une date de début identique à l'horloge. Il envoie pour cela un ordre d'exécution à l'agent correspondant. Cependant, aucune tâche n'est lancée si le résultat du pré-contrôle n'est pas satisfaisant.
- *Ordonnancement* : Cette tâche permet de choisir la meilleure façon de réaliser le plan de tâches affecté à l'agent, c.-à-d. le meilleur séquencement des tâches pour réaliser ses objectifs suivant un critère qui est le plus souvent temporel (réalisation la plus rapide du plan de tâches). Deux situations peuvent alors se présenter. Dans la première, un ordonnancement est élaboré localement par le module *Ordonnancement* sans solliciter d'autres agents. Dans la seconde, cette tâche fait appel à la tâche *Négociation* pour un ordonnancement coordonné.
- *Négociation* : Le rôle de cette tâche est d'étendre le raisonnement de la tâche *Ordonnancement* pour aboutir à un ordonnancement coordonné entre les différents agents du système multi-agents.

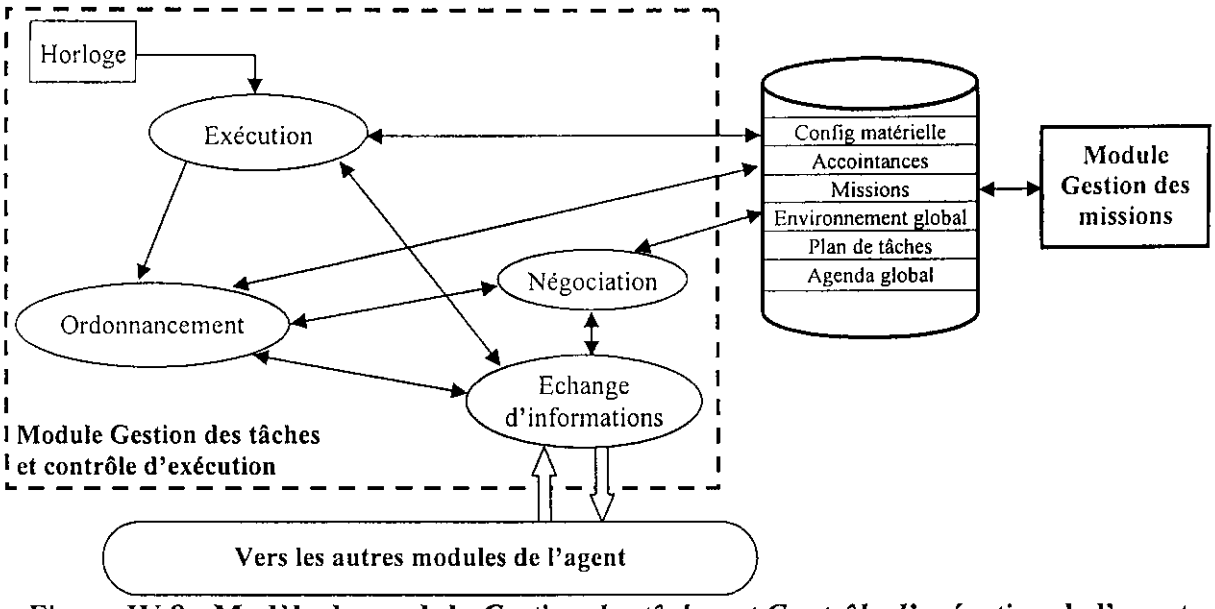


Figure IV.8 : Modèle du module *Gestion des tâches et Contrôle d'exécution* de l'agent *Superviseur*

IV.7.1.8 Module de Configuration

Pour la configuration de l'agent *Superviseur*, l'opérateur doit saisir toutes les informations concernant les connaissances de l'agent. Par la suite, ce dernier se configure automatiquement au niveau de l'agent *Superviseur global* de la cellule et de toutes ses accointances. Il reçoit, en échange, les connaissances partielles sur les autres agents pour devenir une partie du système multi-agents.

Le module de *Configuration*, montré par la figure 6, se compose en plus de la tâche *Echange d'informations*, des tâches suivantes :

- *Accointances* : Cette tâche permet de configurer tous les accointances de l'agent *Superviseur*.
- *Environnement* : Cette tâche permet de donner la configuration initiale de l'environnement (espace de travail du robot manipulateur mobile) et effectuer un rafraîchissement dynamique de la base de connaissances dés qu'un changement de l'environnement survienne.
- *Missions* : Permet d'introduire la liste de toutes les missions que le robot manipulateur mobile peut effectuer.

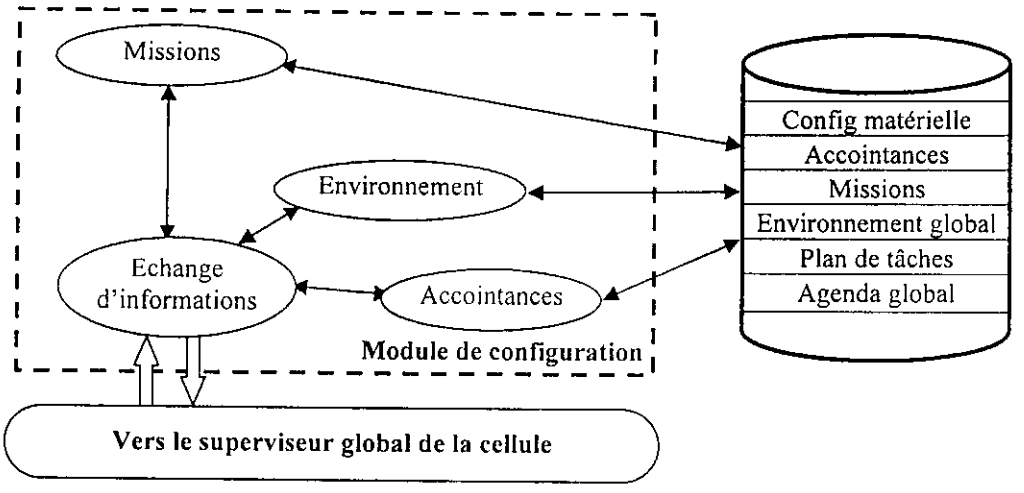


Figure 6 : Modèle du module de *Configuration* de l'agent *Superviseur*

IV.7.2 Agent *RobotMobile/RobotManipulateur*

L'architecture globale de l'agent *RobotMobile* (resp. *RobotManipulateur*) est similaire à celle de l'agent *Superviseur* à l'exception de quelques modules. Cet agent assure la gestion locale de la base mobile (resp. le bras manipulateur) du robot en intégrant les fonctions de contrôle temps réel, à savoir : la planification, l'exécution, la surveillance et la communication.

L'architecture proposée pour cet agent confère à la plateforme mobile (resp. le bras manipulateur), à la fois, des capacités réactives et des capacités délibératives pour raisonner sur des situations complexes :

- L'agent doit adopter, suivant sa situation, plusieurs comportements. Il doit mettre en œuvre des mécanismes de raisonnement intelligents sur ses actions et sur son plan de tâches.
- L'agent doit contrôler de manière autonome l'ensemble de ses actions en fonction de son état interne et de son environnement. Il doit donc connaître à tout moment son état d'exécution qui est lié au contexte de son environnement (perturbations dans le plan de tâches, présence d'obstacles) et s'y adapter en élaborant un autre plan.

L'agent *RobotMobile* (resp. *RobotManipulateur*) doit disposer, en outre de la base de connaissances et du *module Communication*, des compétences suivantes :

montré par la figure 8, se compose, en plus de la tâche *Echange d'informations* et des deux tâches *Accointances* et *Environnement*, des tâches suivantes :

- *Equipements* : Cette tâche permet de rajouter des équipements pour qu'ils soient contrôlés par cet agent. Aussi, pour des besoins de maintenance, on peut être amené à retirer momentanément ou définitivement des équipements de l'agent.
- *Opérations* : Pour chaque équipement contrôlé par cet agent, il faut saisir la liste des opérations qu'il peut effectuer.

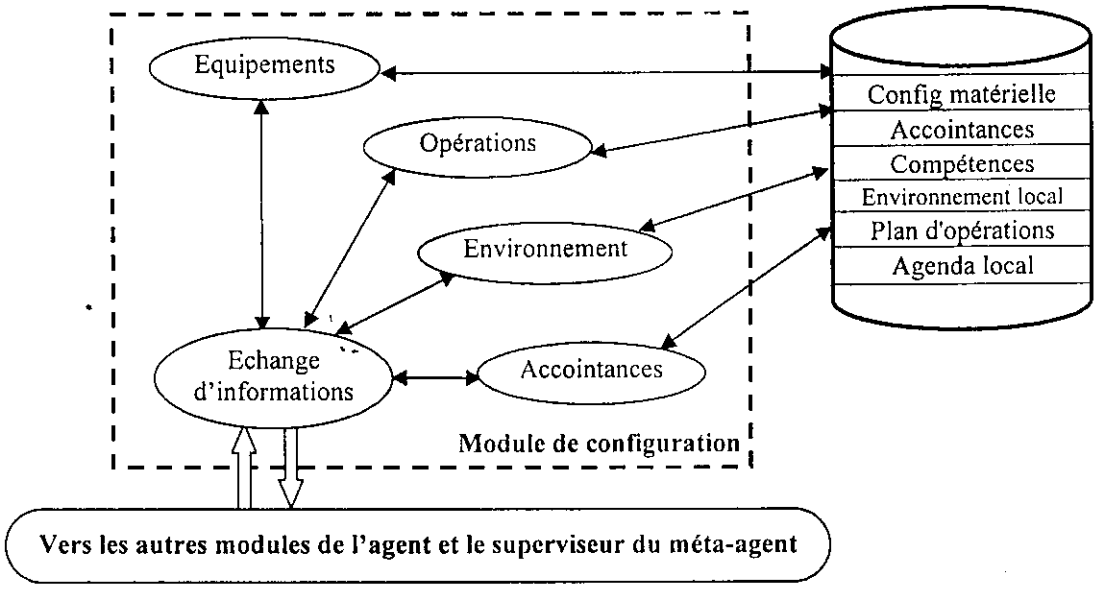


Figure 8 : Modèle du module de *Configuration* de l'agent *RobotMobile* et l'agent *RobotManipulateur*

IV.7.2.2. Module de Communication

Ce module, montré par la figure 9, a la même structure que celui du module de *Communication bas niveau* de l'agent *Superviseur*. La tâche *Interprétation* de ce module interprète les messages suivants :

- *Appel d'Offre* : À la réception d'un appel d'offre, l'agent *RobotMobile/RobotManipulateur* envoie une proposition à l'expéditeur de l'appel d'offre en se basant sur la disponibilité de la ressource qu'il contrôle.

- *Contrat* : C'est un message de demande de contrat. L'agent répond à ce type de message par une acceptation de contrat dans le cas où il n y a pas de conflit. Sinon, il refuse le contrat.
- *Annulation* : C'est un message d'annulation de contrat. L'agent mis à jour sa table d'ordonnancement en supprimant l'entrée de l'action concernée par le message d'annulation.

Les messages envoyés par le module de *Communication* sont les suivants :

- *Acceptation* : C'est un message d'acceptation de contrat. S'il n y a pas de conflit avec une autre tâche, l'agent envoie une acceptation de contrat. Sinon, il envoie un refus de contrat.
- *Refus* : C'est un message de refus de contrat. L'agent mis à jour sa table d'ordonnancement (supprime l'entrée de l'action concernée).

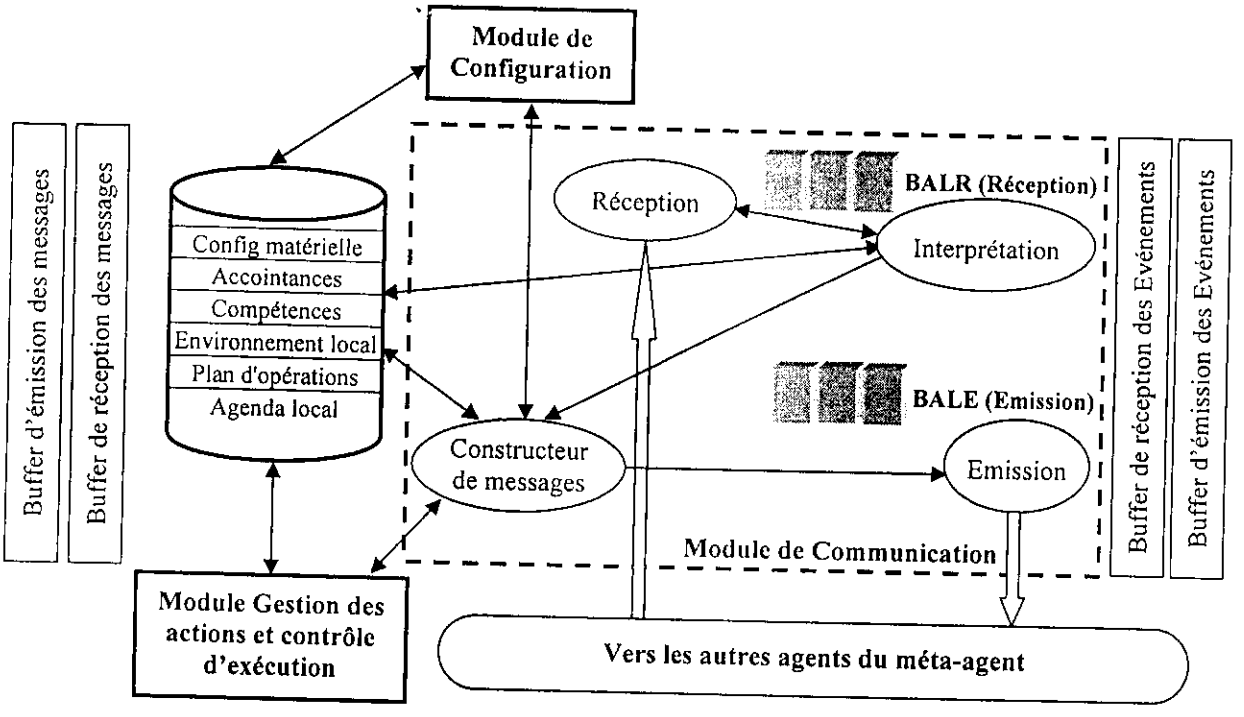


Figure 9 : Modèle du module de Communication de l'agent *RobotMobile* et *RobotManipulateur*

IV.7.2.3. Module Gestion des capteurs

Le module *Gestion des capteurs*, illustré par la figure 10, comporte, en plus de la tâche *Echange d'informations*, les tâches suivantes :

- *Perception* : elle permet de collecter les informations issues des différents capteurs (capteurs odométriques, capteurs ultrasonores, capteur d'efforts, caméra) qui équipent la base mobile (resp. le bras manipulateur) et de leur envoi à la tâche *Prétraitement*.
- *Prétraitement* : prétraitement des données perçues en vu d'en extraire les informations nécessaires au module *Planification de trajectoires et navigation/mouvements*.

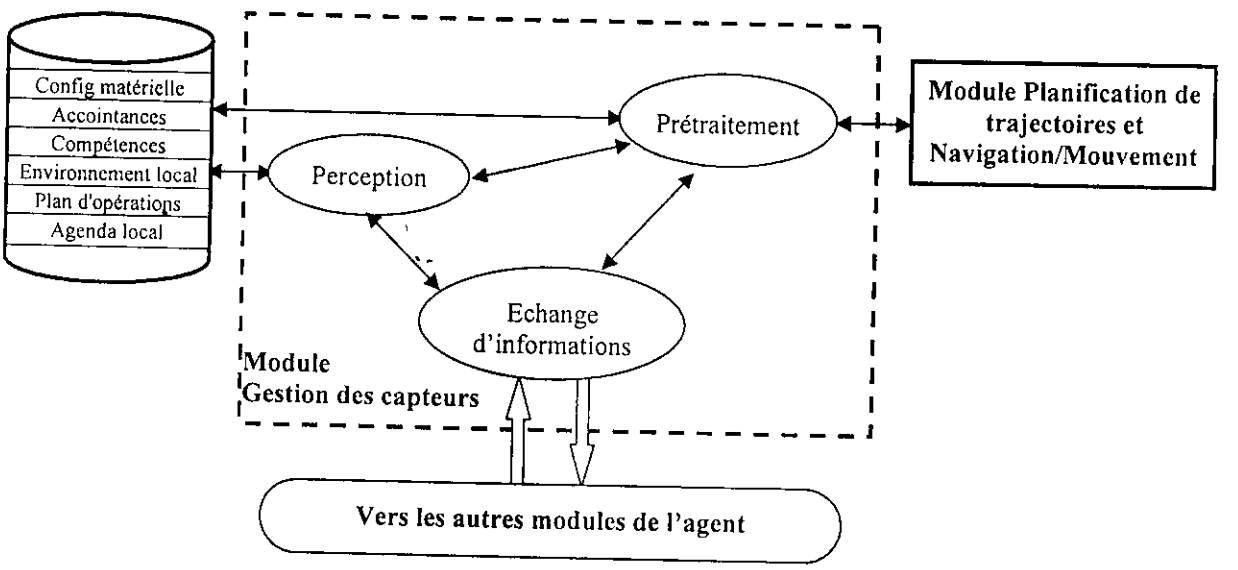


Figure 10 : Modèle du module *Gestion des capteurs* de l'agent *RobotMobile* et l'agent *RobotManipulateur*

IV.7.2.4. Module Planification de trajectoires et Navigation/Mouvements

La structure du module *Planification de trajectoires et Navigation/Mouvements*, illustré par la figure 11, s'articule autour des deux tâches suivantes ainsi que la tâche *Echange d'informations* :

- *Génération de trajectoires/Mouvements* : constitue un générateur de plans d'opérations en réponses aux ordres reçus du module *Gestion des actions et Contrôle d'exécution* et après consultation de la base de connaissances. Le plan fourni est une séquence

d'opérations qui effectue soit une planification ou une replanification en cas de perturbation ou d'une détection d'une anomalie.

- *Evitement d'obstacles et suivi* : calcule continuellement les consignes à envoyer au module *Gestion des actions et Contrôle d'exécution* par la mise en œuvre des algorithmes d'évitements d'obstacles, de suivi de murs, de suivi de cibles, etc. (c.-à-d. le comportement réactif du robot). Cette tâche utilise soit les techniques de la logique floue ou celles des réseaux de neurones en se basant sur les informations reçues du module *Gestion des capteurs*. Elle informe aussi le module *Gestion des actions et Contrôle d'exécution* sur l'état d'exécution des trajectoires (succès ou échec).

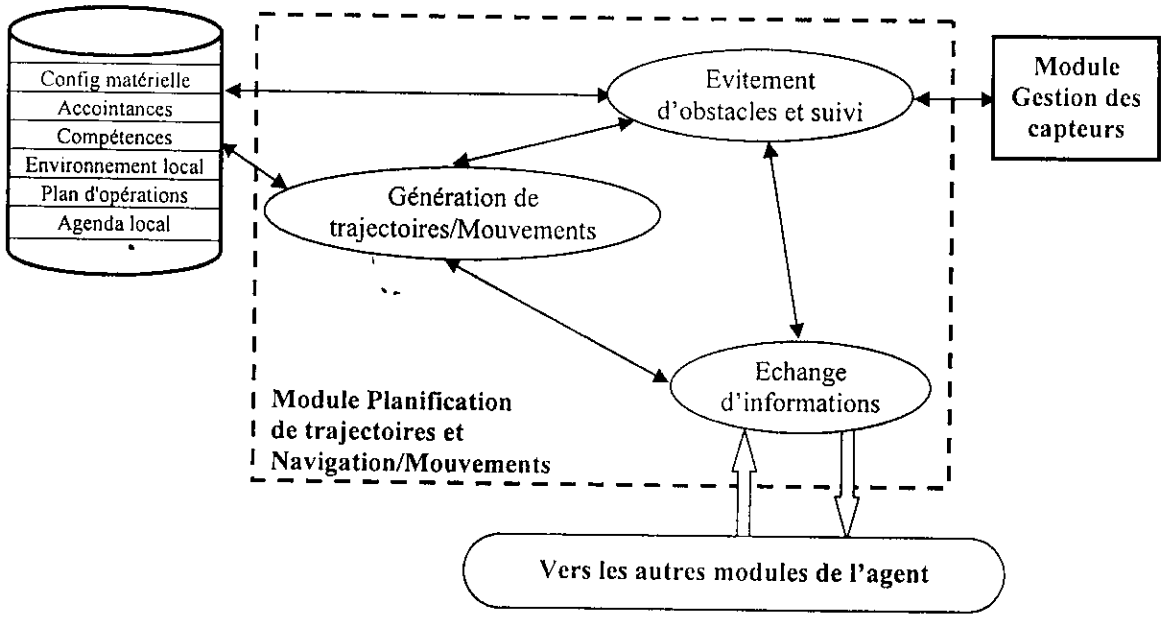


Figure 11 : Modèle du module *Planification de trajectoires et Navigation/Mouvements* de l'agent *RobotMobile* et l'agent *RobotManipulateur*

IV.7.2.5. Module Surveillance

Le module *Surveillance*, illustré par la figure 12, est composé, en plus de la tâche *Echange d'informations*, des tâches suivantes :

- *Suivi* : Le suivi de l'exécution de tâches est effectué avant (pré-contrôle), pendant et après (post-contrôle) l'exécution de chaque opération élémentaire. A la fin d'exécution d'une opération, la tâche *Détection des anomalies* est activée.

- *Détection des anomalies* : Cette tâche se charge de reconnaître les situations qui correspondent au fonctionnement anormal de l'agent (n'est pas conforme à la situation de référence : une pièce tombe au cours de son déplacement par le bras manipulateur, retard dû à la présence d'obstacles sur la trajectoire de référence, etc.). Toute déviation décrit l'ensemble des symptômes d'un dysfonctionnement. Dans le cas où une anomalie est détectée, la tâche *Diagnostic* s'active.
- *Diagnostic* : À la suite d'une détection d'anomalie ou d'une perturbation, cette tâche procède au formatage des faits, leurs chargements dans la base des faits (base de connaissances) et le lancement de l'inférence. Par la suite, elle envoie les résultats obtenus au module *Gestion des actions et Contrôle d'exécution* pour le traitement de l'anomalie dans le cas où la reprise nécessite un ordonnancement. Sinon, elle active la tâche *Reprise*.

  Chaque agent du système dispose d'une expertise de pannes pouvant affecter ses ressources. Cette expertise est constituée d'un ensemble de règles de diagnostic qui servent à identifier les causes des anomalies pouvant survenir lors du fonctionnement de la ressource.
- *Reprise* : Si la reprise ne nécessite pas un processus de réordonnancement qui est confié au module *Gestion des actions et Contrôle d'exécution* elle sera traitée localement au niveau de cette tâche.

  Exemple : contournement d'un obstacle imprévu dont le temps nécessaire est estimé inférieur aux limites temporelles fixées dans le plan d'opérations initial.

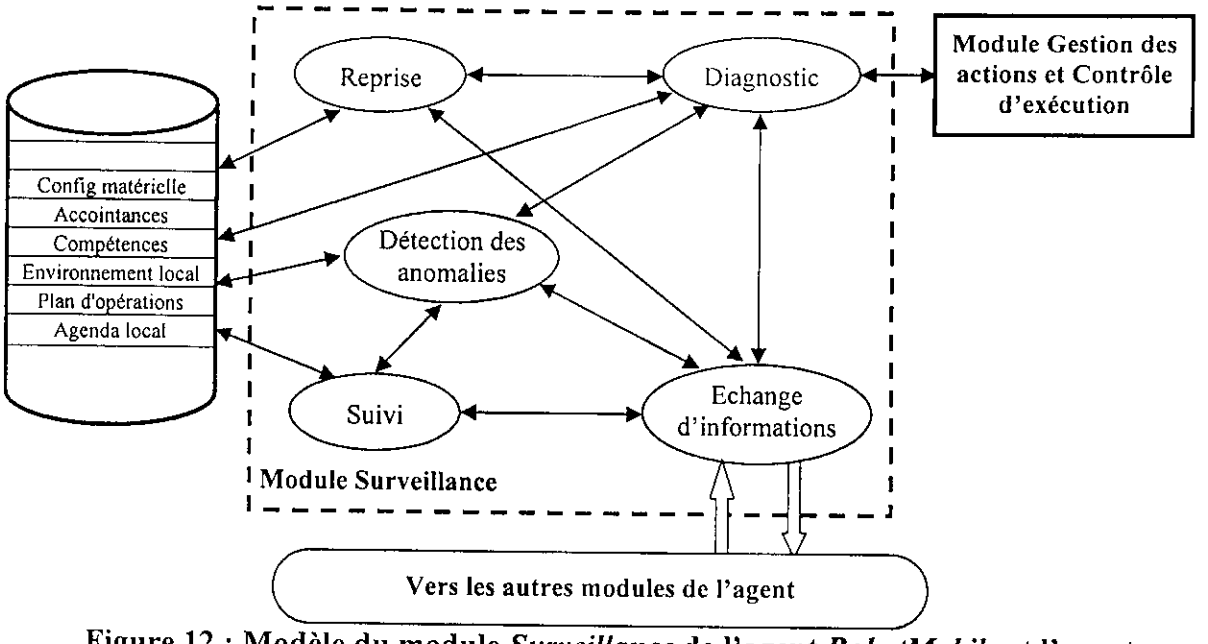


Figure 12 : Modèle du module *Surveillance* de l'agent *RobotMobile* et l'agent *RobotManipulateur*

Inventaire des anomalies possibles

Base mobile

- Obstacle imprévu

Bras manipulateur

- Une pièce tombe au cours de son déplacement par le bras manipulateur

IV.7.2.6. Module Gestion des actions et Contrôle d'exécution

Le module *Gestion des actions et Contrôle d'exécution* est composé, en plus de la tâche *Echange d'informations*, des tâches suivantes :

- *Exécution* : Cette tâche lance les opérations sous contrat figurant dans le calendrier en scrutant son *Horloge*. A la fin de l'exécution de chaque opération, il envoie un compte-rendu au module *Surveillance*.
- *Ordonnancement/Réordonnancement* : Cette tâche permet de choisir le meilleur séquencement des opérations pour réaliser ses objectifs (le plus rapide). Aussi, en cas de panne ne pouvant être traitée au niveau de cet agent, cette tâche construit une

annonce d'un nouvel appel d'offre qui sera envoyée en priorité vers les autres agents du système en sollicitant la tâche *Négociation*.

- *Négociation* : Elle est invoquée par la tâche *Ordonnancement/Réordonnancement* en vue d'effectuer un ordonnancement coordonné entre les différents agents ou dans le cas où le traitement de la panne (établir un nouveau plan) nécessite une négociation avec les autres agents du méta-agent.

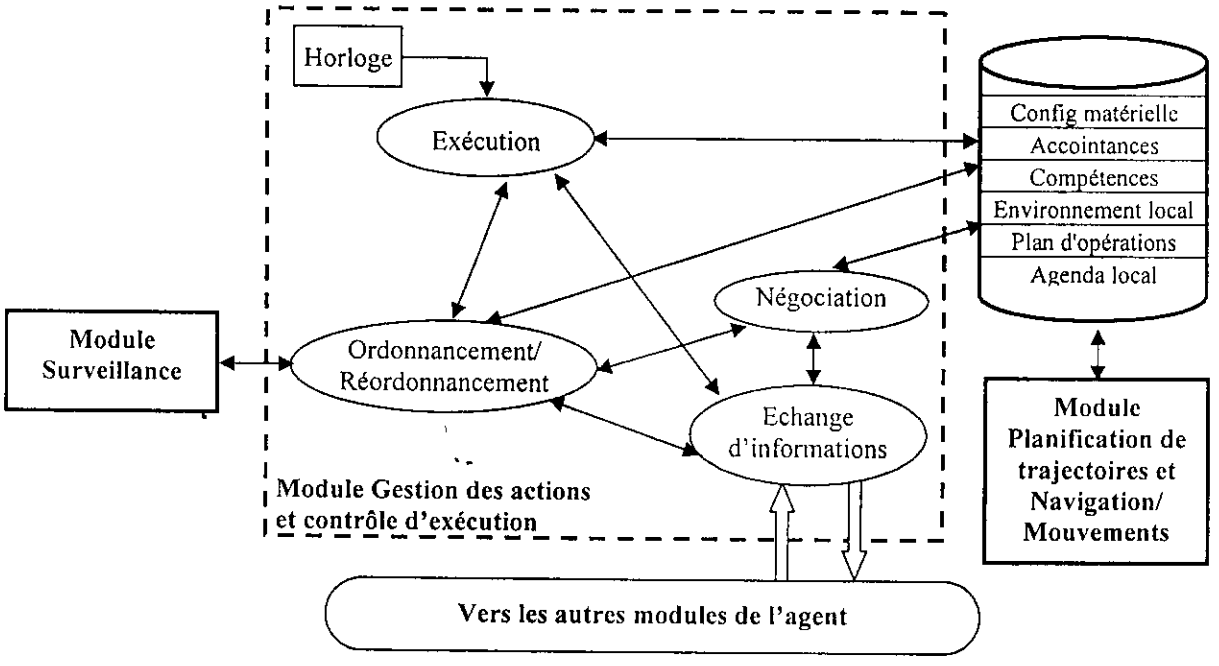


Figure 13 : Modèle du module *Gestion des actions et Contrôle d'exécution* de l'agent *RobotMobile* et l'agent *RobotManipulateur*

IV.7.3. Agent *SimulateurGraphique*

Un quatrième agent a été envisagé pour permettre la visualisation temps réel de l'état d'exécution des tâches du robot. A chaque apparition d'un nouvel événement (début d'une action, perturbation du plan d'exécution, etc.), l'agent *SimulateurGraphique* les reçoit, les interprète et les affiche sur écran au fur et à mesure.

Le modèle d'architecture de l'agent *SimulateurGraphique* est montré par la figure 14.

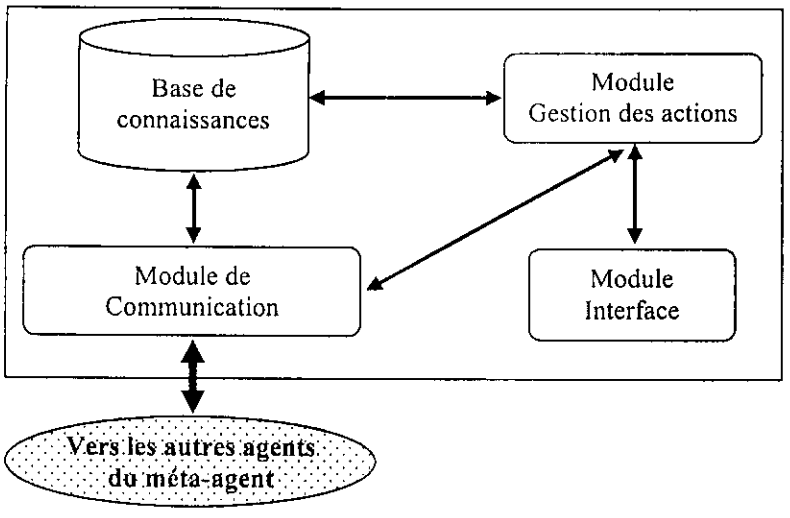


Figure 14 : Modèle de l'agent *SimulateurGraphique*

L'architecture de l'agent *SimulateurGraphique* consiste en un méta-agent constitué d'un ensemble de trois agents (Agent *SuperviseurSim*, Agent *RobotMobileSim* et Agent *RobotManipulateurSim*). Cet agent peut être schématisé de la façon suivante (Figure 15) :

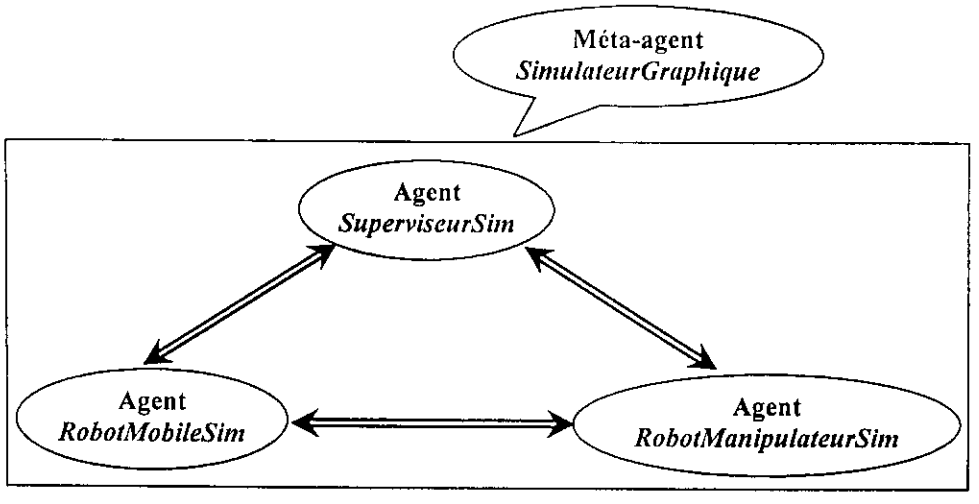


Figure 15 : Représentation du méta-agent *SimulateurGraphique*

Ces agents simulent les trois agents (Agent *Superviseur*, Agent *RobotMobile* et Agent *RobotManipulateur*) constituant le méta-agent *ManipulateuMobile* et expliqués dans les paragraphes précédents.

Tout ce qui a été dit auparavant concernant les architectures internes des trois agents (Agent *Superviseur*, Agent *RobotMobile* et Agent *RobotManipulateur*) constituant le méta-agent *ManipulateuMobile* est valable pour l'architecture interne des trois agents constituant le méta-agent *SimulateurGraphique* (Agent *SuperviseurSim*, Agent *RobotMobileSim* et Agent *RobotManipulateurSim*).

IV.8. Modèle de la base de connaissances

IV.8.1. Architecture à base de règles

Dans les architectures à base de règles, on assigne un système expert (ou système de règles) à un agent. Un système expert est composé de trois parties essentielles :

- *Base des faits (BF)* : La base de faits est la mémoire du système expert. Elle contient la description des états physiques et mentaux du monde (tout ce que l'on sait sur le cas que l'on est entrain de traiter avant l'intervention du moteur d'inférences). Elle peut être ensuite complétée par les données de l'utilisateur ou alors par les informations déduites du moteur d'inférences.
- *Base de règles (BR)* : Elle rassemble les connaissances de l'opérateur sur la ressource et les outils qui peut être enrichie dynamiquement par apprentissage. Elle permet de déduire des faits à partir d'autres faits. Une règle est une expression de la forme :

Si « *Prémisses* » *Alors* « *Conclusions* »

Le terme *Prémisse* équivaut à « *Condition (fait)* ».

Une règle est désignée par :

  - ses conditions de déclenchement (prémisses)
  - ses effets (conclusions)

L'ensemble des règles forme la base de règles.

- *Un moteur d'inférence* : ensemble d'heuristiques de planification qui utilisent la BF et la BR pour effectuer une tâche donnée.

IV.8.2. Base de connaissances locale (Agent *RobotMobile/RobotManipulateur*)

Elle est composée de plusieurs structures de connaissances :

- *Base de connaissances de la configuration matérielle de la base mobile (robot manipulateur)* : Cette base de connaissances est mise à jour automatiquement. L'organisation de cette base est mise à jour automatiquement lors de l'ajout ou du retrait d'une ressource ou d'un équipement de l'agent *RobotMobile* (ou *RobotManipulateur*). Elle comprend :

    - la liste des équipements de la base mobile (resp. bras manipulateur) et leurs états ;
    - la liste des opérations réalisables par les équipements de la base mobile (resp. bras manipulateur) ;

- *Base des accointances* : C'est l'ensemble de tous les agents voisins (accointances physiques) et tous ceux qui contrôlent une ressource similaire (accointances logiques, c.-à-d. les autres méta-agents de la cellule qui contrôlent d'autres robots ou machines). L'organisation de cette base est mise à jour automatiquement lors de l'ajout ou du retrait d'un autre agent.
- *Base de compétences* : Elle représente la mémoire de l'agent. Ces connaissances sont en générale les compétences de l'agent.
- *Environnement local* : Les informations acquisitionnées des différents capteurs qui équipent la base mobile (resp. le bras manipulateur) sont prétraitées par le module *Gestion des capteurs* et stockées dans cette base de données.
- *Plans d'opérations* : Ensemble d'opérations à réaliser par l'agent *RobotMobile* (resp. l'agent *RobotManipulateur*).
- *Agenda local* : En vue d'une analyse ultérieure, l'agenda local de l'agent *RobotMobile* (resp. l'agent *RobotManipulateur*) contient tous les contrats que l'agent s'est engagé à traiter. Il contient aussi les conditions d'exécution des opérations : instants de démarrage et de fin, occurrence d'évènements (perturbation, demande de négociation, fin de négociation), type de décision prise (ordonnancement local, ordonnançement coordonné), etc.

IV.9 Conclusion

Dans ce chapitre nous avons présenté le modèle détaillé de l'architecture de contrôle/commande que nous avons proposé pour le pilotage du robot manipulateur mobile *RobuTER*. Nous avons donné les différentes tâches constituant tous les modules des trois agents de l'organisation ainsi que leurs fonctions et leurs intercations.

Nous avons spécifié les différents messages et événements échangés dans le système multi-agents ainsi que leurs formats et leurs priorités.

Nous avons présenté aussi les différentes bases de connaissances que dispose chacun des deux agents *RobotMobile* et *RobotManipulateur* (base de connaissance locale) ainsi que la base de connaissance globale de l'agent Superviseur.

CHAPITRE V

TEST ET VALIDATION

CHAPITRE V

TEST ET VALIDATION

V.1. INTRODUCTION

Les tests est effectué sous Windows XP nous allons décrit dans ce qui suit tous les étapes de réalisation de notre système et on va exposé le résultat de chaque étape.

La première étape consistait à accompli des outils graphiques 3D pour la modélisation graphique.

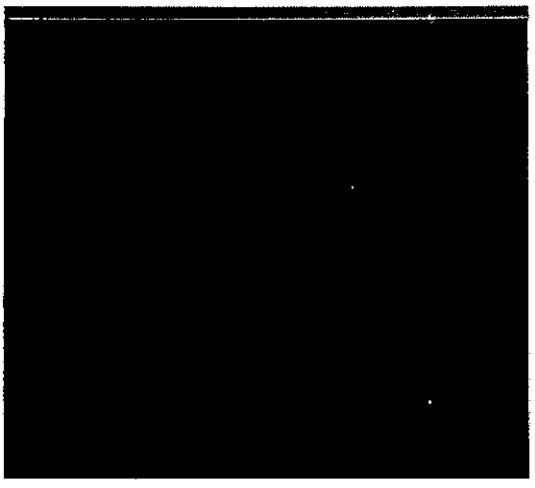
V.2. Création de la base mobile:

On a crée une classe sommet pour structuré les vertex :

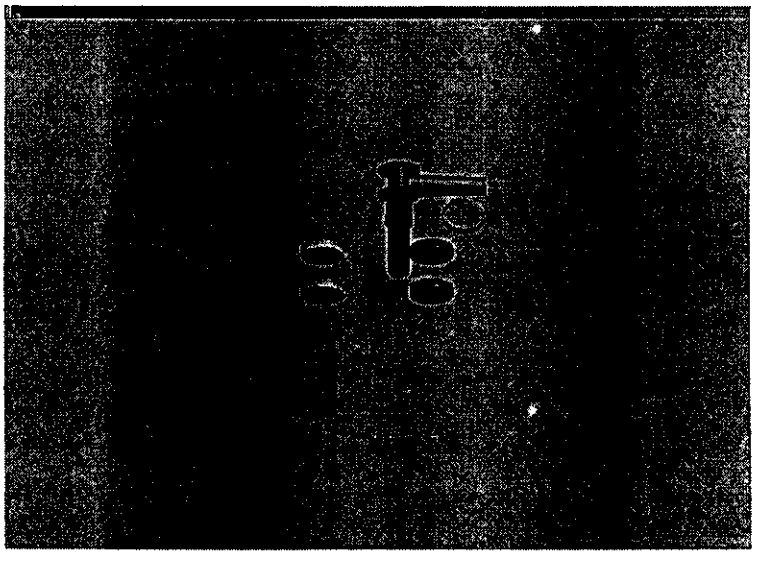
```
class sommet
{
public:
    double x;
    double y;
    double z;
};
```

est un tableau qui contient tous les vertex de type sommet.

Ce tableau contient tous les points pour construire un cube



V.3. Construction de bras manipulateur et les roues:



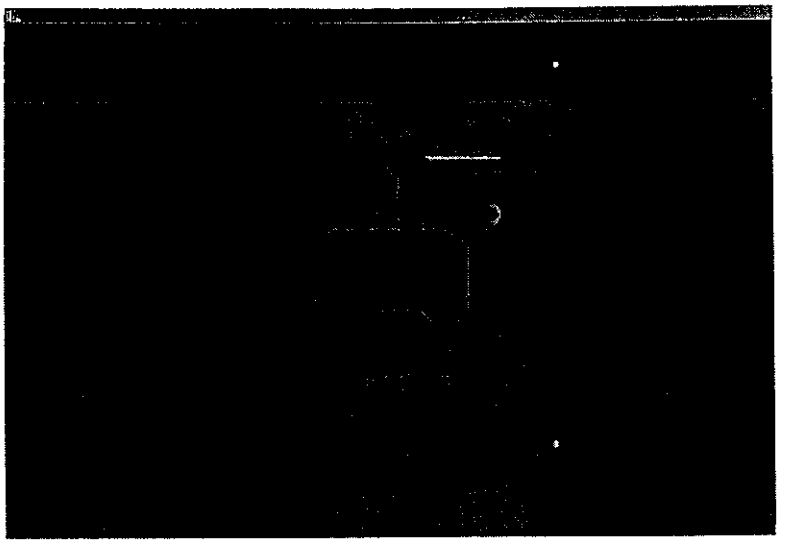
Dans cette étape on a dessiné le bras et les quatre roues les deux verts motrices et les deux autres folles.

Le bras est en mode cylindrique contient les corps plus les liaisons,une liaison c'est la partie qui assure la jointure entre deux corps; on trouve une liaison prismatique (en bleu).

Le bras contient trois corps de type cylindre un tronc, l'épaule, et un corps terminale.

Cette version visualise les deux modules bras manipulateur et la base mobile. Ce modèle géométrique du robot est assuré par les deux méthodes base ( ) et bras ( ) mais la scène n'est pas encore complet il reste l'environnement.

```
void baseM()
 {
  vertic_sup[0].x=-50;
   vertic_sup[0].y=0;
   vertic_sup[0].z=-20;

   vertic_sup[1].x=50;
   vertic_sup[1].y=0;
   vertic_sup[1].z=-20;

   vertic_sup[2].x=70;
   vertic_sup[2].y=20;
   vertic_sup[2].z=-20;

   vertic_sup[3].x=70;
   vertic_sup[3].y=80;
   vertic_sup[3].z=-20;

   vertic_sup[4].x=50;
   vertic_sup[4].y=100;
   vertic_sup[4].z=-20;

   vertic_sup[5].x=-50;
   vertic_sup[5].y=100;
   vertic_sup[5].z=-20;

   vertic_sup[6].x=-70;
   vertic_sup[6].y=80;
   vertic_sup[6].z=-20;

   vertic_sup[7].x=-70;
   vertic_sup[7].y=20;
   vertic_sup[7].z=-20;

  for(int i=0; i<=7;i++)
    {
     vertic_inf[i].x=vertic_sup[i].x;
     vertic_inf[i].y=vertic_sup[i].y;
     vertic_inf[i].z= 0;

    }
  glColor3f(0.20,0.2,0.2);
  glBegin(GL_POLYGON);
 for(int i=0; i<=7;i++)
     glVertex3f(vertic_inf[i].x,vertic_inf[i].y,vertic_inf[i].z);
  glEnd();}
```



La scène est réalisé grace au méthodes Glscène

```
void __fastcall TFormMain::RenderGLScene()
{
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  DrawObjects();
  glFlush();
}
```

V.4 Création de la scène finale:

L'environnement est au choit de l'utilisateur, il contient des pièces de travail, des obstacles, et le sole ou le robot doit navigué.

La forme géométrique du sole est un rectangle, les deux autres objets sont de type cube.

La scène complète est réalisé par la méthode DrawObjects()

```
void __fastcall TFormMain::DrawObjects()
{
 glPointSize(10);
   baseM();
    bras();
    env();
return;
}
```

V.6. Conclusion:

Cette version de test assure uniquement le modèle géométrique du robot ROBUTER et les caractéristiques du robot sont stockés dans une base de connaissance pour les utilisés et les manipulés par l'utilisateur l'hors de création d'une stratégie de navigation de robot.

CONCLUSION GENERALE

CONCLUSION GÉNÉRALE

Dans notre travail on a utilisé les concepts de l'Orienté Objet et des patterns en utilisant une méthode de développement agile. On garantit par l'utilisation de ses principes la réduction de la rigidité, de la fragilité, de l'immobilité et de la viscosité de notre système.

Les patterns, précisément Observer, nous ont permis de réduire les calcules des matrices composites (de chaque lien) en les maintenant au niveau des liens. L'Observer nous a permis de séparer les différents composants du système (implique une réutilisation possible); le modèle géométrique, modèle cinématique du robot, les camera, la scène et l'interface graphique sont séparés.

Dans ce travail on est arrivé a utiliser Observer en dehors de son application standard (MVC); la conclusion est que le pattern Observer peut être utilisé pour la communication entre modules et classes.

Après l'utilisation de l'Observer fournis par java, on a pus implémenter l'Observer avec la possibilité de changer la structure de donnée interne de la classe observable (liste d'Observers, tableau d'observeurs ...etc.).

Le pattern strategy a permis de tester différentes implémentations des modules de camera,

La structure en liste, du modèle cinématique, nous a permis de communiquer facilement la matrice composite du lien précédent celui qui a change.

Le langage java nous a offert un ensemble d'API pour le développement d'interfaces graphiques.

Perspectives

L'objectif principal de notre travail est atteint; on a pus appliquer les concepts de la conception orientée objet et des patterns en travaillant selon les principes de la méthode XP. Néanmoins, il reste a introduire certaines fonctionnalités dans la simulation :

- Nous avons conçu une base de donnée afin de charger la géométrie et les propriétés du robot, Il reste à intégrer ce composant dans le logiciel.
- Implémenter le traitement des faces cachées.
- Implémenter un module d'illumination.
- Concevoir un *solver* de cinématique inverse.

On n'a pas pus terminer les caractéristiques et les modules cités ci-dessus faute de temps.

Extensions possibles

On peut imaginer plusieurs extensions de notre travail :

- Evitements d'obstacles,
- Intégrer de nouveaux objets dans la scène, que le bras pourrait manipuler,
- Introduire la dynamique des manipulateurs dans le système,
- Utiliser des techniques plus élaborées dans le rendu et l'animation du robot,
- Utiliser d'autres patterns dans le développement de notre système (qui peuvent avoir une relation avec les patterns déjà utilisés)
- ... etc.

REFERENCES

REFERENCES

1. Zagal, J.C., del Solar, J.R.: UCHILSIM: A Dynamically and Visually Realistic Simulator for the RoboCup Four Legged League.

2. Smith, R.: Open Dynamics Engine - ODE (2005) www.ode.org.

3. Go, J., Browning, B., Veloso, M.: Accurate and flexible simulation for dynamic,vision-centric robots.

4. **Robert** C. Martin, Designing Object Oriented Applications using UML, 2d ed, Prentice Hall, 1999.

5. Go, J., Browning, B., Veloso, M.: Carnegie Mellon UberSim

6. Koenig, N., Howard, A.: Gazebo - 3D multiple robot simulator with dynamics.

7. Gerkey, B., Vaughan, R.T., Howard, A.: The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems. In: Proceedings of the 11th International onference on Advanced Robotics, Coimbra, Portugal (2003) 317 – 323

8. Michel, O.: Cyberbotics Ltd. - WebotsTM: Professional Mobile Robot Simulation. 1 (2004) 39–42

9. Cyberbotics Ltd.: Cyberbotics Webots (2005).

10. Rofer, T.: Strategies for using a simulation in the development of the Bremen utonomous Wheelchair. In Zobel, R., Moeller, D., eds.: Simulation-Past. Presentand Future, Society for Computer Simulation International (1998) 460–464.

11. Ghazi-Zahedi, K., Laue, T., Rofer, T., Scholl, P., Spiess, K., Twickel, A., Wischmann, S.: Rosiml - robot simulation markup language (2005) .

12. Kurlbaum, J., Laue, T., Luck, B., Mohrmann, B., Poloczek, M., Reinecke, D., Riemenschneider, T., Rofer, T., Simon, H., Visser, U.: Bremen Small Multi-Agent Robot Team (B-Smart) Team Description for RoboCup 2004. In: RoboCup 2004: Robot Soccer World Cup VIII. Lecture Notes in Artificial Intelligence, Springer (2005)

13. Mandel, C., Hübner, K., Vierhuff, T.: A Demonstrator for Cognitive Aspects in Service Robotics. In: XXVII Annual Meeting of the Cognitive Science Society (submitted). (2005)

14. German Team: German Team web site. (2005)

2. WEBOGRAPHIE:

http://www-2.cs.cmu.edu/ robosoccer/ubersim/.

http://playerstage.sourceforge.net/gazebo/gazebo.html.

http://www.cyberbotics.com/

products/webots/.

http://www.tzi.de/simrobot.

UBERSIM http://www-2.cs.cmu.edu/ robosoccer/ubersim/.

SIMROBOT http://www.tzi.de/simrobot.

http://www.cyberbotics.com/.

http://www.robocup.de/germanteam/GT2004.pdf.

http://www.tzi.de/spprobocup/RoSiML.html.

http://www.LAAS.fr