Assistance robotisée à la personne en environnement coopérant

Antonio Andriatrimoson

UNIVERSITE D'EVRY VAL D'ESSONNE

U.F.R Sciences & Technologies : Laboratoire IBISC
Ecole Doctorale : Science et Ingénierie
Département : Génie Informatique

Thèse

présentée pour l'obtention du grade de

Docteur de l'Université d'Évry Val d'Essonne

Spécialité : Robotique

par **Antonio ANDRIATRIMOSON**

Assistance robotisée à la personne en environnement coopérant

Soutenue publiquement le 11 janvier 2012 devant le jury composé de

Président:	Slimane Mohamed, Professeur des Universités
Rapporteurs:	Guessoum Zahia, Professeur des Universités Chatila Raja, Directeur de Recherche
Examinateur:	Slimane Mohamed, Professeur des Universités
Co-Encadrant:	Nadia Abchiche, Maître de Conférences
Directeur de Thèse:	Etienne Colle, Professeur des Universités

---

Informatique, Biologie Intégrative et Systèmes Complexes
Université d'Evry Val d'Essonne

/

Remerciements

Je remercie chaleureusement Étienne Colle, Professeur des Universités à l'Université d'Évry Val d'Essonne. Il dirige avec énergie et brio notre équipe HANDS (Handicap et Santé) du laboratoire IBISC. Il a dirigé avec élégance et finesse mes recherches. Son dynamisme communicatif a été une source de motivation constante pour moi au cours de ces trois années.

Je remercie vivement également Nadia Abchiche, Maître de Conférence à l'Université d'Évry Val d'Essonne. Elle a accepté de co-encadrer cette thèse en cours de route et a joué un rôle décisif dans la maturation de ces travaux grâce à ses idées et son ouverture d'esprit.

J'adresse mes remerciements à Mme Zahia Guessoum, Messieurs Raja Chatila et Mohamed Slimane qui ont acceptés de juger ce travail.

Je remercie aussi Monsieur Philippe Hoppenot, coordinateur du projet QuoVADis qui a permis de financer ma thèse. Il également apporté des précisions dans mes travaux.

Je tiens à souligner combien la contribution de Simon Galerne ainsi que les échanges d'idées avec lui ont été précieux pour mener à bien ces travaux.

Je remercie aussi Séfiane Lyes et Maxime Jubert avec qui j'ai été amené à collaborer pour ces travaux de thèse pendant leur stage.

Je remercie également Sébastien Delarue et Nadjet Delarue pour leur disponibilité et leur gentillesse avec qui j'ai eu l'occasion de partager des idées et de nombreux propos sur autant de sujets différents.

Merci à tous les membres de l'équipe pour toutes ses années de collaboration et de sympathie. Je n'oublie pas non plus de remercier également tous les autres membres et les personnels du laboratoire IBISC avec qui j'ai eu l'occasion de travailler ensemble.

Mes pensées vont enfin vers tous ceux qui m'ont soutenus moralement, c'est à dire à ma femme, ma famille et mes amis, sans leur soutien, sans doute ne seriez-vous pas dans la situation de lire ce manuscrit, ni moi-même d'écrire ces mots, tant la vie réserve parfois des épreuves difficiles.

"Cherche une voie noble, distingue-toi par des valeurs, multiplie toujours les efforts, reste constant." Florent Parfait KAMELAN.

Table des matières

Introduction générale

Ces travaux de thèse s'inscrivent dans le cadre du projet ANR QuoVadis TEcSan 2007-2011.

L'objectif du projet QuoVadis est de compenser les difficultés de communication dues aux pertes de capacités cognitives qui génèrent l'isolement social, la dépression, l'insécurité et l'inconfort dans la vie quotidienne d'une personne généralement âgée ayant des troubles cognitifs. Le système vise à rétablir le lien affectif avec les proches, aidants et soignants par un système mobile interactif accompagnant la personne en difficulté et à lui permettre de se repérer dans son environnement et de le contrôler. Il a pour but de faciliter la prise en charge des pathologies cognitives (Maladie d'Alzheimer (MA) ou apparentées) et d'alléger le fardeau des aidants. L'intérêt d'un système mobile est précisément un accompagnement constant accepté et souhaité et valorisé sur le plan affectif et de la sécurité médicale.
Le maintien à domicile de personnes présentant des troubles cognitifs est une alternative à l'hospitalisation qui répond à la tendance actuelle de réduction du nombre de lits hospitaliers. Le projet QuoVADis a pour objectifs de répondre à deux des problèmes posés par le maintien à domicile : la stimulation cognitive et la sécurité médicale des personnes, ceci de manière transparente, discrète pour l'usager et complémentaire à d'éventuels dispositifs déjà en place.

Stimulation cognitive du patient - Pour les personnes atteintes de troubles cognitifs (Alzheimer par exemple), les enjeux sont de stimuler les capacités cognitives résiduelles en aidant le patient à se repérer dans le temps et dans l'espace, en le sécurisant en cas d'errance, de désorientation et d'angoisse, et en facilitant sa communication avec l'entourage. La stimulation cognitive des patients passe par des exercices avec une tierce personne et/ou un dispositif d'aide issu des STIC (Science et Technologie de l'Information et de la Communication). L'apport principal visé est l'introduction novatrice dans le domaine de la stimulation cognitive du concept de "robot compagnon" multimédia interactif et mobile permettant de stimuler le patient, ce qui est différent des systèmes fixes actuellement proposés (tablettes).

Sécurité du patient - Elle est basée sur un système de télévigilance de patient à domi-

cile qui a pour objectif de permettre à des personnes dépendantes ou fragilisées de rester chez elles et d'être aidées ou "médicalisées" à distance de manière plus réactive en cas d'urgence. Les personnes concernées par cette fonction de sécurité sont les personnes dépendantes mais aussi des personnes atteintes de pathologies cardiaques ou des personnes en phase de convalescence post-clinique nécessitant une surveillance médicale rapprochée. Les points clés de cette partie du projet résident dans la fusion de données provenant de différents capteurs visant à minimiser le taux de fausses alarmes et à améliorer la transmission et stockage des données vers la station domestique fixe.

L'originalité principale du projet QuoVADis réside dans la combinaison de moyens robotiques avec d'une part, un système de stimulation cognitive et d'autre part, un système intégré de télévigilance. Le "robot compagnon" apporte sa capacité de mobilité et d'interaction selon différentes modalités (audiovisuelle, tactile). Dans le cas de la stimulation cognitive, ses capacités permettent de :

- maintenir de manière interactive un contact entre la personne et le dispositif de stimulation cognitive,
- d'enrichir les possibilités de stimulation.

Dans les deux cas, le robot agit soit de façon autonome soit piloté à distance via internet par une personne extérieure.

Dans le cas de la sécurité, les capacités du robot permettent de :

- vérifier la pertinence de l'alarme et son degré d'urgence. En effet, les dispositifs actuels de télévigilance ne permettent pas encore d'identifier les fausses alarmes de façon suffisamment fiable,
- établir un contact visuel et auditif avec la personne afin d'affiner le diagnostic en cas d'alarme avérée,
- maintenir un contact continu en attendant une intervention humaine.

Contexte et enjeux médicaux et sociaux économiques du projet

Pertinence d'un système de soutien précoce, complémentaire au suivi médical.
Parmi les pathologies chroniques présentant des troubles cognitifs, la progression de la maladie d'Alzheimer représente un défi majeur de santé publique : en France, 150 000 nouveaux cas seraient (selon les statistiques de santé) diagnostiqués chaque année. Le caractère évolutif et aujourd'hui encore, irréversible de cette maladie, le rôle limité des médicaments actuellement utilisés au stade léger ou modéré et les risques qui y sont fréquemment associés (perte pondérale, anorexie, confusion, dépression et troubles du système nerveux, vertiges, somnolence, tremblements) rendent nécessaire une aide à la

vie quotidienne qui retarde le plus possible la perte d'autonomie.
Cette aide doit notamment s'efforcer de stimuler les capacités cognitives résiduelles, en aidant le patient à se repérer dans le temps et dans l'espace, en le sécurisant en cas d'errance, de désorientation et d'angoisse, et en facilitant sa communication avec l'entourage.
Outre la perte progressive des capacités cognitives, la maladie d'Alzheimer s'accompagne d'un besoin souvent impérieux de déambulation, lequel s'accommode mal d'obstacles mis à une libre circulation pour éviter les fugues et les errances (crises, angoisse).
A contrario, l'absence de tels obstacles ou les défauts d'une surveillance constante ou d'un accompagnement par une personne de l'entourage créent des situations à risques pour le patient et sont la source d'une angoisse permanente et d'une lourde charge pour les aidants.

Pertinence d'un système "robot compagnon" mobile

Au domicile, le patient est mobile d'une pièce à l'autre. Mais sa perte de capacités cognitives l'amène à se refermer sur lui-même, faute de pouvoir communiquer facilement. Elle crée aussi des situations à risques, oubli de prise de médicament, de l'heure des repas, de contrôle des appareils domestiques, des portes ou des fenêtres. L'aidant ne peut pas toujours être là. Le suivi par des capteurs fixes, outre les problèmes d'installation, d'efficacité, de maintenance, de liaison avec les plateformes d'assistance et de respect de l'intimité qu'il pose, ne résout pas ce problème d'accompagnement au quotidien de la personne désorientée, en complément de la présence de l'aidant.
Il est donc pertinent de rechercher comment un compagnon mobile doté d'une intelligence artificielle pourrait à la fois stimuler les fonctions cognitives, rassurer en aidant au repérage temporel (jour heure, rendez-vous, rappel des choses à faire, repas, coucher), instaurer un dialogue approprié, faciliter la communication avec l'équipe médicale, l'entourage proche ou éloigné, sécuriser en repérant les situations de crises ou à risques et déclencher les liaisons d'assistance, faciliter l'accès aux loisirs : musiques et images préférées, liaisons audio-visuelles. Le bébé phoque artificiel testé en Europe par les japonais et qui répond à la caresse de personnes en perte de capacités cognitives par un regard et des sons a été favorablement accueilli par les patients.

Intérêt économique d'un "robot compagnon" mobile

Toutes les études montrent d'une part que le choix politique pour favoriser le maintien à domicile et retarder le plus possible l'entrée en institution, s'il est partagé par les patients et leurs aidants, fait néanmoins porter la charge sur les aidants. Les projections démographiques montrent que le nombre d'aidants familiaux va tendre à diminuer alors que le nombre de personnes atteintes de la MA risque de fortement augmenter.
Dans cette perspective l'intérêt de pouvoir compléter l'aide humaine par de tels compagnons technologiques est indéniable et peut apporter aussi une solution économique au coût élevé de l'aménagement spécifique d'appartements personnels.

Structure du mémoire de thèse

Les travaux de cette thèse s'appuient sur le scénario de télévigilance, décrit dans le premier chapitre, pour apporter une aide à la levée de doute et au diagnostic, après déclenchement d'une alarme.
La thèse apporte des contributions à l'autonomie du robot grâce à une coopération entre le robot et l'environnement coopérant constitué de réseaux d'objets communicants tels que des capteurs et des actionneurs. Les principales contributions portent sur la localisation du robot, sur des mécanismes permettant à celui-ci de s'adapter à la situation et enfin sur l'architecture permettant cette coopération.

La thèse est organisée comme suit :

- Le premier chapitre fixe le contexte applicatif des travaux de thèse en termes d'usages et de problématiques. Il contient un état de l'art en robotique ambiante permettant d'expliquer et de situer la particularité du système conçu. En particulier, la mobilité du robot est exploitée en coopération avec les objets de l'environnement (capteurs et actionneurs) et les différents utilisateurs du système (patient, aidants, personnel médical) pour répondre à un scénario de levée de doute en cas d'alarme. Dans ce type de scénario, la localisation est cruciale en vue de permettre au robot de se déplacer vers la personne.
- Le deuxième chapitre présente une méthode dont l'objectif est de localiser avec certitude le robot dans une zone de l'habitat. Les dimensions de cette zone peuvent être très variables. Ces dimensions dépendent des besoins de la tâche, voir de l'enchaînement de tâches, que prévoit de réaliser le robot à cet instant, mais découlent aussi de facteurs comme la dispersion des capteurs dans l'environnement, la qualité des mesures disponibles.

  Nous partons du principe qu'une bonne connaissance de sa situation et la certitude d'être dans cette zone, permettra au robot d'adopter la meilleure stratégie possible pour assurer sa mission qui est de retrouver la personne dans son habitat. Aussi, la position et l'orientation du robot doivent être données avec une précision garantie. C'est moins la précision que la certitude que le robot est localisé dans cette zone qui importe.
- Le troisième chapitre propose une architecture informatique permettant la coopération entre le robot et les objets communicants présents dans l'environnement tels que les capteurs ou les actionneurs. Nous nous sommes orientés vers les systèmes multi-agents, afin de concevoir une solution dotée de mécanismes d'adaptation permettant non seulement de prendre en compte l'aspect dynamique de l'environnement, mais aussi des facteurs tel que le degré d'intrusion du système dans la vie privée de la per-

sonne ou le degré de précision des résultats. En effet, un capteur peut être pertinent à un instant donné et ne plus être utilisable l'instant d'après, selon sa situation, le degré d'urgence de l'intervention et le niveau d'intrusion autorisé. Par exemple, en cas de chute du patient détectée, un niveau d'intrusion, tel qu'une caméra est autorisée à effectuer un balayage sera mis en place. Alors qu'en mode de fonctionnement normal, la caméra devra demeurer fixe.

– Le dernier chapitre décrit la mise en œuvre des différents composants de l'architecture de coopération robot-environnement basé sur une coaltion d'agents, les choix des implémentations effectuées, l'évaluation du protocole multi-agent pour la formation des coalitions et enfin l'évaluation dans le contexte applicatif réel de la localisation du robot.
– Enfin, la conclusion reprend les principales contributions de la thèse sans négliger une critique qui ouvre sur des améliorations possibles et cite les perspectives, à court et à moyen terme, que ce travail déploie.

Chapitre I

Cadre et problématique de la thèse

Objectifs du chapitre

Ce chapitre commence par définir la robotique ambiante d'assistance à la personne en perte d'autonomie qui est un sous-ensemble de la robotique ubiquitaire. Une revue de projets représentatifs classés par type de technologie permet de préciser les usages qu'on en attend et de faire émerger les problématiques que se pose la communauté. Le chapitre se poursuit par un scénario d'usage, défini par le SAMU 92-APHP (Assistance Publique Hôpitaux de Paris) qui précise le contexte de la thèse. Ce scénario fixe le rôle attribué au robot et un certain nombre de contraintes dues au contexte. Les travaux de la thèse sont alors situés par rapport aux problématiques de la communauté modulées par le scénario.

I.1 Introduction

Depuis ces dernières années, la rapide et constante évolution des technologies de l'information et de la communication a permis le développement du domaine de l'ubiquité numérique qui a donné naissance à de nouveaux usages dont la liste s'accroît progressivement. La littérature s'appuie sur une riche terminologie pour définir ce domaine en distinguant les notions d'ubiquité numérique, d'informatique diffuse, d'intelligence ambiante, d'informatique omniprésente, d'informatique enfouie ou embarquée, d'informatique auto-régulée ou encore d'informatique nomade. Nous nous limiterons à définir les trois premieres. Marc Weiser a introduit en 1993 [Weiser1993] l'ubiquité numérique comme une technologie invisible à des utilisateurs avec lesquels elle entretient des interactions permanentes. Le concept de réseau ubiquitaire décrit notamment un environnement technologique spontané, qui permet à des utilisateurs d'accéder à des services à travers un ou plusieurs réseaux d'accès. Cet environnement est la plupart du temps composé de plusieurs technologies hétérogènes. L'objectif d'un tel environnement est d'offrir une interaction forte et transparente, la plus naturelle possible à l'utilisateur. Les réseaux ubiquitaires permettent

de communiquer partout avec des terminaux mobiles qui s'adaptent aux infrastructures existantes et offrent la possibilité de commuter d'un réseau à un autre en fonction des ressources demandées ou des services recherchés. [Banatre2007] définit l'informatique diffuse ainsi : "En général, les concepts de l'informatique diffuse reposent essentiellement sur une idée de couplage entre le monde réel et un ensemble de traitements automatiques". Proche de la notion d'ubiquité numérique, le terme d'informatique diffuse est employé généralement pour définir un domaine qui regroupe l'informatique mobile, les télécommunications sans fils, les interfaces homme-machine et l'informatique distribuée. Concrètement, l'informatique diffuse est composée de trois entités distinctes qui sont l'environnement réel, l'environnement virtuel ou numérique et une interface de couplage. L'environnement réel correspond aux différentes entités du monde réel comme les personnes, les objets, les lieux tandis que l'environnement numérique matérialise tous systèmes d'information et enfin l'interface de couplage est utilisée pour la liaison des deux environnements. L'objectif de l'informatique diffuse peut être vue sous deux angles symétriques, soit l'environnement réel devient partie intégrante des systèmes d'information soit ces derniers deviennent partie intégrante de l'environnement réel.

I.2 Intelligence Ambiante (AmI)

L'expression intelligence ambiante est apparue au début des années 1990 et peut être considérée comme un cas particulier de l'informatique diffuse. L'intelligence ambiante est une mise en relation des services et leurs technologies associées aux dispositifs pour personnaliser et adapter les comportements de ces services. Selon Bermejo Nieto et ses coauteurs [Bermejo2004] ainsi que Bergamann [Bergamann2007], notre environnement futur sera constitué de différents systèmes et applications, portés par les technologies des réseaux et l'informatique, destinés à fournir une aide intelligente sans intervention humaine. L'intelligence ambiante est un domaine pluridisciplinaire dont l'objectif principal est de rendre tous les lieux où la personne se trouve, plus profitable pour elle [Augusto2008]. Elle peut être définie par ces quatre caractéristiques :

- *L'ubiquité :* la capacité d'interagir, n'importe où, avec une multitude d'appareils interconnectés, de capteurs, d'actionneurs, et plus globalement avec les systèmes électroniques "enfouis" (embedded software) autour de lui. Tout cela à travers des réseaux adaptés et d'une architecture informatique distribuée.
- *L'attentivité :* la faculté du système à "sentir" en permanence la présence et la localisation des objets, des appareils et des personnes pour prendre en compte le contexte d'usage. Toutes sortes de capteurs sont nécessaires à cette fin : caméras, micros, radars, capteurs biométriques, tags couplés à des lecteurs RFID pour l'identification.
- *L'interaction naturelle :* l'accès aux services doit pouvoir se faire de la façon la plus

naturelle ou intuitive possible. L'interface homme-machine est multimodale.
- *L'intelligence :* la faculté d'analyse du contexte et l'adaptation dynamique aux situations.

I.3 Robotique ambiante pour l'assistance à la personne

I.3.1 Défi sociétal de la robotique d'assistance

[Tapus2010] liste les principaux champs d'application de la robotique d'assistance à la personne en situation de perte d'autonomie, situation qui peut être permanente, temporaire et évolutive :

- *Soins destinés aux personnes âgées.*
  Dans le contexte de la robotique sociale, les premiers efforts ont porté sur la réalisation de robots animaux robotisés, par exemple un bébé phoque, un chat, un ours en peluche ou encore un chien. L'objectif est de réduire le stress ou une tendance dépressive de la personne âgée. Les études ont montré que les utilisateurs âgés souriaient et riaient plus, et globalement devenaient plus sociables.
- *Soins à des personnes handicapées physiques.*
  Des dispositifs robotiques commencent à être disponibles pour suppléer, compenser ou rééduquer différentes parties du corps humain. Les orthèses sont conçues de façon à pouvoir compenser une fonction absente ou déficitaire tandis que les prothèses sont destinées à remplacer complètement une fonction manquante. Il faut signaler que seuls les dispositifs de rééducation sont actuellement réellement commercialisés.
- *Soins à des personnes présentant une déficience cognitive.*
  De manière surprenante, des chercheurs travaillent aujourd'hui à l'utilisation des robots pour venir en aide aux enfants atteints de troubles du développement. C'est le cas de l'autisme, qui se caractérise par des comportements stéréotypés et un retrait social et émotionnel, dont il est possible qu'il soit partiellement dû à la difficulté qu'ont ces enfants à percevoir et interpréter les comportements de leurs congénères qui pour eux sont trop variables. Certains chercheurs ont mis en évidence le fait qu'en présence de robots aux comportements et aux formes "animaloïdes" ou "anthropoïdes", plus simples et plus prévisibles que les humains mais plus interactifs que les objets inanimés, des enfants pouvaient sortir de leurs comportements stéréotypés et même profiter du robot comme un intermédiaire pour interagir avec les autres humains. Ces robots peuvent être utilisés comme outils d'aide au diagnostic et à la thérapie de problèmes de développement chez certains enfants, ainsi que dans le cadre de la stimulation cognitive de personnes âgées, notamment celles atteintes de la maladie d'Alzheimer.

I.3.2 Robotique ambiante

La rencontre de l'intelligence ambiante et de la robotique d'assistance a donné naissance à ce que nous avons appelé robotique ambiante d'assistance car destinée à assister une personne en perte d'autonomie. Elle s'appuie sur l'existence d'un réseau d'objets communicants présents dans l'environnement de la personne pour décliner un ensemble de services et de téléservices destinés à faciliter la vie quotidienne de cette personne et de son entourage. Un, voir plusieurs robots peuvent être présents dans cet environnement. Une communauté récente de recherche s'est construite autour de la robotique ubiquitaire. Tandis que les robots des générations précédentes ont été conçus pour réaliser des tâches spécifiques et construits en tant qu'unité indépendante, la nouvelle génération des robots vise l'ubiquité. L'autonomie du robot est, dans ce cadre, obtenue par une interaction étroite entre le robot et l'environnement ambiant communicant [Nader2008]. Jusqu'à ces dernières années le robot évoluait dans un environnement plutôt hostile qui ne lui facilitait pas la tâche. Dans le contexte de l'intelligence ambiante, les objets communicants de l'environnement peuvent jouer un rôle "facilitateur" en aidant le robot à se localiser, naviguer, rechercher un objet. Inversement le robot peut être vu comme un objet communicant qui peut être mis à contribution par d'autres services que l'assistance à la personne.

Généralement, l'objectif sociétal des projets récents d'assistance ambiante, tant nationaux qu'internationaux, est le maintien de la personne à domicile. Ce sont soit des personnes âgées, soit des personnes présentant des handicaps spécifiques. Dans la présentation de quelques projets phares qui suit, nous avons distingué les projets de recherche qui s'intéressent à la domotique ou aux habitats intelligents destinés à l'amélioration de la vie quotidienne de la personne, de ceux qui intègrent les technologies de la domotique et de la robotique pour le maintien de la personne à domicile.
L'objectif de cette présentation de projets est de situer la problématique de cette thèse par rapport à celle de la communauté.

I.4 Exemples de projets domotiques pour l'assistance de la personne

I.4.1 Projets COHAND, QuatrA , DANAH

Le projet COHAND [Belabbas2007] du laboratoire LAB-STICC vise à fournir une aide à la personne à mobilité réduite en lui proposant les services disponibles selon les caractéristiques de sa maison et ce à partir de son fauteuil. La commande du fauteuil est coordonnée avec le contrôle environnemental de la maison domotisée. Le projet s'est intéressé à la qualité de service pour fournir un service optimal en présence d'obstacles

ou de pannes de services. La solution est basée sur une approche utilisée en productique. Faisant suite au projet COHAND, QuatrA (Aide Ambiante Ajustée Automatiquement) [Lankri2009] donne la possibilité à l'utilisateur de sélectionner un service selon ses capacités et éventuellement une aide via des dispositifs d'assistance spécifiques à son handicap. Le projet a abouti à une véritable plateforme expérimentale déployée chez des utilisateurs en situation réelle. Le projet DANAH (Domotique, Aide à la Navigation et Assistance au Handicap) qui fait suite à QuatrA, affine les principes précédemment développés en apportant des réponses aux problèmes apparus au cours des évaluations. Viennent s'ajouter de nouvelles approches dans la gestion d'un habitat domotisé, telles la distribution du système, sa modularité ainsi que des aspects méthodologiques s'intéressant non plus à la personne dépendante mais à l'expert dont la tâche est justement de construire un environnement adapté pour elle.

I.4.2 GERHOME et DOMUS (Laboratoire de DOmotique et d'informatique Mobile de l'Université de Sherbrooke)

En France, `GERHOME`[1] est un laboratoire d'expérimentation et d'évaluation de services pour le maintien à domicile des personnes âgées qui utilise des technologies "intelligentes". GERHOME vise à définir les bases théoriques et pratiques visant à construire une maison dans laquelle les personnes souffrant de handicaps pourront vivre pendant une durée plus ou moins longue de façon autonome. L'objectif principal est de concevoir et d'expérimenter des solutions techniques supportant des services d'aide au maintien à domicile des personnes âgées, en utilisant des technologies domotiques intelligentes pour assurer autonomie, confort de vie, sécurité, surveillance et assistance à domicile. Ces services doivent permettre aussi de maintenir un lien social notamment avec les membres de la famille de la personne. Concrètement, l'habitat GERHOME ressemble à un appartement deux-pièces meublé de quarante mètres carrés, avec cuisine et salles d'eau. Les données sur les activités de l'occupant et les caractéristiques de la maison sont collectées au moyen de capteurs (mesure de température, d'humidité, de luminosité, de consommation électrique, système d'ouverture de portes-fenêtres ....) et de caméras disposés dans la maison.

Au Canada, le laboratoire `DOMUS`[2] a créé un des premiers appartements conçus en Amérique du Nord selon les principes de l'informatique diffuse. Destiné aux personnes atteintes de troubles cognitifs, c'est un habitat pourvu d'un environnement informatique doté de nombreux capteurs de toutes sortes. DOMUS vise à concevoir des dispositifs informatiques complexes qui communiquent entre eux, analysent les situations et réagissent en fonction d'elles. A la convergence des différentes technologies, les recherches se foca-

1. http ://gerhome.cstb.fr/
2. http ://domus.usherbrooke.ca/accueil/

lisent surtout sur des solutions qui permettraient d'associer plusieurs services issus de différents environnements hétérogènes. L'ensemble du dispositif permettra de choisir un service adapté à chaque situation. Les capteurs installés sous les chaises, sous les matelas ou en périphérie des pièces détectent les allées et venues des personnes ainsi que leurs oublis ou d'éventuelles anomalies. En communiquant avec la puce électronique portée par les patients, le système est en mesure de réagir et de prendre les décisions appropriées.

I.5 Exemples de projets combinant TIC, robotique et domotique pour un objectif applicatif large

I.5.1 Projet ANSER (Airport NonStop Surveillance Expert Robot)

ANSER[3] (qui est l'acronyme en français de Robot Expert de la Surveillance en Continu des Aéroports) est un robot mobile de taille moyenne conçu pour patrouiller à l'intérieur de vastes zones en plein air (comme les aéroports). Le robot est capable de manière autonome de naviguer dans des environnements semi-structurés, en évitant en toute sécurité les collisions avec des obstacles inattendus et en gérant le passage dans des espaces réduits. ANSER, possède aussi différents capteurs destinés à la surveillance, pouvant facilement s'intégrer avec les systèmes traditionnels de surveillance automatique. Grâce à l'utilisation de ses capteurs embarqués, le robot est capable de détecter la présence anormale de personnes et d'objets dans des endroits spécifiques, et d'identifier automatiquement une personne portant des étiquettes d'identification sécurisée, comme le RFID[4]. En outre, lorsque des événements anormaux sont détectés (des intrusions ou des objets égarés, des portes mal réglées, la fumée, le feu, des valeurs de température inattendues ou des bruits suspects) ANSER ouvre automatiquement un canal de communication sécurisé avec un être humain à distance d'une station de surveillance et passe ensuite à la téléopération, qui peut être partielle ou même complète. L'opérateur distant peut ainsi piloter le robot et interagir avec d'autres personnes ou individus se trouvant à proximité du robot. ANSER intègre des solutions pour des problématiques de navigation multicapteurs en robotique. A l'extérieur, sa navigation est basée sur l'utilisation de GPS tandis qu'à l'intérieur des bâtiments, le robot utilise généralement ses télémètres lasers pour se déplacer avec évitement d'obstacles mais il peut aussi combiner ses propres moyens avec différentes informations issues de plusieurs capteurs placés à différents endroits de l'aéroport comme des capteurs de présence ou encore des capteurs de passage de portes. En effet, avec les informations de

3. http ://www.genovarobot.com/

4. RFID signifie "Radio Frequency IDentification", en français, "Identification par Radio Fréquence". Cette technologie permet d'identifier un objet, d'en suivre le cheminement et d'en connaître les caractéristiques à distance grâce à une étiquette émettant des ondes radio

tous les capteurs du bâtiment est réalisée une fusion de données avec prise de décisions.

I.5.2 Projet URUS (Ubiquitous Robotics for Urban) et Projet PEIS (Physically Embedded Intelligent Systems)

URUS [5] est un projet européen qui a pour ambition d'intégrer un réseau de robots dans un environnement intelligent dans le but d'améliorer la qualité de vie en zone urbaine. Il s'interesse aux problématiques sociétales de l'orientation, de l'assistance, du transport des personnes et des biens. Le projet URUS a analysé dans un premier temps les besoins urbains et les contraintes de déploiement de robots dans les villes, et notamment certaines questions juridiques et éthiques portant sur la vie privée et la sécurité des personnes en présence d'un réseau de robots. Du point de vue des sciences et des technologies, le projet URUS se concentre sur la définition et le déploiement des technologies nécessaires pour répondre aux objectifs sociétaux, ainsi que sur les évaluations du dispositif complet en milieu urbain.

Avec des perspectives proches, le projet PEIS [6] [Broxvall2007] [Safiotti2008] combine les domaines de la robotique ubiquitaire et de l'intelligence ambiante pour construire des robots intelligents au service des personnes. C'est un projet de collaboration entre les chercheurs suédois et coréens qui s'interessent aux moyens de mettre en œuvre une application basée sur le paradigme de l'informatique diffuse, qui permettrait d'implémenter une solution efficace pour la gestion et la collaboration de plusieurs robots autonomes. Le principe général étant que chaque robot peut être considéré comme un ensemble de plusieurs composants matérialisant ses différentes fonctionnalités principales. Il peut être possible alors de combiner différents composants hétérogènes minimaux entre eux afin de générer d'autres fonctionnalités complémentaires.

I.6 Exemples de projets combinant TIC, robotique et domotique pour l'assistance à la personne

Plusieurs projets se sont intéressés à combiner les TIC [7] dont la domotique et la robotique, avec pour objectif principal la sécurité du patient à domicile en proposant un service de télévigilance. Le robot intervient soit en étant téléopéré, soit de façon autonome en cas d'alerte ou d'appel volontaire du patient. Selon les projets, d'autres services viennent s'ajouter à celui de la télévigilance comme la stimulation, l'aide technique ou le lien social.

5. http ://urus.upc.es/index.html
6. http ://aass.oru.se/ peis/
7. TIC : Technologie de l'Information et de la Communication

I.6.1 RoboCare

Le projet RoboCare [Bahadori2006] initié par l'ISTC (Institut for Cognitive Science and Technology) en Italie vise à rendre indépendance et qualité de vie aux personnes âgées à domicile en intégrant différentes technologies. L'objectif du projet étant de trouver les meilleures façons de combiner les aidants et la technologie intégrant un ou plusieurs robots, afin que l'ensemble agisse dans un cadre de coopération harmonieuse au service de la personne. Le projet combine différentes techniques d'intelligence artificielle et la vision computationnelle. Cette dernière consiste à installer des caméras placées dans l'environnement qui peuvent détecter la présence de l'utilisateur et suivre son activité quotidienne. Les acteurs de ce projet cherchent des solutions basées sur l'intelligence artificielle, notamment les systèmes multi-agents pour offrir des services personnalisés à la personne à son domicile. Le robot est considéré de façon immergée et donc complètement transparente comme un objet appartenant à l'environnement ambiant.

I.6.2 iDorm (Intelligent Dormitory)

iDorm [Remahnino2006] est un projet développé par le département d'informatique de l'université d'Essex au Royaume-Uni. iDorm est une plateforme composée d'une pièce pour plusieurs usages : dormir, faire des loisirs, travailler, etc. L'habitat est doté de plusieurs capteurs comme des capteurs de température, de présence, d'humidité mais aussi de plusieurs actionneurs pour ouvrir ou fermer la porte, allumer ou éteindre les radiateurs, ouvrir ou fermer les volets. Pour le contrôle de l'environnement et sa visualisation, l'utilisateur dispose d'une application de réalité virtuelle. Le projet iDorm a pour objectif d'évaluer un environnement ambiant classé essentiellement en trois catégories d'objets communicants qui sont les objets statiques associés au bâtiment, un robot et des objets portables. Les objets portables étant destinés au contrôle à distance du système.
L'architecture d'iDorm est constituée d'un système multi-agent qui gère le fonctionnement de l'ensemble des capteurs et du robot. Un type d'agent gère les capteurs et un autre gère le robot. Le premier type, reçoit les différentes mesures issues des capteurs et contrôle les actionneurs qui y sont rattachés. Ces agents repèrent aussi dans le temps les habitudes de l'utilisateur et ses préférences par apprentissage. L'agent robot réagit comme un serveur de données et coordonne les échanges d'informations issues de l'interaction avec l'utilisateur. Il gère généralement la navigation du robot dans l'habitat en combinant différentes fonctions comme l'évitement d'obstacles ou encore la recherche d'objectifs.

I.6.3 QuoVADis, CompanionAble et RoboDOMO

QuoVADis [8] (Aide à Distance à la Vie Quotidienne pour des personnes âgées atteintes de troubles cognitifs) et CompanionAble [9] sont deux projets destinés aux personnes atteintes de troubles cognitifs légers. Ils ont pour objectif principal d'assurer une stimulation cognitive du patient assisté par un "robot compagnon" multimédia interactif et mobile. Suivant le projet, le robot peut être autonome et/ou téléopéré par des professionnels. Le deuxième objectif de ces projets est d'assurer la sécurité du patient par la surveillance de ses fonctions physiologiques et son activité à l'aide d'un dispositif porté par le patient et un réseau de capteurs liés à l'environnement. Le robot intervient en cas d'alarme pour lever le doute, facilitant ainsi le travail des aidants tout en assurant le minimum d'intrusion. Le robot est impliqué dans le service qui assure la sécurité du patient et joue un rôle dans la stimulation cognitive. La stimulation des capacités cognitives résiduelles consiste à aider le patient à se repérer dans le temps et dans l'espace, en le sécurisant en cas d'errance, de désorientation et d'angoisse, et en facilitant sa communication avec l'entourage. La sécurité du patient est basée sur un système de télésurveillance avec un minimum d'intrusion à domicile qui a pour objectif de permettre à des personnes dépendantes ou fragilisées de rester chez elles et d'être aidées ou "médicalisées" à distance de manière plus réactive en cas d'urgence. La fusion de données provenant de différents capteurs ambiants permet de minimiser le taux de fausses alarmes.

Le projet RoboDOMO, destiné aux personnes âgées isolées est un projet qui a pour ambition de produire un support de lien social entre la personne âgée et l'extérieur. Initié par le MBDS [10] de l'Université Nice-Sophia Antipolis, RoboDOMO combine plusieurs technologies des STIC, télécommunication, informatique distribuée et traitement vidéo. RoboDOMO est un robot mobile qui reconnaît l'environnement de l'habitat et peut se déplacer de manière autonome dans la maison. Mais l'objectif initial du projet est de démontrer la faisabilité du pilotage à distance d'un robot depuis un téléphone multimédia connecté à un réseau de téléphonie mobile. Le téléphone 3G permet de piloter le déplacement du robot et recevoir des données audio et vidéo provenant de la caméra fixée au robot lorsque l'option "vidéoconférence" est activée. Il peut aussi être piloté par une télécommande Bluetooth, ou via une interface. Grâce à des capteurs embarqués comme des capteurs de bruits, de présence, de fumée ou des capteurs de dégâts des eaux, il présente un large choix de fonctionnalités de télésurveillance.

8. http ://quovadis.ibisc.univ-evry.fr/
9. http ://www.companionable.net/
10. MBDS : Mobiquité, Base de Données et intégration de Systèmes, http ://www.mbds-fr.org/

I.7 Principaux usages et problématiques de l'assistance ambiante

Le tableau à la page suivante donne un aperçu de ce que la coopération de la domotique, de la robotique et des technologies de l'information et de la communication permet d'imaginer en termes de services rendus à la personne, notamment en perte d'autonomie. Cette coopération ouvre de nouvelles problématiques de recherche. La revue de projets représentatifs permet de dégager les questions que se pose cette communauté récente, par nature pluridisciplinaire. Pour comparaison, la dernière colonne du tableau cite quelques projets de robotique d'assistance à la personne qui n'intègrent pas la dimension intelligence ambiante afin d'illustrer l'évolution des usages et des problématiques apportées par la robotique ubiquitaire.

Nom des projets	COHAND, QuatrA et DANAH	QuoVADis, ROBODOMO, CompanionAble, ROBOCARE	Gerhome, DOMUS	ANSER, URUS, PEIS, iDorm	ROBOTS@HOME, CARE-O-BOT, ARPH
Usage 1	Aide à la personne à mobilité réduite	Aide à la personne âgée avec ou sans troubles cognitifs	Aide à la personne âgée avec ou sans troubles cognitifs	Amélioration de la vie quotidienne dans les zones urbaine, publique et privé	Aide à la personne à domicile et/ou à mobilité réduite
Usage 2	Aide à l'opérateur qui configure l'habitat de la personne à mobilité réduite	Aide à l'aidant (famille ou professionnels)		Le projet ANSER se focalise sur la télésurveillance	
Problématique 1	Création de services fiables à la demande suivant l'handicap de la personne, modularité des services	Intégration de techniques et de services pour le maintien à domicile	Conception de dispositifs informatiques qui communiquent entre eux, analysent les situations et réagissent en fonction d'elles	Autonomie d'un ou plusieurs robots assistée par l'intelligence ambiante	Autonomie propre du robot, Fusion de données multicapteurs
Problématique 2	Reconfiguration	Fusion de données multicapteurs	Fusion de données et apprentissage	Autonomie propre d'un ou plusieurs robots	Aide technique (cas d'un usager handicapé)
Problématique 3	Navigation autonome de fauteuil, Flot de conceptions	Navigation du robot autonome ou téléopéré par l'aidant		Apprentissage, téléopération	Téléopération du robot d'assistance par le patient
Commentaires	Le changement d'échelle est aussi prévu afin que le système puisse aussi équiper des institutions	Les projets QuoVADis et CompanionAble ajoutent à la stimulation cognitive, le service de télévigilance	Ces projets s'appuient sur le profil des utilisateurs et l'analyse de leurs habitudes de vie pour adapter la réponse du système	Chaque robot est considéré comme un élément diffus de l'environnement	L'aide technique supplée un handicap physique (mobilité et/ou manipulation)

Le rôle attribué au robot dans ces projets de service à la personne est d'automatiser des tâches dans un environnement humain standard. Jusqu'à récemment, la communauté des roboticiens s'était focalisée sur l'autonomie du robot, celui-ci n'utilisant que ses propres capacités perceptives, actives et cognitives pour se déplacer dans l'environnement. Depuis quelques années, l'alternative qui a émergée autour du concept de robot ubiquitaire, part du principe que la perception, l'action et l'intelligence sont distribuées dans un environnement ambiant constitué d'objets communicants plus ou moins complexes. Les interactions entre les objets permettent d'étendre les capacités et la fiabilité du robot présent dans l'environnement ambiant. Ce sont principalement les mécanismes de collaboration et de coopération entre les objets ambiants et le robot qui permettent l'amélioration de la qualité du service rendu à l'utilisateur humain par le robot. Les projets de recherche sur la robotique ubiquitaire sont variés aussi bien en termes de services rendus à la personne que d'objectifs scientifiques et techniques. La robotique ubiquitaire doit relever des défis dont les réponses réclament des compétences pluridisciplinaires. L'un des efforts principaux de la recherche porte sur la couche logicielle appelée généralement "middleware". Les délimitations floues du périmètre de cette couche logicielle se traduisent par le fait que, selon les projets, le rôle qui lui est dévolu est plus ou moins étendu et complexe. Une étude [Nader2008] dresse une liste des fonctions qu'elle pourrait (devrait) assurer :

1. Environnement de développement [Enderle2001], [Songmin2007].
2. Gestion de l'interopérabilité [Cote2006], gestion des ressources.
3. Ensemble de services souvent utilisés par les robots [Cote2006], [Kranz2006].
4. Autodécouverte, autoconfiguration et autoadaptation à la situation [Nader2008].
5. Intégration de composants dédiés et de ressources bas-niveaux [Nader2008].

Nous nous appuierons dans le paragraphe suivant sur cette liste pour mieux préciser les contributions de la thèse.

I.8 Contributions de la thèse

Nous nous sommes appuyés sur un scénario d'usage extrait du projet QuoVADis afin de présenter les questions auxquelles la thèse a tenté de proposer des réponses. Ce scénario jouera, tout au long de ce mémoire, le rôle de fil rouge pour conduire notre réflexion.

I.8.1 Scénario d'usage

Ce scénario a été défini avec le SAMU-92/AP-HP (Assistance Publique- Hôpitaux de Paris).
Usage : Télévigilance dans le contexte du maintien à domicile d'une personne en perte d'autonomie.

Objectif : Levée de doute après alarme.
Description : Un signal émis depuis le domicile de Mme Dubois parvient soudain au centre de téléalarme,. L'opérateur se connecte aussitôt aux données correspondantes qui lui sont accessibles, et constate que le système détecte bel et bien une chute (capteur de mouvement, analyseur d'image ou détecteur de chute porté par la personne). Pour conforter ce diagnostic, le rapport d'activité de Mme Dubois révèle un brusque arrêt de toute activité dans l'habitat. L'opérateur prend la décision de se mettre en liaison avec Mme Dubois :
La réponse **se fait immédiatement entendre** :
Mme Dubois est effectivement tombée et éprouve des difficultés à se remettre debout. "Mme Dubois, je vais appeler votre parrain pour qu'il vienne vous aider. Vous êtes certaine que tout va bien ?". Pour précision, l'opérateur dispose d'une liste de correspondants, appelés "parrain", dont il a les coordonnées, et qui peuvent intervenir chez cette personne.
C'est heureusement une chute sans conséquence grave. Dans le cas inverse, ou si Mme Dubois n'avait pas répondu, **l'opérateur aurait eu suffisamment d'éléments pour estimer la gravité de la situation, et aurait pris la décision de contacter le service d'urgence**. Le service d'urgence, après une nouvelle tentative de communication, aurait eu à disposition un panel d'outils plus large pour agir : l'accès aux données médicales de Mme Dubois, la connaissance précise de son habitat et de sa localisation, l'accès à des images en temps réel de l'intérieur du domicile, aurait confirmé la gravité de cette chute et aurait permis une intervention appropriée.
Après discussion avec le SAMU, ce scénario idéal pose des problèmes opérationnels et éthiques qui peuvent justifier l'utilisation d'un robot au domicile de cette personne.

1. *se fait immédiatement entendre :* sauf si la personne est incapable de répondre,
2. *l'opérateur disposerait suffisamment d'éléments pour estimer la gravité de la situation :* généralement non, le thérapeute distant aimerait bénéficier d'un contact audiovisuel pour préciser son diagnostic et les moyens d'intervention,
3. *l'accès à des images en temps réel :* mettre des caméras dans chaque pièce est une solution très intrusive. Un robot muni d'un dispositif audiovisuel, activé essentiellement en cas de besoin semble du point de vue éthique plus "léger". D'autres éléments peuvent justifier le choix du robot plutôt qu'un réseau de caméras. Le robot pouvant se rapprocher de la personne, l'opérateur a une meilleure vision de la scène. Nous avons par exemple montré, lors d'essais grandeur nature avec le SAMU, qu'il est possible de vérifier que la personne respire ; ce qui est plus difficile, voire impossible avec des caméras d'usage courant, situées au plafond. Un autre argument est d'éviter le déploiement d'un réseau de caméras pour des situations temporaires ; par exemple une surveillance à titre préventif après un retour d'hospitalisation.

Le robot, par ses capacités de mobilité et de perception peut participer à la levée de doute. Pour cela il doit être capable de se déplacer dans le domicile du patient. Dans le projet QuoVADis , terminé en juin 2011, le robot était téléopéré par l'opérateur distant du centre de télévigilance. Des évaluations en situations quasi réelles ont montré la validité de cette approche. Toutefois, afin de répondre à des demandes `simultanées` [11] , il s'avère nécessaire d'automatiser la recherche de la personne par le robot à l'intérieur du domicile. Cette recherche de la personne pourrait être activée dès la réception d'une alarme.

I.8.2 Problématique de la thèse

En termes d'usage, dans le scénario de télévigilance présenté précédemment, le robot participe à la levée de doute. La réussite de la mission dépend de la capacité du robot à se déplacer d'un point de départ vers une destination de façon fiable, et ce quelles que soient les situations rencontrées. Pour effectuer ce déplacement, le robot a besoin de se localiser régulièrement. Malgré de très nombreux travaux, la fiabilité de cette fonction n'est toujours pas assurée, au jour d'aujourd'hui, dans un habitat non adapté.

En termes de problématique scientifique, les travaux de la thèse ont pour ambition de contribuer à fiabiliser la fonction localisation dans l'habitat standard d'une personne, à domicile ou en institution, grâce à la coopération entre le robot et les objets communicants présents dans l'environnement tels que les capteurs ou les actionneurs (contribution au point 3 de la liste du paragraphe I.7). Une des difficultés est que l'environnement ambiant est dynamique. Les capteurs ambiants peuvent être dynamiquement disponibles/indisponibles pour la localisation du robot selon le contexte. L'autoadaptation en terme de découverte et d'utilisation des ressources, ici le réseau multicapteur, est une caractéristique dont doit être dotée le robot. Le mécanisme d'autoadaptation à la situation et l'architecture informatique capable de l'accueillir constituent la deuxième contribution de ce travail (contribution au point 4 de la liste du paragraphe I.7).

Dans le contexte du scénario décrit précédemment, le mécanisme d'adaptation doit prendre en compte deux contraintes. La première est liée à l'obligation de résultat. Le mécanisme d'adaptation doit apporter une réponse face à des situations diverses voir imprévues afin qu'au final le robot réussisse sa mission qui est de se déplacer d'un point A à un point B, si possible en tenant compte du degré d'urgence. L'autre contrainte est éthique, dimension très débattue actuellement dans les projets de robotique d'assistance à domicile. L'intrusion d'une intelligence ambiante capable d'agir et de percevoir en présence de la personne occasionne une gêne tout à fait légitime. Il est indispensable de limiter cette

11. Le centre de télévigilance lié au SAMU 92 gère environ 15000 abonnés. Il est courant qu'un opérateur reçoive plusieurs appels simultanément.

gêne autant que possible. C'est la raison pour laquelle nous avons introduit la notion de niveau d'intrusion du système dynamiquement modulé en fonction du profil du patient et de la situation, notamment le degré d'urgence nécessaire pour l'intervention.

I.9 Conclusion

La rencontre de l'intelligence ambiante et de la robotique d'assistance a donné naissance à ce que nous avons appelé robotique ambiante d'assistance car destinée à assister une personne en perte d'autonomie. Elle s'appuie sur l'existence de réseaux d'objets communicants présents dans l'environnement de la personne pour décliner un ensemble de services et de téléservices destinés à faciliter la vie quotidienne de cette personne et de son entourage. Un, voir plusieurs robots peuvent être présents dans cet environnement.
Une communauté scientifique récente s'est construite autour de la robotique ubiquitaire. Tandis que les robots des générations précédentes ont été conçus pour réaliser des tâches spécifiques et construits en tant qu'unité indépendante, la nouvelle génération vise l'ubiquité. L'autonomie du robot est, dans ce cadre, obtenue par une interaction étroite entre le robot et l'environnement ambiant communicant. Jusqu'à ces dernières années le robot évoluait dans un environnement plutôt hostile qui ne lui facilitait pas la tâche. Dans le contexte de l'intelligence ambiante, les objets communicants de l'environnement peuvent jouer un rôle "facilitateur" en aidant le robot à se localiser, naviguer, rechercher un objet. Inversement, le robot peut être vu comme un objet communicant qui est mis à contribution par des services autres que l'assistance à la personne en perte d'autonomie.

Dans ce chapitre, nous avons défini le contexte en termes d'usages et de problématiques. Nous avons ensuite introduit ce qui est appelé par la communauté, *middleware*, qui est la couche logicielle qui supporte les interactions des différents sous-systèmes composant un environnement ambiant. Les contours et les fonctions attribués au middleware sont très variables selon les projets, il est toutefois possible de faire émerger les principaux rôles qu'on peut en attendre. Nous avons situé les problématiques auxquelles la thèse s'est intéressée par rapport aux items d'une liste de ces rôles.

Les trois questions principales auxquelles nous avons cherché à apporter des éléments de réponse sont :

- La localisation du robot en environnement intérieur par coopération avec un environnement ambiant. Comment obtenir un résultat de localisation garanti, sous certaines hypothèses à définir, en présence d'un réseau de capteurs hétérogènes dont le nombre et les caractéristiques sont variables ? L'idée est que la connaissance fiable de sa localisation, même si celle-ci est dégradée (imprécise), permettra au robot d'adapter sa stratégie de déplacement en fonction de la situation.

- L'adaptation au contexte. Quel(s) mécanisme(s) d'adaptation pour gérer : i) la variété et la disponibilité des capteurs, ii) les contraintes liées au service rendu à la personne (degré d'intrusion, degré d'urgence ...) ?
- Une architecture informatique capable de gérer la coopération entre le robot avec l'environnement ambiant et d'accueillir ce(s) mécanisme(s) d'adaptation.

Chapitre II

Localisation par coopération robot environnement

Objectifs du chapitre

L'objectif à terme de la méthode présentée dans ce chapitre est de localiser avec certitude le robot dans une zone de l'habitat. Les dimensions de cette zone peuvent être très variables. Ces dimensions dépendent des besoins de la tâche, voire de l'enchaînement de tâches, que prévoit de réaliser le robot à cet instant, mais découlent aussi de facteurs comme la dispersion des capteurs dans l'environnement, la qualité des mesures disponibles. Nous partons du principe qu'une bonne connaissance de sa situation, la certitude d'être dans cette zone, permettra au robot d'adopter la meilleure stratégie possible pour assurer sa mission qui est de retrouver la personne dans son habitat. Aussi, la position et l'orientation du robot doivent être données avec une précision garantie. C'est moins la précision que la certitude que le robot est localisé dans cette zone qui importe.
Avant de décrire le contenu du chapitre, il est important de préciser que l'algorithme qui découle de cette approche sera intégré aux comportements de certains agents de l'architecture de coopération robot-environnement proposée dans le chapitre suivant. C'est un maillon important qui permettra d'évaluer les capacités d'adaptation de cette architecture et les mécanismes de coalition et de négociation.
En outre, cette approche de localisation a pour ambition d'utiliser les mesures provenant d'un réseau de capteurs, pouvant être très variés, situés dans l'environnement ou sur le robot, elle n'a pas vocation, du moins dans sa version de base, à répondre à tous les cas de figure. Nous allons préciser le périmètre de son utilisation. La localisation du robot est obtenue par multilatération, c'est-à-dire à partir essentiellement des mesures angulaires. Les mesures sont issues de capteurs du robot et de l'environnement. Les mesures sont des angles connues avec une certaine incertitude. Si le capteur est localisé dans l'environnement, la mesure est l'angle formé par la droite passant par ce capteur et le robot avec une

droite de référence. Si le capteur est localisé sur le robot, la mesure est l'angle formé par la droite passant par ce capteur et un marqueur localisé dans l'environnement avec l'axe principal du robot.
Nous avons défini plusieurs contraintes auxquelles cette méthode doit répondre afin de lui donner la capacité d'accepter une forte diversité de capteurs, du plus fruste au plus complexe, sous l'hypothèse que le capteur fournit une mesure angulaire :

- un nombre variable de mesures ;
- des mesures hétérogènes au sens métrologique du terme (précision, portée) ;
- une connaissance statistique limitée sur les mesures. Généralement l'erreur de mesure n'est connue que sous la forme d'une tolérance ;
- une localisation des capteurs/marqueurs par rapport au repère de l'environnement ou du robot ;
- une dispersion plus ou moins importante des capteurs et des marqueurs dans l'environnement.

On peut ajouter à cette liste que l'algorithme devrait pouvoir fournir un résultat de localisation, même partiel, dès qu'une donnée mesurée est exploitable.

Le chapitre est organisé comme suit. Il commence par une étude bibliographique situant ces travaux par rapport à ceux portant d'une part sur la localisation et d'autre part sur l'analyse par intervalle. Suit une présentation de la méthode de localisation basée sur la multilatération qui précise les principales caractéristiques du problème. Le chapitre se poursuit par une introduction détaillée de l'analyse par intervalle qui sera l'outil mathématique utilisé pour résoudre le problème de localisation tel que nous l'avons défini. La quatrième partie du chapitre décrit la résolution du problème et sa mise en œuvre. Le chapitre se termine par un ensemble de simulations destinées à évaluer l'algorithme. Il est à noter que l'algorithme a été testé en environnement réel. Le protocole et les résultats expérimentaux sont reportés au chapitre IV "Mise en œuvre et évaluations".

II.1 État de l'art : Principe de la localisation

Une des difficultés de la localisation est de déterminer et de modéliser les erreurs de mesure. L'erreur de mesure est définie comme la différence entre la valeur mesurée et la valeur exacte de la grandeur. La caractérisation fine de ces erreurs, envisageable dans un contexte de laboratoire de recherche, l'est difficilement si on regarde la diversité des capteurs qui peuvent être intégrés à un réseau dans un contexte d'intelligence ambiante. Il existe deux approches principales pour modéliser ces erreurs. [Jaulin2000a] compare deux approches statistiques permettant de représenter et de manipuler les erreurs de mesures comme des variables incertaines : l'approche probabiliste et l'approche ensembliste. L'approche probabiliste, la plus répandue, représente la variable par une densité de probabilité qui donne à la fois un ensemble mais aussi une information sur la manière dont

est distribuée cette variable aléatoire sur cet ensemble. L'approche ensembliste représente cette variable uniquement par un ensemble appelé domaine. La représentation ensembliste est donc plus pauvre mais elle permet des calculs plus aisés sur les variables aléatoires. Elle demande moins de connaissances statistiques sur les variables et permet de traiter une classe beaucoup plus grande de problèmes. Quand l'erreur de mesure sur une donnée expérimentale n'est connue que sous la forme de tolérance, ce qui est souvent le cas pour les capteurs ou réseau de capteurs utilisés en domotique et plus généralement dans un contexte d'intelligence ambiante, et non en termes de densité de probabilité, l'approche ensembliste est une approche relativement intuitive pour traiter les mesures. Dans le cas contraire, et si de plus le problème est linéaire et gaussien, cette approche ne se justifie pas par rapport à l'approche probabiliste.

Parmi les classes de problèmes qui peuvent être résolus par l'approche ensembliste, on trouve les algorithmes *branch and bound* pour la recherche de solutions de problèmes non-linéaires. Le résultat est garanti, sous réserve de mesures non aberrantes, c'est-à-dire que la solution contient surement la valeur ou l'ensemble recherché. Il faut noter que les méthodes formelles permettent aussi d'obtenir des résultats garantis mais très rarement en temps réel et sont donc inadaptées au domaine des systèmes embarqués, notamment de la robotique mobile.

L'approche ensembliste reste peu utilisée dans le milieu des sciences pour l'ingénieur et seulement très récemment en robotique mobile. Dans le domaine de la robotique mobile, [Jaulin2002] s'est intéressé à la localisation d'un robot à partir de mesures issues de capteurs ultrasonores en utilisant l'analyse par intervalle et en proposant un traitement des valeurs aberrantes sous certaines conditions. [Drocourt2002] utilise l'analyse par intervalle pour modéliser l'imprécision de mesures issues de deux capteurs omnidirectionnels. Ces travaux utilisent uniquement les mesures fournies par les capteurs du robot pour la localisation. Cette idée avait été appliquée par [Léveque1997] pour localiser un véhicule à partir de données télémétriques imprécises.

On peut citer aussi, dans le domaine des véhicules, les travaux qui utilisent différentes sources de mesures extérieures ou bien liées au véhicule. [Gning2006] s'est intéressé à la fusion multisensorielle ensembliste par propagation de contraintes sur les intervalles de mesure provenant de l'hybridation d'un système GPS/gyromètre/odométrie. Vincent Drevelle [Drevelle2010] s'est focalisé sur les méthodes robustes ensemblistes pour une localisation multi-sensorielle dite hautement intègre qui s'appuie sur les travaux de Jaulin pour assurer la robustesse [Jaulin2002], plus précisément l'algorithme RSIVIA qui permet le calcul de solutions en tolérant un nombre q de valeurs aberrantes.

Dans [Reynet2009] décrit une application de *Quimper*, un outil puissant capable de résoudre sans approximations un ensemble d'équations non-linéaires telles que les équations hyperboliques de localisation d'un mobile grâce à l'analyse par intervalle et la programmation de contracteurs spécifiques. Le formalisme de Quimper peut aisément mixer dif-

férentes approches de localisation passive comme la multilatération ou la goniométrie.

Bien que les avantages des méthodes probabilistes sont de loin les plus utilisées et les mieux connues, soient nombreux : la richesse des outils de la théorie sur les probabilités, une estimation ponctuelle qui représente le point le plus probable, nous avons opté pour une approche à erreurs bornées basée sur l'analyse par intervalle pour les raisons suivantes. La seule hypothèse à vérifier est que toutes les erreurs sont bornées. Le respect de cette hypothèse est difficile à prouver mais il existe des techniques pour rejeter les mesures aberrantes [Jaulin2009]. Si cette hypothèse est vérifiée alors le résultat est garanti. De plus, comme la dimension de la variable, dans notre cas la position et l'orientation du robot, est inférieure ou égale à trois, le traitement est relativement simple et rapide. Enfin, cette approche permet de prendre en compte simplement les contraintes listées en début de chapitre que nous rappelons brièvement :

- nombre variable de mesures,
- mesures hétérogènes,
- connaissance statistique limitée sur les mesures,
- localisation des capteurs/marqueurs par rapport au repère de l'environnement ou du robot,
- dispersion plus ou moins importante des capteurs et des marqueurs dans l'environnement.

Par rapport aux travaux existants sur la localisation d'un robot mobile ou d'un véhicule qui utilise l'analyse par intervalle, l'originalité de notre approche, comme nous allons tenter de le montrer dans la suite du chapitre, repose sur la capacité à intégrer une grande variété de capteurs, du plus fruste au plus complexe. Il suffit que le capteur délivre une mesure d'angle, voire une zone active (dalle).

II.2 Principe de la localisation par triangulation

Le principe de la triangulation est illustré dans la figure II.1. La mesure des trois angles λ_i, i entier de 1 à 3, entre trois balises B_i, i entier de 1 à 3, permet de connaître la position (x, y) du robot (point A).

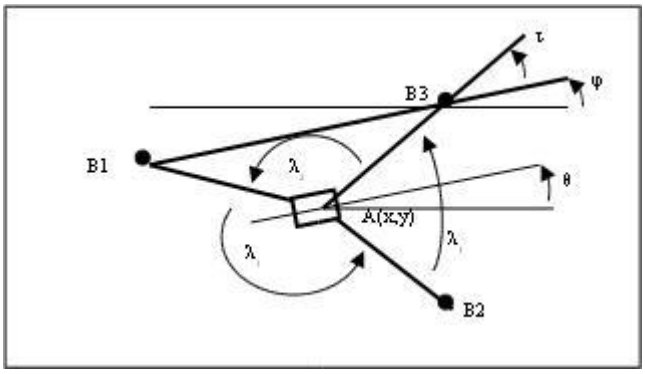

Figure II.1 : Principe de la triangulation

II.2.1 Calcul des coordonnées du point triangulé

Soient :

- les variables, coordonnées du point A (figure II.1), connues dans le repère fixe R_e de l'environnement. c'est aussi le vecteur d'état $x = (x, y)^T$ du robot,
- les mesures, λ_i, i entier de 1 à 3. Les mesures sont effectuées par le robot,
- les données, coordonnées des balises $B_j = (x_j, y_j)$,

une solution pour le calcul des coordonnées du point triangulé A(x,y) prend comme point de départ le centre de gravité du triangle formé par les balises.

$$\begin{aligned} x &= \frac{g_1 x_1 + g_2 x_2 + g_3 x_3}{g_1 + g_2 + g_3} \\ y &= \frac{g_1 y_1 + g_2 y_2 + g_3 y_3}{g_1 + g_2 + g_3} \\ &\text{avec} \\ g_1 &= \frac{1}{\cot\alpha - \cot g\lambda_1} \\ g_2 &= \frac{1}{\cot\beta - \cot g\lambda_2} \\ g_3 &= \frac{1}{\cot\gamma - \cot g\lambda_3} \end{aligned} \tag{II.1}$$

Avec α l'angle formé par $(B_2B_1B_3)$, β l'angle formé par $(B_3B_2B_1)$ et γ l'angle formé par $(B_1B_3B_2)$. Les angles α β γ sont calculés à partir des coordonnées des balises.

$$\alpha = arcos\frac{L_{12}^2 + L_{31}^2 - L_{23}^2}{2L_{12}L_{31}}$$
$$\beta = arcos\frac{L_{23}^2 + L_{12}^2 - L_{31}^2}{2L_{23}L_{12}}$$
$$\gamma = arcos\frac{L_{23}^2 + L_{31}^2 - L_{12}^2}{2L_{23}L_{31}} \quad \text{(II.2)}$$

avec

$L_v = \sqrt{(y_j - y_i)^2 + (x_j - x_i)^2}$ distance entre les balises i et j

II.2.2 Limites de la méthode

Plusieurs limitations sont inhérentes à cette méthode :

- Le point triangulé A ne doit pas appartenir au cercle circonscrit au triangle formé par les trois balises. Cependant, si les balises sont placées sur les murs des pièces de l'habitat, il y a peu de risques que cette configuration survienne sauf peut être dans des pièces non rectangulaires, en forme de L par exemple. Si ce cas se présente, le calcul de la position est impossible.
- Il faut tenir compte de l'ordre des balises afin d'apparier correctement les couples angulaires dans les calculs, par exemple α avec λ_1.
- L'imprécision sur les mesures se traduit, à l'intérieur du triangle formé par les balises, par une imprécision sur les coordonnées du point A et à l'extérieur du triangle par des résultats de plus en plus imprécis voire erronés.
- Les mesures sont acquises du robot. La méthode est différente si les mesures sont acquises à partir de l'environnement.
- La méthode ne permet pas exploiter plus de trois mesures (multiangulation) alors que la précision du résultat pourrait être améliorée quand plus de trois mesures sont disponibles. Afin d'étendre les capacités de la méthode précédente en corrigeant les limitations évoquées ci-dessus, nous proposons l'approche suivante.

II.3 Description de la méthode de localisation par multiangulation basée sur l'analyse par intervalle

Hypothèse sur les capteurs

Dans la suite du chapitre, nous distinguons deux types de capteurs que seront dénommés *marqueur* et *capteur*. Le marqueur, noté M, est placé dans l'environnement et est détecté par le robot. Le capteur, noté C, est placé dans l'environnement et détecte le robot. Chaque capteur ou marqueur est identifié, ce qui permet de faire la distinction entre ces deux types et d'appliquer des traitements spécifiques à leurs mesures. Quel que

soit le type, capteur ou marqueur, le modèle de mesure est un cône à l'intérieur duquel la présence du robot est garantie. Ce modèle est suffisamment général pour englober une grande variété de capteurs tels que détecteur de présence, télémètre laser, télémètre ultrason, télémètre infrarouge, caméra, RFID, ...

Variables et mesures :

Variable : Vecteur d'état $x = (X_R, Y_R, \theta_R)^T$, position et orientation du robot dans le repère R_e de l'environnement. Par rapport au cas précédent le vecteur est de dimension 3.
Mesure : λ_i

Domaine de définition des mesures

$[\lambda_i]$, l'intervalle de définition de la mesure est déduit de l'ontologie capteur qui fournit le $\Delta\lambda_i$ du capteur dont est issu la mesure. C'est l'identificateur associé à chaque capteur ou marqueur, qui permet de retrouver le $\Delta\lambda_i$ dans l'ontologie (cf. chapitre III "Architecture informatique"). L'intervalle est défini de la manière suivante :

$$[\lambda_i] = [\lambda_i - \Delta\lambda_i, \lambda_i + \Delta\lambda_i] \tag{II.3}$$

Données

Les coordonnées du marqueur $M_j = (x_j, y_j)$ ou les coordonnées et l'orientation du capteur de l'environnement $C_i = (x_i, y_i, \theta_i)$ sont connues par rapport au repère R_e de l'environnement. Nous supposerons, dans le cadre de cette thèse, que ces données sont connues avec une grande précision. La méthode proposée peut très facilement intégrer les imprécisions sur ces données. Autre remarque, le marqueur n'a pas d'orientation. On suppose qu'il est visible de n'importe quel point de la pièce s'il est à portée.

Equations :

Deux cas sont à envisager :
Cas 1 : Le robot détecte un marqueur M placé dans l'environnement (figure II.2).

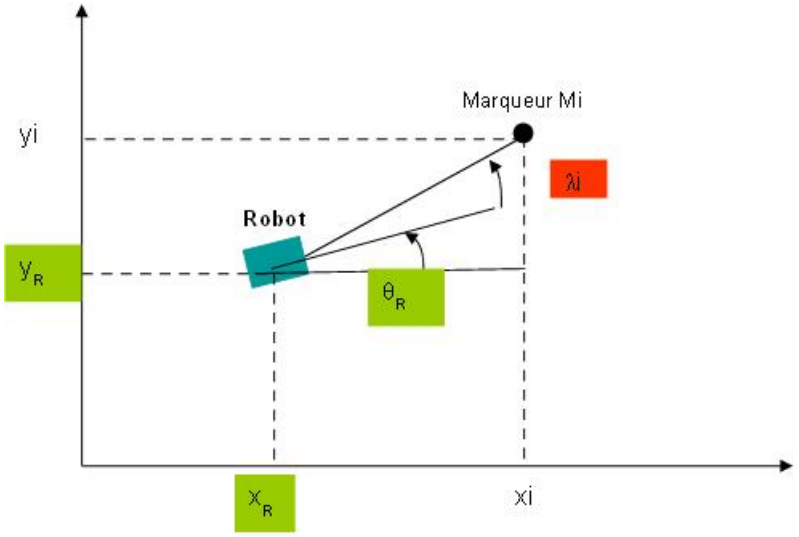

FIGURE II.2 : Le robot détecte un marqueur de l'environnement, noté C

$$\lambda_i = \arctan(\frac{Y_R - y_i}{X_R - x_i}) - \Theta_R \tag{II.4}$$

avec (x_i, y_i), coordonnées du marqueur M_i dans le repère lié à l'environnement (figure II.2).

Cas 2 : Le robot est détecté par un capteur de l'environnement, noté C ().

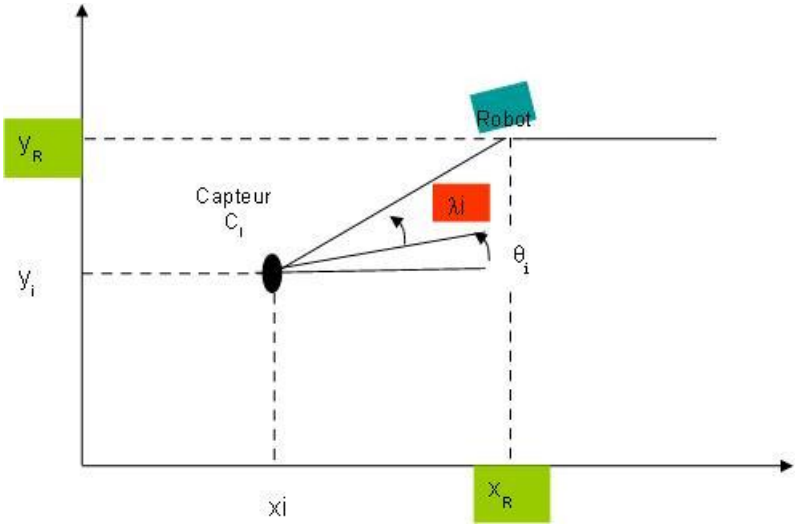

FIGURE II.3 : Le robot est détecté par un capteur de l'environnement, noté C

$$\lambda_i = \arctan(\frac{Y_R - y_i}{X_R - x_i}) - \Theta_i \quad \text{(II.5)}$$

avec (x_i, y_i), les coordonnées et Θ_i l'orientation du capteur C_i dans le repère lié à l'environnement.

Placement des capteurs ou des marqueurs dans l'environnement.

Les capteurs et les marqueurs de l'environnement peuvent être placés n'importe où dans la pièce (figure II.4) : sur les murs, près d'éléments particuliers comme une porte, au plafond.

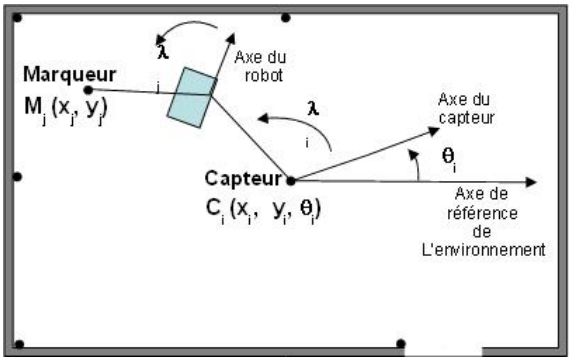

Figure II.4 : Placement des capteurs ou des marqueurs, vue de dessus. Un symbole • représente un capteur ou un marqueur.

La figure permet de déduire que quelle que soit la position et l'orientation du robot et le type de mesure (marqueur ou capteur), la mesure $\lambda_i \in [0, 2\pi]$.

Principe de la méthode de localisation

Le principe de localisation consiste à inverser un ensemble d'équations en utilisant l'analyse par intervalle qui permet de prendre en compte l'imprécision des mesures. L'analyse par intervalle donne un résultat garanti au sens ou l'ensemble des solutions à un problème appartient de façon garantie à l'intervalle solution. Nous avons opté pour cette approche afin d'obtenir une localisation garantie. Pour ne pas surcharger l'exposé de la méthode, il été choisi de ne pas intégrer l'imprécision sur la position des marqueurs et la position et l'orientation des capteurs de l'environnement. Mais avec cette approche, cette intégration ne pose aucune complication méthodologique.

Les principales caractéristiques du problème sont les suivantes :

- Système d'équations non linéaires.
- Deux ou trois paramètres inconnus par équation selon l'origine de la mesure (environnement ou robot).

– Nombre variable d'équations du système.
– Des mesures entachées d'erreur.
– Une mesure par équation.
– Domaine de recherche initial pouvant être très imprécis.
– Contrainte temps réel (de l'ordre de la seconde).

Limitation : une hypothèse forte est que la mesure appartient à l'intervalle d'incertitude. Nous n'avons pas traité dans le cadre de cette thèse la violation de cette hypothèse. La littérature propose des solutions, comme par exemple la relaxation de contraintes, pour le traitement de mesures aberrantes qui peuvent répondre partiellement à ce problème.

II.4 Analyse par intervalle

L'analyse par intervalles [Moore1979] consiste à représenter les nombres réels ou entiers par les intervalles qui les contiennent. Cette idée a permis le développement d'algorithmes dont les résultats obtenus sont garantis, par exemple pour trouver toutes les solutions d'un ensemble d'équations non-linéaires [Jaulin1993], [Jaulin2009], [Kieffer2000]. L'analyse par intervalle a été développée dans les années 1970 pour quantifier l'erreur due à la précision du codage des nombres sur le résultat d'un calcul numérique.
Pour le problème qui nous intéresse, on peut ainsi tenir compte d'une erreur de mesure en remplaçant une valeur mesurée x avec une incertitude ϵ par l'intervalle $[x-\epsilon, x+\epsilon]$ et obtenir une localisation avec un résultat garanti c'est-à-dire qui contient l'ensemble cherché.

II.4.1 Notion sur la théorie des ensembles

Considérons deux ensembles $\mathcal{X}$ **et** $\mathcal{Y}$.

Définitions 3.1

$$\mathcal{X} \cap \mathcal{Y} = \{x | x \in \mathcal{X} \ \textbf{et} \ \ y \in \mathcal{Y}\} \ \textit{intersection}$$
$$\mathcal{X} \cap \mathcal{Y} = \{x | \mathcal{X} \in \mathcal{X} \ \textbf{ou} \ y \in \mathcal{Y}\} \ \textit{union}$$
$$\mathcal{X} \subset \mathcal{Y} = \{x \in \mathcal{Y} | y \in \mathcal{Y}\} \ \textit{union}$$
$$\mathcal{X} \times \mathcal{Y} = \{(x, y) \in \mathcal{X} \text{ et } y \in \mathcal{Y}\} \ \textit{produit cartésien}$$

Définition 3.2
Un intervalle $[x]$ est un ensemble de $\mathbb{R}$ qui peut s'écrire sous la forme

$$[x] = \{x \in \mathbb{R} | x^- \leq x \leq x^+, x^- \in \mathbb{R}, x^+ \in \mathbb{R}\} \tag{II.6}$$

dans laquelle x^- et x^+ sont respectivement des bornes inférieures et supérieures de $[x]$. La notation utilisée pour un intervalle est $[x^-, x^+]$.
Nous pouvons aussi définir :

Définition 3.3
L'ensemble des intervalles de $\mathbb{R}$ est

$$\mathcal{R} = \{\mathcal{A} \subset \mathbb{R}, \mathcal{A} = \{x \in \mathbb{R} | a \leq x \leq b, a \in \mathbb{R}, b \in \mathbb{R}\}\} \tag{II.7}$$

Définition 3.4

$$\text{La largeur d'un intervalle } [x] \text{ est } \omega([x]) = x^+ - x^- \tag{II.8}$$

$$\text{Le centre d'un intervalle } [x] \text{ est c}([x]) = (x^+ + x^-)/2 \tag{II.9}$$

II.4.2 Vecteur d'intervalles, produit cartésien d'intervalles, boîte, pavé

Un pavé ou vecteur d'intervalles est un sous-ensemble borné de $\mathbb{R}^n$, défini par le produit cartésien de n intervalles réels :

Définition 3.5

$$[\mathbf{x}] = [\mathbf{x}^-, \mathbf{x}^+] = [x_1^-, x_1^+] \times [x_2^-, x_2^+] \times .... \times [x_1^-, x_1^+] = [x_1] \times [x_2] \times ... \times [x_n] \tag{II.10}$$

Un pavé approxime un vecteur incertain. Chaque composante du vecteur appartient à un intervalle. Cette représentation par pavé entraîne un pessimisme dû à l'effet d'enveloppement. Un ensemble $\mathcal{S}$ de forme quelconque peut être encadré par un pavé $[\mathbf{x}]$. L'encadrement réalisé contient tous les points de $\mathcal{S}$ mais aussi tous les points du pavé n'appartenant pas à $\mathcal{S}$.

II.4.3 Opérations sur les intervalles

Opérations élémentaires
Un ensemble d'opérations élémentaires peut être effectué sur les intervalles : c'est la généralisation des opérations élémentaires sur les réels.

Définition 3.6

$$\begin{aligned} &\text{Si } \diamondsuit \in \{+, -, *, /, min, max\} \\ &\text{et si } [x] \text{ et } [y] \text{ sont deux intervalles} \\ &\text{alors } [x]\diamondsuit[y] = [\{x\diamondsuit y | x \in [x], y \in [y]\}] \end{aligned} \tag{II.11}$$

Exemples 3.1

$$\begin{aligned} [x] + [y] &= [x^- + y^-, x^+ + y^+] \\ [x] * [y] &= [min(x^-y^-, x^-y^+, x^+y^-, x^+y^+), max(x^-y^-, x^-y^+, x^+y^-, x^+y^+)] \end{aligned} \tag{II.12}$$

Une généralisation de ces règles est possible avec l'ajout des bornes infinies. En effet les intervalles permettent de prendre en compte les valeurs infinies, la division par un intervalle contenant zéro en particulier.

Exemples 3.2

$$\begin{aligned} \frac{1}{y} &= \emptyset \text{ si } [y] = [0, 0] \\ &= [\frac{1}{y^+}, \frac{1}{y^-}] \text{ si } 0 \notin [y] \\ &= [\frac{1}{y^+}, +\infty] \text{ si } y^- = 0 \text{ et } y^+ > 0 \\ &=]-\infty, \frac{1}{y^-}] \text{ si } y^- < 0 \text{ et } y^+ = 0 \\ &=]-\infty, +\infty] \text{ si } y^- < 0 \text{ et } y^+ > 0 \\ [x]/[y] &= [x] * 1/[y] \end{aligned} \tag{II.13}$$

Plus généralement, l'ajout des bornes $-\infty$ et $+\infty$ permet d'obtenir un résultat quels que soient les singularités et les points pour lesquels les fonctions ne sont pas définies et la possibilité d'explorer $\mathbb{R}$ dans son intégralité pour la recherche de solution.

Fonctions élémentaires

Définition 3.7

Une fonction f de $\mathbb{R}$ dans $\mathbb{R}$ telle que $f \in$ cos, sin, arctan, sqr, sqrt, log, exp, ... est définie sur un intervalle [x] comme suit :

$$\mathcal{R} \to \mathcal{R}, [x] \to [f]([x]) = [(f(x)|x \in [x]] \tag{II.14}$$

Par exemple, $[arctan]([0, +\infty]) = [0, \pi/2[$

II.4.4 Principes de base

II.4.4.1 Fonction d'inclusion

Définition 3.8

Soit f une fonction de $\mathbb{R}^n$ à valeurs dans $\mathbb{R}^m$. La fonction intervalle [f] :

$$\begin{aligned} \mathcal{R}^{\mathrm{n}} \to \mathcal{R}^{\mathrm{m}} &\textit{est une fonction d'inclusion pour } f \textit{ si} \\ &\forall [\mathbf{x}] \in \mathcal{R}^{\mathrm{n}}, f([\mathbf{x}]) \subset [f](\mathbf{x}) \end{aligned} \tag{II.15}$$

Exemple 3.3

Pour la fonction $f : x \to sin(x)$ une fonction d'inclusion triviale est définie de la façon suivante :
$\forall [x] \in \mathcal{R}, [sin]([x]) = [-1, 1]$

Définition 3.9

$$\text{Une fonction d'inclusion}[\mathbf{f}] \text{ est minimale si } \forall [\mathbf{x}] \in \mathcal{R}, f([\mathbf{x}]) = [\mathbf{f}]([\mathbf{x}]) \tag{II.16}$$

Exemple 3.4

La fonction d'inclusion triviale pour la fonction sinus de l'exemple ci-dessus n'est pas minimale sauf si $x \in [-\pi, +\pi]$, pour obtenir une fonction plus intéressante, il faut prendre en compte la monotonie de la fonction sinus.

Définition 3.10

Une fonction d'inclusion $[\mathbf{f}]$ est monotone si $[\mathbf{x}] \subset [\mathbf{y}] \Rightarrow [\mathbf{f}]([\mathbf{x}]) \subset [\mathbf{f}]([\mathbf{y}])$

Exemple 3.5

La fonction carré est divisée en deux parties, $[x] \cap]-\infty; 0]$ où f est décroissante et $[x] \cap]0; +\infty]$ où f est croissante. La fonction finale est alors donnée par l'union des deux intervalles $[(x^+)^2, (x^-)^2]$ et $[(x^-)^2, (x^+)^2]$.

Remarque 1 : Les fonctions périodiques demandent des algorithmes spécifiques qui permettent le calcul de ces fonctions avec des intervalles. L'évaluation de la fonction d'inclusion de fonctions périodiques comme les fonctions trigonométriques est décomposée en domaines dans lesquelles la fonction est monotone. Cette technique qui permet d'obtenir directement une fonction d'évaluation minimale sera mise en œuvre pour la fonction *arctan*.

Remarque 2 : Le résultat d'une série d'opérations avec des intervalles n'est pas forcément minimal. On dit que l'intervalle obtenu est pessimiste. Le pessimisme est dû

principalement à la dépendance et l'enveloppement qui a été déjà mentionné. Ces phénomènes et les techniques qui permettent d'y répondre, sont largement étudiés dans la littérature.

Définition 3.11
Une fonction d'inclusion [**f**] est convergente si :

$$\lim_{k\to\infty} \omega([\mathbf{x}](k)) = 0 \Rightarrow \lim_{k\to\infty} \omega([\mathbf{f}])([\mathbf{x}](k)) = 0 \tag{II.17}$$

Cette propriété de convergence est nécessaire pour démontrer la convergence de l'algorithme ensembliste.

Fonction d'inclusion naturelle
La fonction d'inclusion naturelle est obtenue par le remplacement de chacune des variables de la fonction par son domaine intervalle associé et toutes les opérations sur les réels par leur correspondant intervalle. Le résultat permet d'obtenir aisément une fonction d'inclusion mais ne garantit pas qu'elle soit minimale. Des techniques sont proposées dans la littérature pour obtenir une fonction d'inclusion minimale.

II.4.4.2 Test d'inclusion

Soit le pavé [**x**]. Pour comparer ce pavé avec un autre pavé **[y]**, le recours à un test d'inclusion est nécessaire et permet de traiter les résultats que nous donne l'analyse par intervalles.
Si [**x**] est inclus dans **[y]**, le test est vrai.
S'il existe i tel que $[\mathbf{x}_i]$ et $[\mathbf{y}_i]$ sont disjoints, le test est faux.
S'il existe i tel que $\mathbf{x}_i^- < \mathbf{y}_i^-$ ou $\mathbf{x}_i^- < \mathbf{y}_i^+ < \mathbf{x}_i^+$, le test est indéterminé car [**x**] contient à la fois des valeurs appartenant et n'appartenant pas à [**y**]. Il est pratique d'introduire la notion d'intervalle booléen pour tenir compte du fait que le test pourra être satisfait, non satisfait ou être indéterminé.

II.4.4.3 Sous pavage

Définition 3.12
Un sous-pavage de $\mathcal{R}^n$ est un ensemble de pavés ne se recouvrant pas. Un ensemble $\mathcal{X}$ peut être encadré entre un sous-pavage interne et un sous-pavage externe.

$$\mathcal{X}^- \subset \mathcal{X} \subset \mathcal{X}^+ \tag{II.18}$$

Inversion ensembliste
Soit $\mathcal{X}$ un ensemble de $\mathcal{R}^n$ et une fonction $\boldsymbol{f} : \mathcal{R}^n \to \mathcal{R}^m$, **f** pouvant être non linéaire.

Soit $\mathcal{Y}$ un sous-ensemble de $\mathcal{R}^m$ (par exemple, un sous-pavage) tel que $\mathcal{Y} = \boldsymbol{f}(\mathcal{X})$

Définition 3.13
L'inversion d'ensemble permet de définir :

$$\mathcal{X} = \{\mathbf{x} \in \mathcal{X} | \boldsymbol{f}(\mathbf{x}) \in \mathcal{Y}\} = \boldsymbol{f}^{-1}(\mathcal{Y}) \tag{II.19}$$

Pour tout $\mathcal{Y} \subset \mathcal{R}^m$ et pour toute fonction $\boldsymbol{f}$ admettant une fonction d'inclusion convergente $[\boldsymbol{f}]()$, deux sous-pavages $\mathcal{X}^-$, $\mathcal{X}^+$ tel que :

$$\mathcal{X}^- \subset \mathcal{X} \subset \mathcal{X}^+$$

peuvent être obtenus avec l'algorithme SIVIA (Set Inverter Via Internal Analysis) [Jaulin1993] qui permet, à partir d'un pavé initial relativement large d'obtenir un encadrement de l'ensemble solution $\mathcal{X}$.

Deux tests sont utilisés pour vérifier qu'un pavé $[\mathbf{x}]$ est à l'intérieur ou à l'extérieur de X.

$$[\boldsymbol{f}]([\boldsymbol{x}]) \subset \mathcal{Y} \Rightarrow [\mathbf{x}] \subset \mathcal{X}, \text{le pavé est solution (acceptable)}$$
$$[\boldsymbol{f}]([\boldsymbol{x}]) \cap \mathcal{Y} = \varnothing \Rightarrow [\mathbf{x}] \cap \mathcal{X} = \varnothing, \text{le pavé n'est pas solution (inacceptable)}$$

L'encadrement intérieure $\mathcal{X}^-$ est constituée de l'ensemble des pavés acceptables. Si aucune décision n'est prise pour un pavé $\mathbf{[x]}$, on dit qu'il est ambigu et il est alors partitionné. Les deux pavés générés sont testés. L'algorithme itère tant qu'il reste un pavé ambigu. La condition d'arrêt de cet algorithme récursif est l'obtention d'une largeur $\omega\mathbf{[x]}$ inférieure à une valeur ϵ précisée par l'utilisateur.

L'approximation extérieure est définie par :

$$\mathcal{X}^+ = \mathcal{X}^- \cup \Delta\mathcal{X}$$

Où $\Delta\mathcal{X}$ est l'ensemble des pavés ambigus dont la taille est inférieure à ϵ. Les algorithmes d'inversion ensembliste peuvent être résumés en trois étapes :

1. le choix d'un pavé initial $[x_0]$ supposé contenir les solutions recherchées ;
2. l'élaboration d'un test d'inclusion, qui permet de déterminer si un pavé est acceptable, inacceptable ou ambigu. Le test d'inclusion permet de comparer l'évaluation $[\boldsymbol{f}]([\mathbf{x}])$ avec $\mathcal{Y}$.
3. le partitionnement des pavés ambigus afin de réduire le volume de $\Delta\mathcal{X}$.

Si le pavé courant $[x]$ contient à la fois des points solutions et non solutions, l'algorithme est alors relancé en découpant en deux, puis en évaluant l'image par $\boldsymbol{f}$ des pavés $[x_1], [x_2]$. La condition d'arrêt de cet algorithme récursif est l'obtention d'une largeur $\omega[\mathbf{x}]$ inférieure à une valeur ϵ donnée par l'utilisateur (algorithme 1).

si $[\mathbf{f}]([\mathbf{x}_0]) \subset \mathcal{Y}$ **alors**
  $[\mathbf{x}_0]$ est solution
sinon si $[\boldsymbol{f}]([\mathbf{x}]) \cap \mathcal{Y} = \varnothing$ **alors**
  $[\mathbf{x}_0]$ n'est pas solution
sinon si $\omega([\mathbf{x}_0]) < \epsilon$ **alors**
  $[\mathbf{x}_0]$ est ambigu
sinon
  bisect$(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2)$
  SIVIA $(\mathbf{x}_1)$
  SIVIA $(\mathbf{x}_2)$
finsi
finsi
finsi

Algorithme 1: SIVIA $([x_0])$

La complexité de SIVIA est exponentielle en fonction de la dimension du vecteur $[x]$. Selon [Jaulin1993], le nombre N de bissection est inférieur à

$$N = (\frac{\omega|(x_0)|}{\epsilon} + 1)^n$$

avec $[\mathbf{x}_0]$ pavé initial de recherche, la précision ϵ qui fixe la dimension minimale des pavés et n la dimension du vecteur $[\mathbf{x}]$.
Dans la pratique, cet algorithme est utilisable pour des applications dans lesquelles le facteur temps est important, quand la dimension du vecteur paramètre ne dépasse pas trois.
Pour des dimensions supérieures du vecteur $[\mathbf{x}]$, SIVIA est associé à des contracteurs dont l'objectif est de limiter le nombre de bissections.
Dans le cas d la localisation d'un robot mobile, le vecteur paramètre $[\mathbf{x}]$ étant de dimension deux ou trois, (position (x, y) et orientation θ, l'algorithme SIVIA se révèle suffisant pour apporter une solution temps réel.

II.5 Application à la localisation en environnement communicant

II.5.1 Estimation à erreur bornée de paramètres, cas 2D

Seule la position (X_R, Y_R) du robot est recherchée. Les deux paramètres ont les mêmes unités. Les éléments du problème sont :
Équations : $f(\mathbf{x}, \mathbf{x}_i) = arctan(\frac{Y_R - Y_i}{X_R - X_i}) - \theta_i$ avec (x_i, y_i) et θ_i du capteur C_i.

Paramètres : $\mathbf{x} = (X_R, Y_R)^T$, Domaine de recherche des paramètres : $[\mathbf{x}]$.

Données capteurs : $\mathbf{x}_i = (x_i, y_i, \theta_i)$ respectivement coordonnées et orientation dans le repère lié à l'environnement.

Mesures : $\lambda_1, \lambda_2, ...\lambda_m$ avec leur intervalle associé $[\lambda_1], [\lambda_2], ...[\lambda_m]$

Ensemble recherché : $\mathbf{S} = \{\mathbf{x} \in [\mathbf{x}], f(\mathbf{x}, \mathbf{x}_1) \in [\lambda_1], f(\mathbf{x}, \mathbf{x}_2) \in [\lambda_2], ...f(\mathbf{x}, \mathbf{x}_m) \in [\lambda_m]\}$
En posant $F(\mathbf{p}) = (f(\mathbf{x}, \mathbf{x}_1), f(\mathbf{x}, \mathbf{x}_2), ..., f(\mathbf{x}, \mathbf{x}_m))^T$ et $[\mathbf{lambda}] = [\lambda_1] \times [\lambda_2] \times ... \times [\lambda_m]$
Alors $\mathcal{S} = [\mathbf{p}] \cap F^{-1}(\mathbf{p})$

Test d'inclusion (Algorithme 2)

si $([\mathbf{f}]([\mathbf{x}], \mathbf{x}_i, t_i)) \subset [\lambda_i])$ **alors**
  $[\mathbf{x}]$ est solution
sinon si $([\mathbf{f}]([\mathbf{x}], \mathbf{x}_i, t_i)) \cap [\lambda_i] = \varnothing$ **alors**
  $[\mathbf{x}]$ n'est pas solution
sinon
  $[\mathbf{x}]$ est ambigu
finsi
finsi

Algorithme 2: Test d'inclusion $([\mathbf{x}], [\lambda_i], \mathbf{x}_i, t_i)$

Les arguments de cette fonction sont : $[\mathbf{x}]$ les paramètres inconnus, $[\lambda i]$ la mesure, $\mathbf{x}_i$ les coordonnées du marqueur ou du capteur et t_i le type, marqueur ou capteur.

Si n mesures sont disponibles à un instant donné, le principe consiste à effectuer n tests d'inclusion, noté T, avec les arguments associés à la mesure traitée i. La fusion des n tests est basée sur la règle suivante (Algorithme 3).

si $T_1 == T_2 == ... == T_n$ **alors**
  $Fusion\_Test = T_i$
sinon si (($T_1 ==$ inacceptable ou ... ou$T_n ==$ inacceptable **alors**
  $Fusion\_Test = inacceptable$
finsi
finsi

Algorithme 3: Règle de fusion de n tests d'inclusion

Avec T_i le résultat du test d'inclusion correspondant à la mesure i et $Fusion\_test$ le résultat de la fusion des n test d'inclusion.

Remarque : Cette règle simple conduit à ne pas obtenir de résultat quand une des mesures est aberrante mais elle permet de valider l'ensemble de la méthode de localisation.

Calcul de la fonction d'inclusion $[arctan(\alpha)]$
Les mesures angulaires $[\lambda_i]$ effectuées par les capteurs appartiennent au domaineă$[0, 2\pi]$. Le $\Delta\lambda_{max}$ peut attendre $\pi/2$ si on veut prendre en compte des capteurs comme les capteurs de présence dont l'incertitude qui peut couvrir un secteur angulaire allant jusqu'à p radians.

Comme le montre la figure II.5, la fonction *arctan()* est discontinue sur le domaine $[0, 2\pi]$. La fonction d'inclusion doit prendre en compte les discontinuités sur tout ce domaine ainsi que les effets de bord au voisinage des bornes du domaine dû au fait que nous manipulons des intervalles de valeurs et non une valeur.

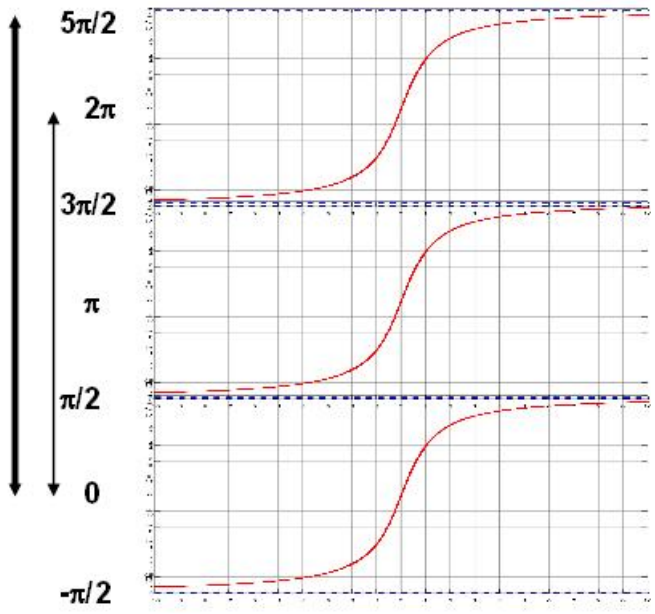

Figure II.5 : Fonction $arctan(\alpha)$ pour $\alpha \in [0, 2\pi]$

Le principe de calcul de la fonction d'inclusion $[arctan\alpha]$ que nous avons retenu pour gérer les discontinuités, est de considérer le signe de $tan(\alpha)$ à partir de ses membres : $N = Y_R - y_i$ et $D = X_R - x_i$ afin de distinguer puis de traiter séparément chaque cas pour le calcul de l'angle α.

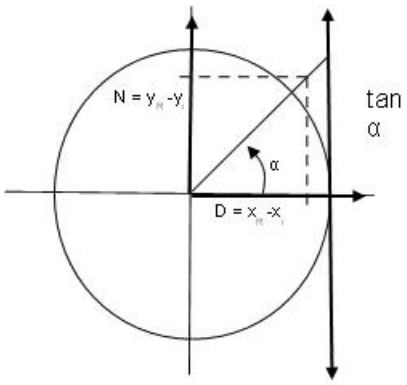

FIGURE II.6 : Fonction $arctan(\alpha)$ pour $\alpha \in [0, 2\pi]$

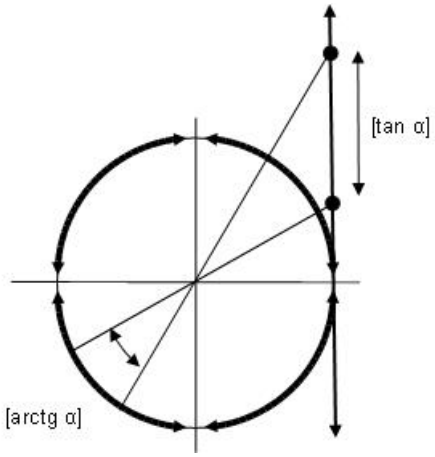

FIGURE II.7 : $[tan\alpha]$ quand L'intervalle appartient à un même quadrant angulaire

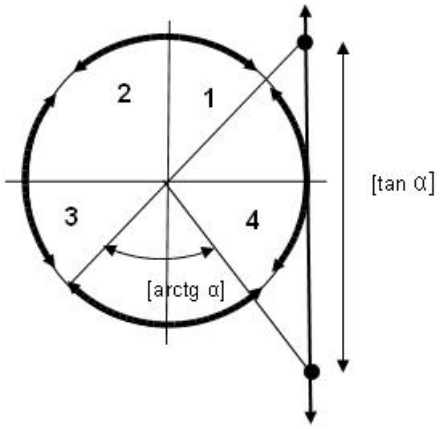

Figure II.8 : $[tan\alpha]$ quand L'intervalle n'appartient pas à un même quadrant angulaire

L'application de ces principes donnent l'algorithme 4

// Notations
// Si = sin i, Ci=cos i, Ti=tan i
// avec i=1 pour la borne inférieure de l'intervalle et i = 2 pour la borne
// supérieure de l'intervalle.
si $C1 == 0$ **alors**
  $C1 == \epsilon$
finsi
si $C2 == 0$ **alors**
  $C2 == \epsilon$
finsi
$\mathbf{v} = [S1/C1, S1/C2, S2/C1, S2/C2]$
$\mathbf{T} = [min\mathbf{v}, max\mathbf{v}]$
//L'intervalle appartient à un même quadrant angulaire
si $(S1 * S2 \geq 0)$ et $(C1 * C2) \geq 0$ **alors**
  si $C1 \geq 0$ **alors**
    si $S1 \geq 0$ **alors**
      angle $= [arctan(T1), arctan(T2)]$
    sinon
      angle $= [arctan(T1) + 2\pi, arctan(T2) + 2\pi]$
    finsi
  sinon
    angle $= [arctan(T1) + \pi, arctan(T2) + pi]$ //L'intervalle appartient à deux quadrants angulaires contigus
  sinon si $S1 * S2 < 0$ **alors**
    // quadrants 4-1 ou 2-3
    si $C1 > 0$ et $C2 > 0$ **alors**
      angle $= [arctan(T1) + 2\pi, arctan(T2) + 2\pi]$
    finsi
    si $C1 < 0$ et $C2 < 0$ **alors**
      angle $= [arctan(T1) + pi, arctan(T2) + \pi]$
    finsi//L'intervalle dépasse deux quadrants
    si $C1 * C2 < 0$ **alors**
      angle $=[0, 2\pi]$
    finsi//Quadrants 1-2 ou 3-4 (C1*C2 < 0)
  sinon
    $\mathbf{t} = \text{sort}(\mathbf{v})$
    si $S1 \geq 0 S2 > 0$ **alors**
      angle $=[arctan(\mathbf{t}(3)), arctan(\mathbf{t}(2)) + \pi]$
    finsi
    si $S1 < 0 S2 \geq 0$ **alors**
      angle $=[arctan(\mathbf{t}(3)) + \pi, arctan(\mathbf{t}(2)) + 2\pi]$
    finsi
  finsi
finsi

Algorithme 4: Calcul de la fonction intervalle $[arctan\alpha]$

II.5.2 Estimation à erreur bornée de paramètres, cas 3D

La position (X_R, Y_R) et l'orientation θ_R du robot sont recherchées. Les paramètres ont des unités hétérogènes, mètre et radian. Les éléments du problème sont :

Equations :

Type 1 : $f(\mathbf{x}, \mathbf{x}_i) = arctan(\frac{Y_R - y_i}{X_R - \mathbf{x}_i}) - \theta_i$ avec $(\mathbf{x}_i, y_i)$ et θ_i avec $(\mathbf{x}_i, \mathbf{x}_i)$ et θ_i respectivement coordonnées et orientation du capteur C_i dans le repère lié à l'environnement.

Type 2 : $f(\mathbf{x}, \mathbf{x}_i) = arctan(\frac{Y_R - y_i}{X_R - \mathbf{x}_i}) - \theta_R$ $(\mathbf{x}_i, y_i)$ coordonnées du marqueur M_i dans le repère lié à l'environnement.

Paramètres :

$\mathbf{x} = (X_R, Y_R, \theta_R)^T$, **Domaine de recherche des paramètres : [x]**

Données capteurs :

$\mathbf{x}_i = (\mathbf{x}_i, y_i, \theta_i)$ respectivement coordonnées et orientation dans le repère lié à l'environnement.

Mesures :

$\lambda_1, \lambda_2, ...\lambda_m$ avec leur intervalle associé $[\lambda_1], [\lambda_2], ...[\lambda_m]$

Ensemble recherché :

$\mathcal{S} = \{\mathbf{x} \in [\mathbf{x}], f(\mathbf{x}, \mathbf{x}_1) \in [\lambda_1], f(\mathbf{x}, \mathbf{x}_2) \in [\lambda_2], ...f(\mathbf{x}, \mathbf{x}_m) \in [\lambda_m]\}$

En posant $F(\mathbf{p}) = (f(\mathbf{x}, \mathbf{x}_1), f(\mathbf{x}, \mathbf{x}_2), ..., f(\mathbf{x}, \mathbf{x}_m))^T$ et $[\mathbf{lambda}_1] \times [\lambda_2] \times ... \times [\lambda_m]$

Alors $\mathcal{S} = [\mathbf{p}] \cap F^{-1}(\mathbf{p})$

Fonction d'inclusion et test d'inclusion

La différence par rapport au cas 2D est que la fonction d'inclusion et le test d'inclusion dépendent de l'origine de la mesure : environnement ou robot. De plus, le paramètre ?R ne peut être déterminé que si l'algorithme dispose d'au moins une mesure prise à partir du robot.

II.6 Résultats de simulation

La simulation a été réalisée avec le logiciel Matlab. La simulation a pour objectifs de :

- Montrer que l'approche est opérationnelle quelles que soient la position des balises, la position relative robot-balises tant que les mesures ne violent pas l'hypothèse d'appartenance à son intervalle de mesure.
- Montrer les capacités d'intégration aisées de mesures supplémentaires, de capteurs

dont les caractéristiques métrologiques sont différentes mais qui respectent le modèle défini en début de chapitre et de mesures effectuées indifféremment par le robot ou l'environnement.
– Montrer l'influence du paramètre ϵ, de mesures supplémentaires sur le résultat de la localisation.

La dimension de la pièce est de $7m \times 7m$

II.6.1 Cas 2D

La figure II.9 donne la position (X_R, Y_R) du robot donné par l'algorithme de localisation qui utilise les mesures acquises par trois capteurs de l'environnement C_1, C_2, C_3. La largeur de l'intervalle de mesure des trois capteurs est de 1 degré. Le pavage rouge est constitué de pavés acceptables, le pavage bleu/jaune de pavés ambigus dont l'état n'a pas été déterminé, l'algorithme ayant atteint la précision de bissection epsilon demandée.

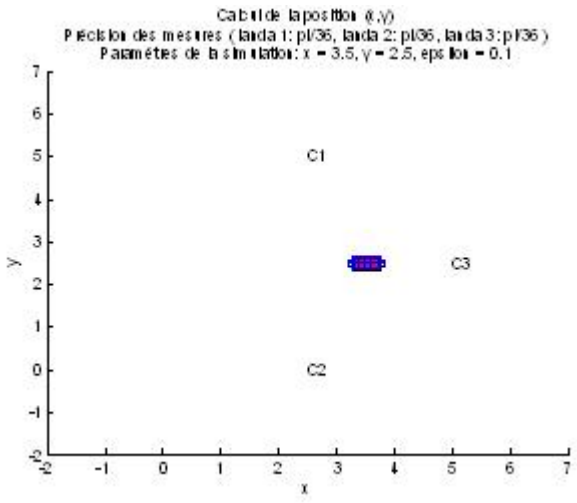

Figure II.9 : Localisation (X_R, Y_R) du robot (rouge, pavage acceptableă ; bleu/jaune, pavage ambigu

La figure II.10 suivante donne la précision de localisation du robot en fonction de sa position relative par rapport aux balises. Chaque pavage rouge cerné de pavés bleus/jaunes correspond à la réponse de l'algorithme en un point (x, y). L'incrément en x et en y est de 1 m. pour x et y compris entre -1 et 6 m.

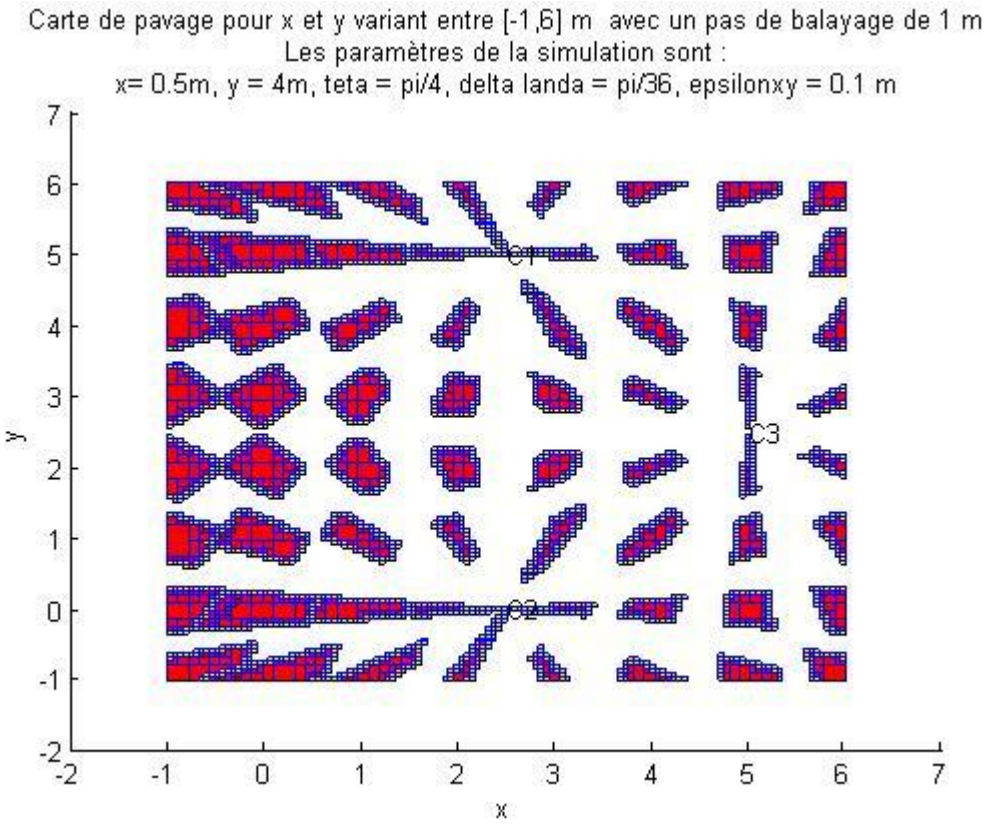

Figure II.10 : Localisation (Carte de précision de localisation (X_R, Y_R) du robot dans une pièce de 7mx7m(rouge, pavage acceptableă ; bleu/jaune, pavage ambigu)

On retrouve le fait connu avec les algorithmes de triangulation classiques que le résultat est plus précis à l'intérieur du rectangle formé par les capteurs. Pour être certain de la zone de localisation, il faut considérer l'ensemble des pavés rouges et bleus/jaunes.
La figure II.11 illustre l'influence du paramètre ϵ qui définit la taille minimale des pavés après bissection. Le choix d'une décomposition plus fine réduit le pavage ambigu mais au détriment du temps de calcul.

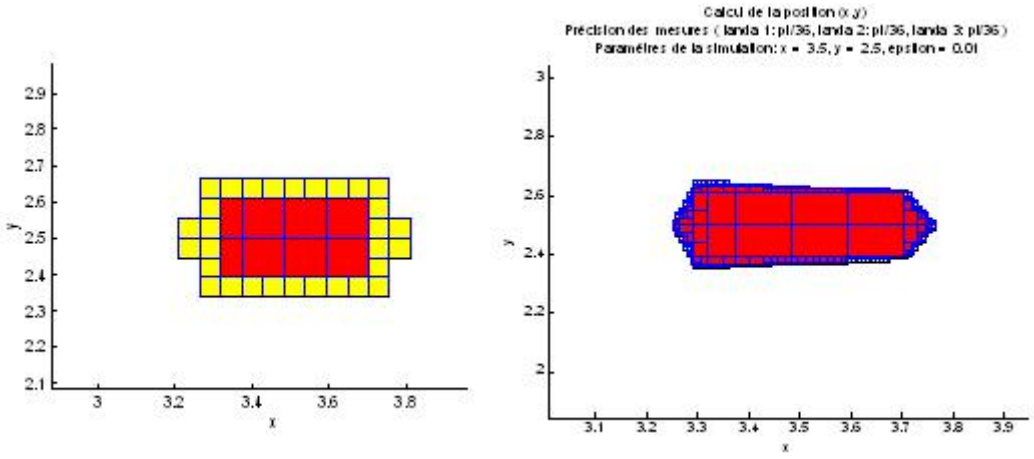

Figure II.11 : Localisation (X_R, Y_R) du robot (rouge, pavage acceptable ; bleu/jaune ou jaune, pavage ambigu). A gauche : Taille minimale des pavés, $\epsilon = 0.1$ m, à droite Taille minimale des pavés, $\epsilon = 0.01$ m

On retrouve le compromis classique précision/temps. En jouant sur le paramètre ϵ, l'algorithme est capable d'adapter sa réponse à la contrainte la plus forte d'une requête de type : demande de localisation (précision, temps). Cependant, une des difficultés de ce type d'approche est d'évaluer a priori le temps mis pour obtenir un résultat.

II.6.2 Cas 3D

La figure II.12 donne la position et l'orientation (X_R, Y_R, Θ_R) du robot. Les mesures sont acquises par le robot qui a détecté trois marqueurs M_1, M_2, M_3. La largeur de l'intervalle de mesure des trois capteurs est de 1 degré. La dimension des pavés est d'ordre trois. Les unités des axes sont hétérogènes, x_R et y_R en mètres et θ_R en radians.
La figure II.12 présente une projection sur les trois plans du résultat 3D de la localisation pour un θ_R de $\pi/4$. Par souci de lisibilité, seul le pavage acceptable est conservé.

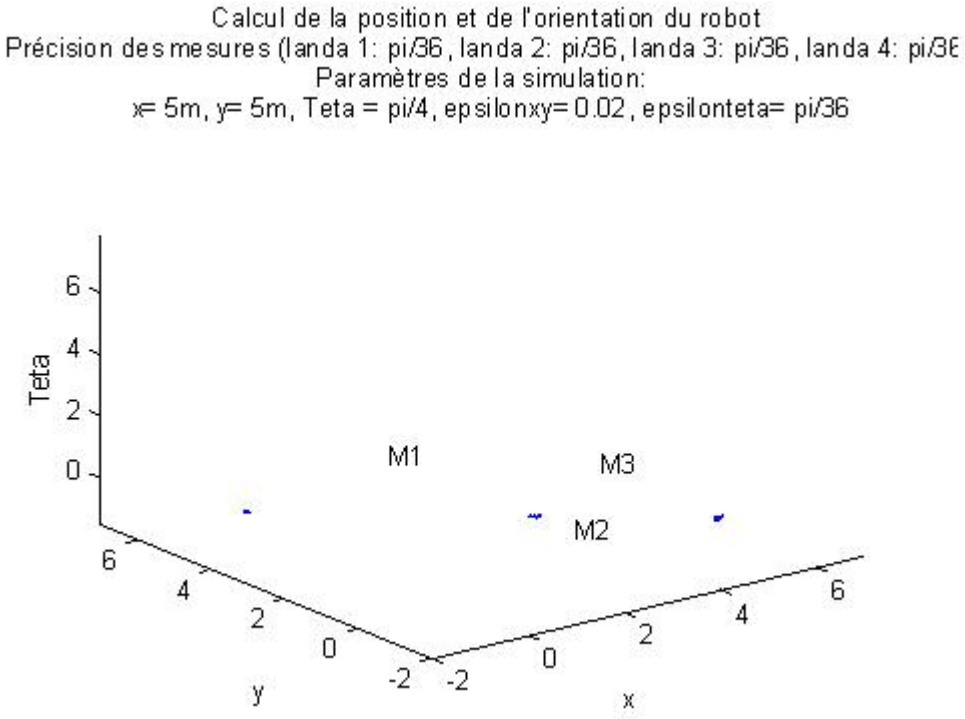

Figure II.12 : Projection sur les plans x-y, $x-\theta$, $y-\theta$ de la localisation. Seul le pavage acceptable est représenté.

La figure II.13 montre les projections sur les plan $x-y$ et $x-\theta$. Seul le pavage acceptable est représenté.

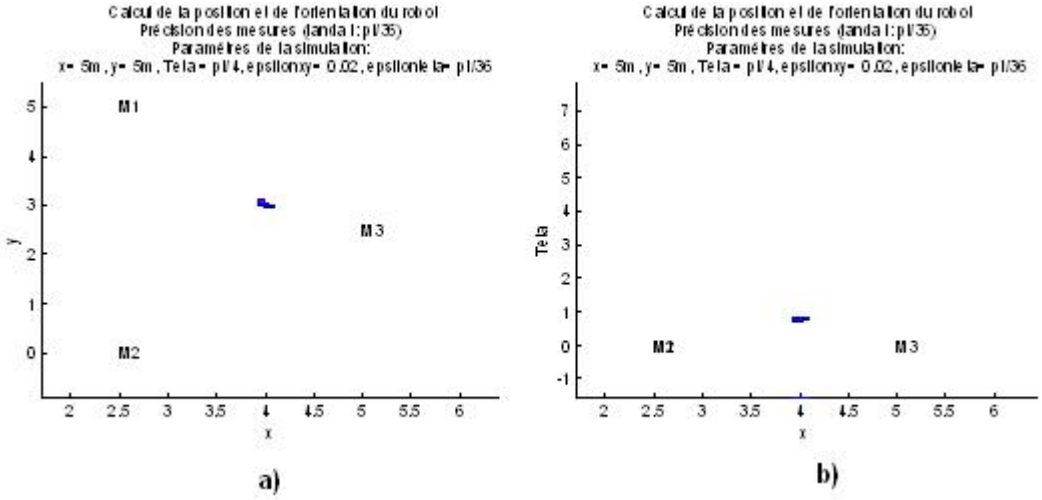

Figure II.13 : Localisation 3D, a) Projection sur le plan $x-y$, b) projections sur le plan $x-\theta$

Comme l'illustrent les deux figures précédentes, les résultats sont satisfaisants en terme

de précision. Cependant l'ajout d'une troisième inconnue, l'orientation θ_R du robot se traduit par un pavage ambigu plus important comme le montre la figure II.14 ci-dessous.

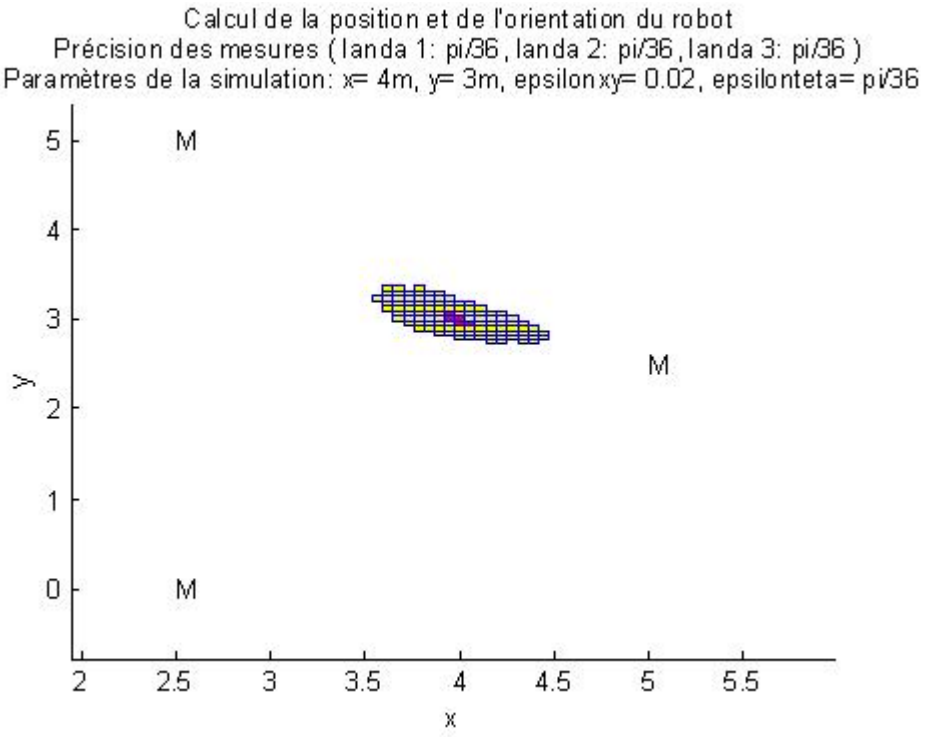

Figure II.14 : Zoom de la projection sur le plan x-y. Les pavés acceptables sont en rouge et les pavés ambigus en jaune encadré bleu.

La figure II.15 illustre l'influence du choix des paramètres epsilon sur la réduction de la taille du pavage ambigu.

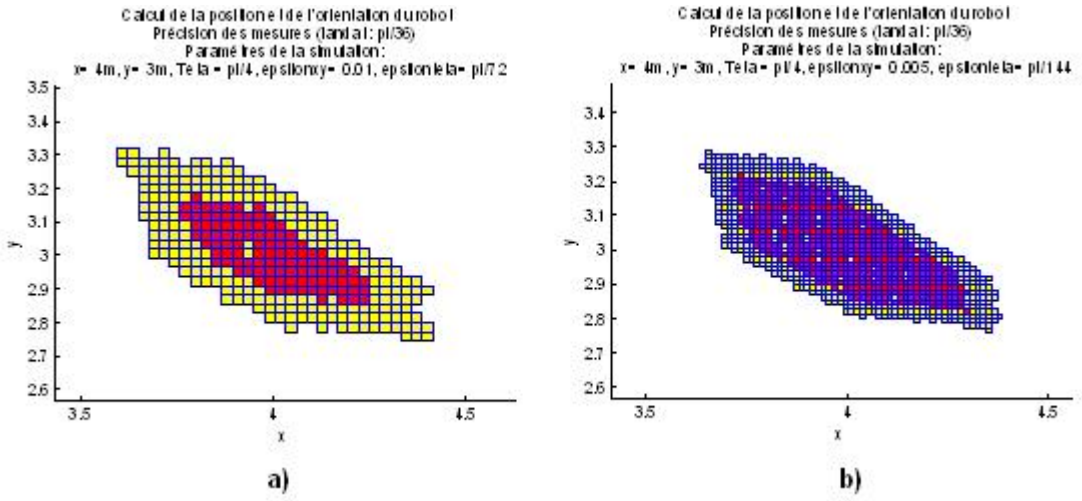

Figure II.15 : Influence du paramètre epsilon. Les pavés acceptables sont en rouge et les pavés ambigus en jaune encadré bleu (projection sur le plan x-y). a) epsilon = 0.01 m pour les inconnues x et y et $\epsilon = \pi/72$ pour l'inconnue θ b) epsilon = 0.005 m pour les inconnues x et y et $\epsilon = \pi/72$ pour l'inconnue θ

Diminuer la valeur des paramètres ϵ réduit le pavage ambigu. L'ajout d'une mesure supplémentaire sous réserve des caractéristiques de la mesure et de la position du capteur ou du marqueur peut améliorer la précision de localisation comme l'illustre la figure II.16.

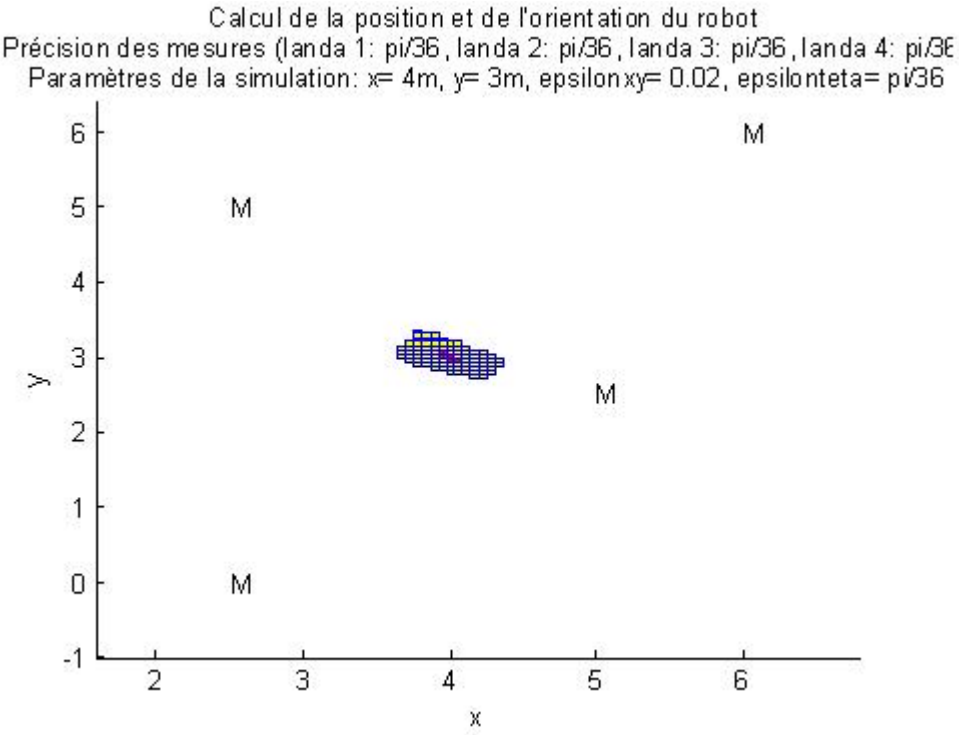

Figure II.16 : Localisation 3D utilisant un quatrième mesure (projection sur le plan x-y, teta simulé : $\pi/4$)

On peut évaluer de façon grossière l'amélioration apportée par cette nouvelle mesure en comparant le nombre de pavés solution de la figure II.16(nb = 107) et de la II.14 (nb =86) ce qui donne une amélioration de 20%.

II.6.3 Localisation avec des mesures issues du robot et de l'environnement

Cette approche permet d'intégrer dans le même calcul les mesures issues du robot et celles de l'environnement comme l'illustre la figure II.17.

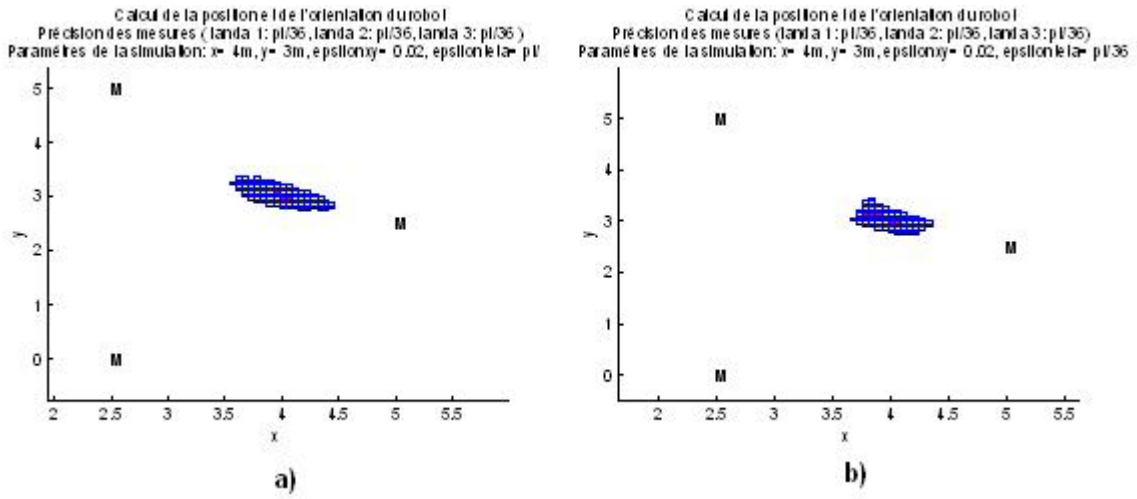

Figure II.17 : Zoom de la projection sur le plan x-y. Les pavés acceptables sont en rouge et les pavés ambigus en jaune encadré bleu.

Dans la figure II.17 a), les trois mesures sont acquises par le robot. Dans la figure II.17 b) une des mesures est acquise par un capteur de l'environnement.

II.6.4 Localisation avec mesures prises du robot situé à l'extérieur du triangle formé par les marqueurs

Quand le robot est situé à l'extérieur du triangle formé par les capteurs ou les marqueurs, le pavage ambigu peut devenir très important (figure II.18). Une nouvelle mesure réduit notablement cette zone d'incertitude. Dans cet exemple, le robot se trouve alors à l'intérieur de la nouvelle zone délimitée par les balises.

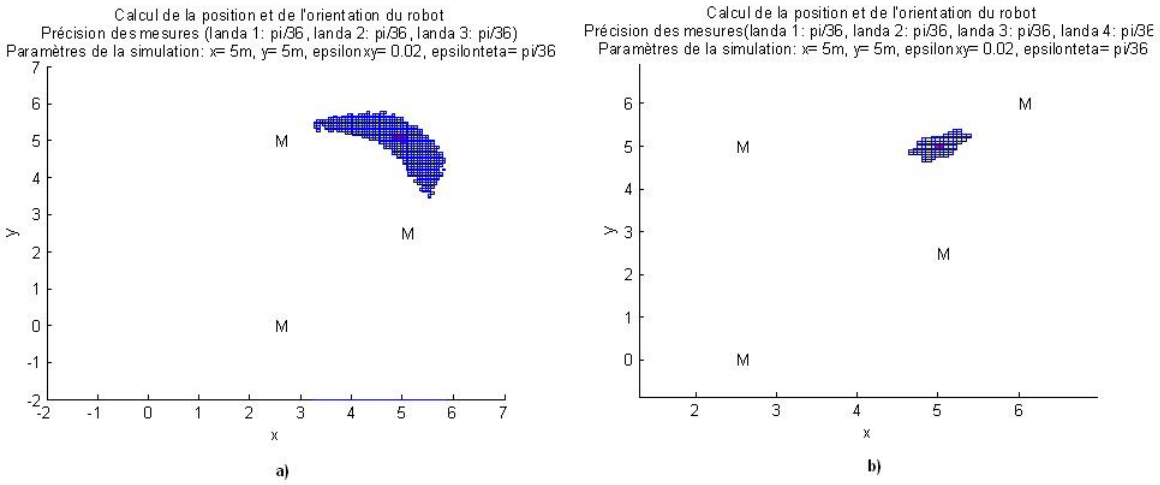

Figure II.18 : Localisation 3D (projection sur le plan x-y, θ simulé = 0) a) Localisation 3D utilisant trois mesures, b) Localisation 3D utilisant quatre mesures.

II.7 Conclusion

Nous proposons une méthode de localisation basée sur le principe de la multiangulation. La résolution du problème est obtenue par une approche ensembliste, plus précisément l'analyse par intervalles. Comme la dimension du vecteur d'état est inférieure ou égale à trois, les calculs sont relativement simples et a priori rapide (la simulation étant réalisée sous Matlab, une évaluation réaliste devra être effectuée pour estimer le temps moyen d'exécution).
Le modèle simple retenu pour les capteurs donne la possibilité d'intégrer une grande variété de capteurs, du plus fruste au plus complexe, si celui-ci délivre une mesure d'angle. Accessoirement, l'algorithme de calcul de la fonction intervalle $[arctana]$ est original et peut être utile à d'autres travaux. A ma connaissance, on ne trouve pas cette fonction dans les bibliothèques de calcul par intervalle.

La méthode est capable de s'adapter à plusieurs facteurs :
- un nombre variable de mesures ;
- des mesures hétérogènes au sens métrologique du terme (précision, portée) ;
- une connaissance statistique limitée sur ces mesures. Généralement l'erreur de mesure n'est connue que sous la forme d'une tolérance ;
- un placement des capteurs/marqueurs dans le repère de l'environnement et dans celui du robot ;
- une dispersion plus ou moins importante des capteurs et des marqueurs dans l'environnement. Un résultat de localisation, même partiel, peut être obtenu dès qu'une donnée mesurée est exploitable.

Les résultats de simulation montrent l'intérêt de cette approche. Le choix du paramètre ϵ permet régler le compromis entre la précision, par la réduction du pavage ambigu, et le temps de calcul.
La version de l'algorithme proposé dans cette thèse a pour vocation d'une part, de montrer l'intérêt de l'approche par intervalles, et d'autre part, de disposer d'une méthode de localisation qui sera utilisée par les agents capteurs du SMA au chapitre suivant. C'est un maillon important qui permettra d'évaluer les capacités d'adaptation permises par les mécanismes de coalition et de négociation implantés dans le SMA.
Des améliorations peuvent être apportées sur plusieurs points. Tout d'abord, les mesures aberrantes qui violent de l'hypothèse d'erreur bornée ne sont pas traitées. Actuellement cette violation se traduit par une absence de solution dont on est informé. Il existe des pistes dans la littérature pour y remédier. Par ailleurs, il a été considéré, pour simplifier la présentation de l'approche, que la localisation des capteurs et des marqueurs de l'environnement était connue. Mais il est facile d'introduire cette incertitude dans l'algorithme, bien entendu au détriment de la précision de localisation.

Chapitre III

Architecture informatique de Coopération Robot-Environnement

Objectifs du chapitre

Ce chapitre décrit l'architecture informatique pour la coopération robot-environnement basée sur un SMA. Un état de l'art introduit et positionne l'approche. Ensuite, ce chapitre décrit l'architecture générale du système suivi de la description des différents modules qui le constituent avec les mécanismes qui y sont mis en œuvre.

III.1 Introduction

Le deuxième objectif de cette thèse est de proposer une architecture informatique permettant la coopération entre le robot et les objets communicants présents dans l'environnement tels que les capteurs et les actionneurs. Une des difficultés est que le robot et l'environnement ambiant sont des systèmes dynamiques au sens où, par exemple, les capteurs ambiants peuvent être disponibles/indisponibles, accessibles/inaccessibles. En considérant en plus, le fait que les capteurs ambiants sont généralement inégalement répartis dans l'environnement, il devient difficile de concevoir une approche déterministe capable de répondre aux objectifs des scénarios pour lesquels le système est conçu. C'est pourquoi, nous nous sommes orientés vers une approche dotée de mécanismes d'adaptation permettant non seulement de prendre en compte l'aspect dynamique de l'environnement, mais aussi des facteurs comme le degré d'intrusion du système dans la vie privée de la personne et la précision des mesures des capteurs.

Dans le contexte du scénario décrit au chapitre I concernant la levée de doute après une alarme, le mécanisme d'adaptation doit prendre en compte deux contraintes. La première est liée à l'obligation de résultat. Le mécanisme d'adaptation doit apporter une réponse face à des situations diverses voir imprévues afin qu'au final le robot réussisse sa

mission, se déplacer d'un point A à un point B, si possible en tenant compte du degré d'urgence. L'autre contrainte est d'ordre éthique, l'intrusion d'une intelligence ambiante capable d'agir et de percevoir en présence de la personne occasionne une gêne pour celle-ci et son entourage. Il est indispensable de limiter cette gêne autant que possible. C'est la raison pour laquelle nous avons introduit la notion d'intrusion du système, dont le degré est dynamiquement modulé en fonction du profil du patient et de la situation, notamment le degré d'urgence nécessaire pour l'intervention du robot.

Dans le cadre de cette thèse nous évaluerons l'architecture informatique de coopération entre le robot et le réseau de capteurs, ainsi que les mécanismes d'adaptation avec la tâche de localisation du robot ; l'algorithme utilisé étant celui décrit au chapitre II. L'idée d'obligation de résultat, en ce qui concerne la tâche de localisation, respecte le point de vue adopté au chapitre localisation. C'est moins la précision (dimension de la zone) que la certitude que le robot appartient à cette zone qui importe. Nous partons du principe qu'avec une bonne connaissance de sa situation, la certitude d'être dans cette zone, permettra au robot de choisir, avec une meilleure pertinence, une stratégie pour assurer sa mission. Pour répondre à cette obligation de résultat, nous avons introduit la notion d'effet souhaité. Le système cherche à se rapprocher le plus près possible de l'effet fixé comme étant l'objectif à atteindre. Il peut s'agir d'un besoin du robot de se localiser avant de se diriger vers la personne, d'éclairer une zone afin d'améliorer la qualité de l'image délivrée par une caméra de surveillance ou de proposer un exercice de stimulation cognitive afin d'entraîner les capacités cognitives de la personne. Le système s'ajustera en fonction de certains critères (précision, niveau d'intrusion, informations disponibles) sans forcément rechercher la solution optimale. Si on considère l'exemple de la localisation, l'effet souhaité peut être exprimé de la manière suivante : se localiser avec une précision donnée, en prenant en compte les informations disponibles les plus récentes.

D'une part, le principe est que l'environnement est composé d'objets communicants (robot compris) et que ces capteurs ou actionneurs proposent des services (détecter une présence, éclairer, ...). D'autre part, la combinaison de certains de ces services permet d'aboutir à une autonomie fiable du robot par un modèle de coopération Robot-Environnement. Afin de contrôler l'état et le comportement de l'environnement (capteurs + robot) et d'agir sur ce dernier, mais aussi de pouvoir réagir dynamiquement aux changements de cet environnement, les systèmes multi-agents (SMA) fournissent une solution dont l'adéquation n'est plus à démontrer. En effet, pour atteindre des objectifs dans un environnement composé d'objets en interaction et qui évoluent, on cherche à répondre à des problématiques comme la réactivité et l'adaptativité, qui est l'une des caractéristiques les plus importantes des SMA.

Pour notre cadre d'étude, en supposant que chaque objet de l'environnement soit représenté par un agent et que les SMA proposent un paradigme efficace et souple pour le développement de systèmes à plusieurs composantes autonomes (agents) pouvant co-

opérer. Il est possible d'affirmer qu'ils permettent de modéliser notre système qui est hétérogène, complexe, non linéaire et évolutif, et de faire apparaître une intelligence et des capacités globalement supérieures à celles des agents qui les composent. C'est ce qui sera illustré tout au long de ce travail.

Ainsi, ce chapitre commence par une étude bibliographique qui positionne notre approche basée sur les systèmes multi-agents et le principe de formation de coalitions entre agents. Cela permettra de mettre en évidence les mécanismes d'adaptation sous-jacents à de telles approches. Nous décrirons ensuite le schéma général de notre architecture de Coopération Robot-Environnement ambiant et deux scénarios utilisés pour illustrer les principes de fonctionnement de notre système. Puis, nous présentons une ontologie dédiée à l'assistance ambiante conçue pour modéliser l'ensemble des connaissances utiles à la représentation de l'environnement, et au fonctionnement du SMA. Ce dernier est décrit en détails en illustrant les mécanismes de formation de coalitions pour l'obtention d'effets souhaités, sur les scénarios considérés. Le protocole d'évaluation de l'approche avec les résultats est décrit au chapitre IV.

III.2 Les systèmes multi-agents et la formation de coalitions

Cette première partie a pour but d'éclairer les techniques multi-agents que nous avons mis en œuvre dans notre système. Elle ne prétend pas être un état de l'art exhaustif sur ce qui se fait dans le domaine des SMA. Après la présentation de quelques notions générales telles que la définition d'un agent, d'un SMA et de la notion d'interaction, nous définirons les mécanismes de coopération employés dans le cadre de l'Assistance Ambiante. Enfin, nous détaillerons les mécanismes de coopération basés sur la formation de coalitions que nous avons implantés.

III.2.1 Généralités

Les (SMA) offrent un paradigme de premier choix pour le développement et le déploiement de systèmes à plusieurs composantes autonomes (agents) pouvant coopérer. Ils permettent de modéliser des systèmes hétérogènes, complexes et évolutifs, et de faire apparaître une intelligence et des capacités qui sont différentes et globalement supérieures à celles des agents qui les composent. Cette intelligence est le fruit de la coexistence d'agents autonomes aux multiples interactions dans un environnement dynamique.

Les SMA ont été utilisés avec succès dans de nombreux domaines, dont : le développement de systèmes informatiques décentralisés (ou ingénierie logicielle orientée multi-agents) où l'approche SMA permet l'intégration flexible et la coopération de logiciels et de services autonomes, la résolution collective de problème pour laquelle il s'agit de résoudre

de manière distribuée un problème qui se pose globalement à la collectivité d'agents, la simulation de phénomènes complexes où la modélisation multi-agent apporte un cadre conceptuel permettant la représentation et la simulation de systèmes faisant intervenir différentes entités en interaction [Zahia2009].

L'utilisation des SMA pour résoudre une problématique complexe offre de nombreux avantages. En effet, de part son caractère réparti, ouvert et hétérogène, la gestion et la construction du modèle agent matérialisant le système devient moins complexe. Les SMA permettent en outre la distribution des ressources et de l'expertise, ce qui amène logiquement à une facilité d'interopérabilité avec des systèmes déjà existants.

Il existe trois grands courants dans la recherche en SMA. Les SMA considérés comme des systèmes répartis, les SMA du point de vue intelligence artificielle distribuée et enfin en ingénierie logicielle, les SMA comme paradigme de programmation dite "agent".

Dans le domaine des systèmes répartis, un SMA est défini comme un système composé de plusieurs systèmes calculatoires ou entités autonomes (sinon, non réparti) sans mémoire physique commune (sinon c'est un système parallèle, cas dégénéré) qui communiquent par l'intermédiaire d'un support quelconque. Le système permet le partage et la gestion de l'hétérogénéité des ressources ainsi que des données ou des traitements. Samuel Tardieu [Tardieu2004] considère qu'un système réparti est composé de plusieurs systèmes calculatoires autonomes, sans mémoire physique commune et qui communiquent par l'intermédiaire d'un réseau.

L'intelligence artificielle distribuée (IAD) est définie comme étant une branche de l'intelligence artificielle. Elle s'intéresse à la modélisation de comportements "intelligents" par la coopération d'un ensemble d'entités dotées de caractéristiques de haut niveau. Ces entités sont les agents et les systèmes correspondants sont appelés systèmes multi-agents. L'IAD propose la distribution de l'expertise sur un groupe d'agents. plusieurs systèmes interagissent pour résoudre un problème commun [Moulin1996].

Enfin, avec l'évolution des techniques et technologies de la programmation, un nouveau thème de génie logiciel basé sur les SMA, s'est beaucoup développé ces dernières années, c'est la programmation orientée agent. Introduit par Yoav Shoham en 1992 [Shoham1992]. Son objectif principal est de développer des modèles, des méthodes et des outils qui facilitent le développement et la maintenance d'applications multi-agents fiables. L'intérêt étant de pouvoir décomposer un problème en de multiples composants autonomes qui sont des agents ayant un objectif [Shoham1992].

C'est l'IAD qui a produit les concepts tels que la coopération, l'organisation, l'interaction avec les notions de modèles et de protocoles associés. C'est dans ce courant que se situe l'approche multi-agents que nous avons développée.

III.2.2 Définitions

S'il est aisé d'établir un consensus sur ce qu'est un SMA, il n'en est pas de même pour le terme agent lui-même. En effet, il existe dans la littérature de nombreuses définitions qui, sans se contredire, sont plus ou moins pertinentes selon l'aspect qui est mis en avant. La définition adoptée dans ce travail, est celle proposée par Wooldrige et Jennings en 1995 et révisée en 2009 [Wooldridge2009]. Elle décrit un agent comme un système informatique, situé dans un environnement, et qui agit d'une façon autonome pour atteindre un objectif commun (but) pour lequel il a été conçu.

Un SMA est défini comme un système distribué, composé d'un ensemble d'agents possédant, chacun des capacités de résolution individuelles limitées et des capacités d'interaction avec les autres agents et l'environnement. Il se caractérise par un contrôle distribué, c'est-à-dire, qu'il n'y a pas d'ordonnanceur global de l'activité des agents, et par des données décentralisées sur lesquelles les calculs se font de façon asynchrone. Nous accordons à notre SMA, les facultés d'auto-organisation et d'auto-adaptation. Un système organisé peut être défini comme un système possédant une structure et un ensemble de fonctionnalités associées. La structure sert à organiser les entités et à faciliter leurs interactions. Quant aux fonctionnalités, elles ont comme objectif de maintenir cette structure et faciliter son utilisation. L'auto-organisation fait alors référence à un système organisé sans entité centralisée et sans contrôle extérieur. Cette propriété permettra de faire émerger un comportement global (structurant) à partir d'interactions locales. Quant à la faculté d'auto-adaptation, elle s'avère nécessaire pour une exploitation fiable et efficace des informations de l'environnement ambiant. Cette faculté d'auto-adaptation demeure une tâche sensible à cause de l'évolution dynamique et continuelle du système.

Selon le type de connaissances et le mode employé pour le mécanisme de décision, les agents sont classés en trois catégories : les agents réactifs, les agents cognitifs et les agents hybrides. Leur utilisation varie selon le contexte et le degré de difficulté du problème à résoudre.

III.2.2.1 Agents réactifs

C'est Brooks [Brooks1986], [Brooks1991a], [Brooks1991b] qui a introduit les agents réactifs qu'il définit comme étant des agents ayant comme caractéristique principale un comportement très simple qui consiste à percevoir l'environnement (les entrées) et à agir sur ce dernier (les sorties) (figure III.1). De tels agents possèdent un état interne, mais pas de représentation symbolique de l'environnement, ni de connaissances sur les autres agents. Le comportement intelligent d'un SMA dit "réactif" émerge uniquement des interactions entre les agents et avec l'environnement.

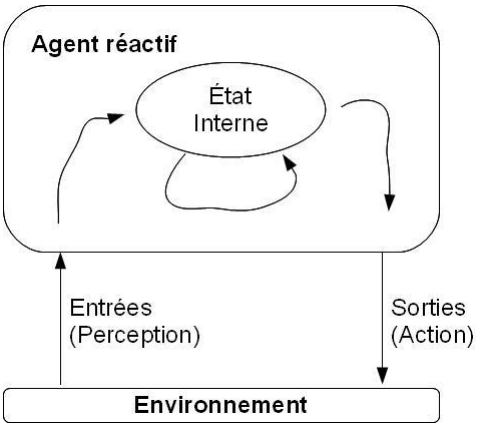

Figure III.1 : Architecture réactive

III.2.2.2 Agents cognitifs

Contrairement aux agents réactifs, les agents qui sont dits cognitifs possèdent une représentation symbolique et partielle de leur environnement et des autres agents. Plus généralement, ces agents possèdent des connaissances, des croyances, des désirs et des buts. Une architecture connue sous l'appellation d'architecture BDI (Believe, Desire, Intention) qui a été introduit par Rao et Georgeff [Rao1991] est de cette nature. Un agent cognitif raisonne sur ses croyances qui représentent sa compréhension du monde dans lequel il évolue. Il raisonne aussi sur ses désirs et ses intentions en relation avec ses croyances et capacités, afin de prendre des décisions auxquelles sont associés les plans qu'il va accomplir pour agir sur son environnement.

III.2.2.3 Agents hybrides

Les agents hybrides sont une résultante entre les agents cognitifs purs qui suivent imperturbablement le plan qu'ils ont établi, et les agents strictement réactifs qui ne prennent leurs décisions qu'en fonction des données provenant de leurs senseurs. Cette nouvelle catégorie d'agent a été introduite par [Georgeff1987] et appuyée par des travaux tels que [Chaib-Draa1994]. Les agents hybrides possèdent une représentation partielle de leur environnement, ils savent élaborer et suivre un plan pour arriver à satisfaire leurs désirs. De plus, ils disposent d'une souplesse comportementale leur permettant d'abandonner temporairement l'exécution de leur plan pour saisir les opportunités qui s'offrent à eux. Les systèmes multiagents que nous étudions dans nos travaux sont à la fois réactifs et cognitifs, c'est-à-dire, hybrides.

III.2.3 SMA et intelligence ambiante

Quelques travaux ont été proposés dans le domaine des SMA pour la modélisation de problèmes relevant du domaine de l'Intelligence Ambiante. A part les projets iDorm et RoboCare qui ont été déjà définis au chapitre I, le projet MavHome, [Cook2003] utilise plusieurs disciplines de l'intelligence artificielle comme la fouille de données et les systèmes multi-agents. C'est un projet développé à l'université du Texas basé sur la construction d'une maison intelligente avec diverses fonctions. Les capacités de la maison sont : le contrôle adaptatif de la température, de l'eau, de la lumière, du nettoyage et du loisir multimédia (audio et vidéo).

iDorm, RoboCare et MavHome sont des projets destinés principalement à assister des personne à domicile. Les contraintes et les caractéristiques des maisons intelligentes sont telles que les systèmes multi-agents ont été choisis comme solution, généralement pour gérer la complexité des systèmes mais aussi pour assurer des services adéquats pour l'utilisateur final. Les SMA utilisés dans ces systèmes ne possèdent pas de faculté d'adaptation. La capacité d'adaptation est une contrainte à laquelle nous cherchons une solution. Nous explorons le SMA par leur faculté d'auto-adaptation. Cette capacité est mise en oeuvre non seulement au niveau des interactions et de la coopération entre les agents, mais également vis-à-vis des aspects dynamique et difficilement prédictible de l'environnement.

III.2.4 Protocoles d'interaction et coopération

Dans ce qui suit, nous présentons une définition de ce qu'est un protocole d'interaction qui répond à notre objectif d'exploitation des interactions entre plusieurs agents pour la construction de scénarios de coopération. Cette définition est suffisamment générique pour pouvoir permettre de doter le SMA de capacité d'adaptation :

"Interaction protocols are descriptions of standard patterns of interaction between two or more agents. They constrain the possible sequences of messages that can be sent amongst a set of agents to form a conversation of a particular type [Greaves2000]."

Ainsi, un protocole d'interaction est un modèle d'interactions représenté d'une manière générique, indépendamment du contexte d'application. Il définit des règles qui permettent de régir une interaction. Ces règles définissent l'ordonnancement des messages, ainsi que les actions auxquelles le protocole fait appel.

Ces protocoles sont abordés dans la littérature de diverses façons. Sandholm [Sandholm1999] identifie cinq grands types de protocoles d'interaction : les systèmes de vote, les enchères, le marchandage, les systèmes contractuels et la formation de coalitions. À un niveau plus fonctionnel, Mazouzi [Mazouzi2002] classe les protocoles d'interaction en quatre principales classes qui sont les protocoles de coordination, les protocoles de coopération, les protocoles de négociation et les protocoles de vente aux enchères. Verrons [Verrons2004]

regroupe quant à lui, les différentes formes de négociation en plusieurs familles. La première famille est celle des systèmes de vote qui permettent de choisir une solution parmi plusieurs. La deuxième famille concerne les enchères dont les principaux types sont l'enchère anglaise, l'enchère hollandaise et l'enchère Vikrey. La troisième famille regroupe les négociations se basant sur le protocole Contract Net [Smith1980]. Enfin, citons les protocoles à base d'argumentation tels que celui dans [Kraus1998] où chaque agent utilise des arguments pour convaincre l'autre d'accepter la proposition faite.

L'interaction est le support de la matérialisation de la coopération. Tandis que la coopération est un concept observable au niveau global dans un SMA, l'interaction s'observe entre les agents deux à deux. La manière avec laquelle un ensemble d'agents forment une entité coopérative (un groupe, une coalition, une organisation ou une équipe) est ce qui distingue les différents travaux de recherche sur la coopération. Nous en décrirons quelques uns ci-dessous, avant de présenter et d'illustrer notre approche.

III.2.4.1 États mentaux de groupes et réseaux de dépendance sociale

La notion d'état mental de groupe englobe une grande variété de concepts, comme les connaissances ou intentions communes ou encore les croyances mutuelles. L'exemple le plus célèbre d'une architecture basée sur les états mentaux de groupe est celle proposée dans [Gray1978]. Il traite le problème de deux généraux qui essaient de répondre à la question de savoir comment, et dans quelle mesure, il est possible de prendre en compte une information dont la source pourrait être suspecte. Les auteurs démontrent qu'un état de connaissance commun au sens strict du terme ne peut être atteint que simultanément par tous les agents concernés.

Les notions de puissance et de dépendance sociales ont également été introduites dans le domaine des SMA [Castelfranchi1990]. Un réseau de dépendance sociale représente les relations de dépendance au sein d'un groupe d'agents. Un agent A dépend d'un autre agent B s'il a besoin de B pour accomplir ses buts, et inversement on dira que B a du pouvoir sur A. La connaissance de ces relations par un agent délibératif fait partie des modèles dont il dispose et doit lui permettre de raisonner à propos des autres agents afin de se rapprocher de ses propres buts en fonction de ce qu'il connaît des autres.

III.2.4.2 Travail d'équipe

Le travail d'équipe est utilisé dans les approches top-down. Il consiste à rassembler des agents dès le départ dans un but précis et de manière à ce que leurs capacités se complètent pour l'accomplissement de la tâche de l'équipe. Chaque agent possède une compétence de base et un agent particulier, spécifie pour chacun, la tâche à accomplir

selon le plan de l'équipe qui est déjà établi. Dignum et ces co-auteurs proposent d'utiliser des conversations entre agents plutôt que des protocoles rigides (comme Contract Net [Smith1980]) afin de former les équipes [Dignum1999] [Dignum2001]. Après une phase qui permet de déterminer les membres et les équipes potentiels pour accomplir une tâche, l'agent à l'initiative du processus de création de l'équipe dispose d'un ensemble d'équipes potentielles (classées par ordre de préférence). Vient alors une phase de négociation entre l'"initiateur" de l'équipe et les membres potentiels afin de les amener à adopter les états mentaux nécessaires pour faire partie de l'équipe.

III.2.4.3 Formation dynamique de groupes

Récemment, un certain nombre de travaux ont porté sur la formation dynamique de groupes, c'est-à-dire, donner à des agents la possibilité de former rapidement des coalitions pour répondre aux évolutions de l'environnement, mais également de pouvoir faire évoluer ces coalitions en fonction des conditions. En effet, la plupart des SMA doivent faire face à des environnements dynamiques dans lesquels les agents n'ont ni le temps ni les informations nécessaires pour aboutir à une formation de coalitions optimale.

Un exemple d'application de la nécessité de former dynamiquement des groupes est l'application au suivi de cible multicapteur dans [Soh2002]. L'agent qui est à l'origine du processus de formation de coalitions sélectionne à la hâte un ensemble d'agents qui sont des voisins, puis affine le processus en conduisant en parallèle des négociations avec chaque membre potentiel dans un processus d'argumentation au cours duquel contraintes et engagements sont échangés. L'objectif consiste à classer les candidats sur leurs valeurs d'utilité potentielle pour la coalition de telle sorte que l'agent initiateur peut négocier avec les agents avec la plus grande utilité. Pour finir, l'initiateur scelle le statut de la coalition par un engagement. Dans [Soh2002], les auteurs proposent un algorithme de formation dynamique de coalition à base de négociation dont l'objectif est d'assurer (1) une bonne probabilité de formation d'une coalition ; (2) la gestion d'informations partielles ; (3) la robustesse du processus de formation ; et (4) la flexibilité des coalitions formées, qui doivent pouvoir adapter dynamiquement leur composition. Cet algorithme a été utilisé dans le domaine du suivi de cible multicapteurs pour lequel un ensemble de capteurs (les agents) doivent coopérer pour suivre un objet mouvant. Ce système a été conçu pour faire face à deux types d'événements : l'arrivée d'une nouvelle cible (trois agents sont nécessaires pour suivre une cible de manière précise) et un manque de ressources de calcul. Ils y introduisent la notion de voisinage des agents qui est utilisée comme critère pour la sélection des agents qui doivent participer en priorité à une coalition en réponse à un événement.

III.2.5 Formation de coalitions

Nous avons orienté notre choix vers la famille des protocoles basés sur la formation rapide de "groupe" capable de modifier leur organisation en cas de nécessité. De plus, la complexité et la particularité de notre domaine d'application nous ont amené à réfléchir à des mécanismes de formations de groupe qui prennent en compte des critères internes aux agents (capacité) et des critères liés au contexte (niveau d'intrusion, précision) tout en gardant une certaine simplicité de conception. C'est pourquoi, afin de limiter notre étude, nous nous intéresserons directement et essentiellement aux protocoles d'interaction à base de formation de coalitions. Dans le cas d'agents physiques comme des robots ou des satellites, l'environnement a une influence cruciale sur la coordination en raison de sa dynamicité et des contraintes physiques qu'il fait peser sur les agents. Les mécanismes de formation de coalitions nous intéressent particulièrement en ce point du fait que les agents s'associent pour réaliser une tâche collective ou se partager des tâches ; ils s'organisent en structures au cycle de vie court, ce qui permet une adaptation à notre environnement dynamique ; et enfin les agents tentent de trouver des solutions en présence d'informations incertaines et (ou) incomplètes. Si nous y ajoutons les contraintes de communication et la nécessité de proposer une solution pour chaque situation, notre approche peut être considérée comme une problématique de formation de coalitions décentralisée avec des agents hybrides.

Qu'est ce qu'un protocole de formation de coalitions ?

Une coalition est par définition, une alliance momentanée de plusieurs agents dans le but de réaliser un objectif commun. Les mécanismes de formation de coalitions cherchent à rassembler des agents en fonction de leurs intérêts. Les approches basées sur la formation de coalitions se distinguent des approches basées sur la formation dynamique de groupes du fait qu'elles sont fondées sur la notion d'utilité. En effet, elles suivent le principe que ces agents coopèrent les uns avec les autres en se fixant comme objectif de maximiser leur propre utilité individuelle.

Le problème est caractérisé d'un ensemble d'agents dont tout sous-ensemble est appelé coalition, et d'une fonction qui associe à toute coalition de l'ensemble d'agents sa valeur, c'est à dire le gain total espéré pour cette coalition. Les agents qui forment une coalition sont noués par un contrat de redistribution de la valeur de cette coalition. On suppose en outre que la valeur d'une coalition ne dépend pas des agents qui n'en font pas partie et qu'il ne peut y avoir de redistribution à l'extérieur de la coalition. Et la dernière hypothèse est que ce contrat est supposé connu par tous les agents.

L'objectif d'un mécanisme de formation de coalitions est de former des coalitions stables. La question de la stabilité peut être formulée du point de vue individuel ou collectif et peut être évaluée selon des critères directement issus de la théorie des jeux. La valeur de Shapley [Shapley1953] est un bon exemple. Le critère individuel principal

de stabilité est la rationalité individuelle : chaque agent d'une coalition doit obtenir une utilité au moins égale à celle qu'il recevrait en étant seul. Une telle situation correspond à ce qu'on appelle un équilibre de Nash [Nash1951] c'est à dire une situation dans laquelle aucun agent n'a pas intérêt à dévier unilatéralement de sa position (ici, son appartenance ou non à une coalition).

D'un point de vue collectif, le concept d'optimum de Pareto [Pareto1896] est plus adapté : on considère qu'une configuration A (de manière générale une distribution de coalitions associées à leurs fonctions de répartition) "Pareto-domine" une autre configuration B si et seulement si tous les agents ont une utilité dans la configuration A au moins égale à celle qu'ils ont dans la configuration B et si au moins un agent possède dans A une utilité strictement supérieure à celle qu'il obtient dans B. On parle d'optimum de Pareto pour une configuration qui n'est Pareto-dominée par aucune autre, c'est vers cet objectif que tend un mécanisme de formation de coalitions "collectivement rationnel".

Généralement la formation de coalitions se fait en trois grandes étapes [Sandholm1999]. La première étape étant la constitution de toutes les coalitions pouvant répondre au problème posé. Un agent initie la négociation, il devient l'initiateur en informant les autres agents et en recevant leur accord. Chaque agent devant attendre la fin d'une négociation avant d'en commencer une nouvelle. Tous les agents se transmettent alors leurs tâches. Ils peuvent ainsi former les coalitions représentant chacune des tâches ainsi que leurs paramètres associés. Vient ensuite, la phase de négociation où la nomination de la structure de coalition la plus optimale pour résoudre le problème a lieu. Enfin, la dernière étape est le déploiement de la coalition retenue.

Plusieurs stratégies ont été explorées dans la formation de coalitions. On peut citer les approches incrémentales, les approches par références, les approches aléatoires et les approches adaptatives.

III.2.5.1 Approches incrémentales

Les approches dites incrémentales [Abdallah2004], [Ortiz2002], [Shehory1998], [Shehory1999], [Sims2003] visent à former des coalitions au fur et à mesure qu'on se rapproche ou qu'on trouve la solution finale. Le principe est de générer un comportement global satisfaisant l'ensemble du système de manière incrémentale en partant d'un comportement très simple et en les combinant pour en avoir de plus élaborées. La cardinalité des coalitions augmentent au fur et à mesure. Dans [Abdallah2004], Abdallah et Vesser définissent une structure de coalition hiérarchique dans laquelle, chaque "Manager" de chaque coalition attribue de façon incrémentale, les tâches à leurs membres. Marc Sims et co-auteurs dans [Sims2003], présentent une approche distribuée de l'auto-organisation dans un réseau de capteurs distribués. Les agents du système déploient une série de négociations pour former progressivement des coalitions appropriées en fonction des ressources des capteurs.

Les auteurs ont développé un ensemble de protocoles que les agents peuvent utiliser en fonction de leurs besoins.

III.2.5.2 Approches par références

Les approches dites par références [Kraus2003] qui, au contraire des approches incrémentales, implémentent des coalitions de taille maximale suivant la compétence des agents. L'optimisation des coalitions dans un second temps, par le retrait des agents inutiles. Cette approche suppose un système centralisé dans lequel évolue un agent ordonnanceur pour publier les appels d'offres disponibles, recueillir les propositions de coalitions qui répondent à ces appels d'offres, afin de déterminer la coalition gagnante, et enfin de réduire le coût de la coalition dans le temps. A l'apparition de la coalition optimale, l'ordonnanceur est également responsable de la distribution des gains aux membres de la coalition.

III.2.5.3 Approches aléatoires

Les approches aléatoires [Scully2004], [Sen2000] visent à former de façon aléatoire les coalitions d'agents pour ensuite fusionner celles qui sont inacceptables. Dans [Scully2004], Scully et ses co-auteurs expliquent leur approche de formation de coalitions basée sur l'optimum de Pareto selon un critère précis. Les coalitions sont formées initialement de manière totalement aléatoire puis au fur et à mesure que le système évolue, les coalitions sont fusionnées progressivement jusqu'à la stabilité de l'ensemble.

III.2.5.4 Approches adaptatives

Enfin, de nombreux travaux s'intéressent aux approches adaptatives [Soh2001], [Soh2002], [Soh2003] où chaque agent possède un historique des coalitions qui ont été acceptées pour entamer un processus de génération de coalitions avec son voisinage. Dans [Soh2001], Soh et Tsatsoulis décrivent leur modèle de formation de coalitions. Au cours de l'initialisation de la coalition, l'agent initiateur crée une liste classée des agents qu'il estime utiles. Ensuite, l'agent initiateur approche les partenaires potentiels et demande des négociations lors d'une étape de finalisation.

III.2.5.5 Discussion

La formation de coalitions nous nous semble bien adaptée à notre problématique en raison du fait que les agents s'organisent en structures avec un cycle de vie court, ce qui permet une adaptation à un environnement dynamique et qu'enfin les agents tentent de trouver des solutions en présence d'informations soit incertaine, soit incomplète. La formation de coalitions peut être une solution car son organisation adaptative en fonction

du contexte permet de trouver une solution émergente à partir des ressources disponibles de l'environnement qui peuvent varier à tout moment.

Le protocole de formation de coalitions est une solution qui est un compromis entre une approche techniquement réalisable pour la coopération robot-environnement et la modélisation d'un ensemble de règles génériques modélisant cette coopération. Ainsi, chaque capteur et/ou actionneur de l'environnement ainsi que le robot sont encapsulés par des agents capables de coaliser pour répondre à une demande d'un effet souhaité.

Ce qui distingue notre approche de formation de coalitions des autres protocoles de formation de coalitions cités ci-dessus est la prise en compte de critères externes aux agents en plus de leurs critères internes. En effet, dans les approches classiques de formation de coalitions, c'est la notion d'utilité individuelle qui est exclusivement employée puis redistribuée dans la coalition tandis que dans notre protocole, les coalitions se réajustent en fonction du contexte global. De plus, tout agent peut être initiateur de coalitions. Enfin, l'agent peut reconstruire sa coalition lorsque son déploiement ne satisfait pas à l'effet souhaité et si le temps le permet. Ce modèle permet à un agent de former une coalition initiale rapidement pour réagir à une demande d'effet souhaité. Cette coalition évolue au cours du temps tant que le système n'a pas atteint l'effet souhaité.

La suite de ce chapitre présente en détails, la conception de l'approche et les résultats obtenus sont reportés au chapitre suivant. Nous allons décrire le système dans lequel les agents évoluent.

III.3 Architecture générale

Le système SMA constitue la partie décisionnelle de l'architecture. Il perçoit et agit sur l'environnement ambiant au travers de la passerelle. Cette dernière a pour rôle principal de gérer l'hétérogénéité des réseaux et des objets communicants. Pour prendre ses décisions, le SMA s'appuie sur une base de connaissances, implantée sous la forme d'une ontologie. L'utilisateur a un double rôle ; il peut émettre une demande d'un effet souhaité ou bien répondre à une sollicitation du SMA pour valider un résultat.

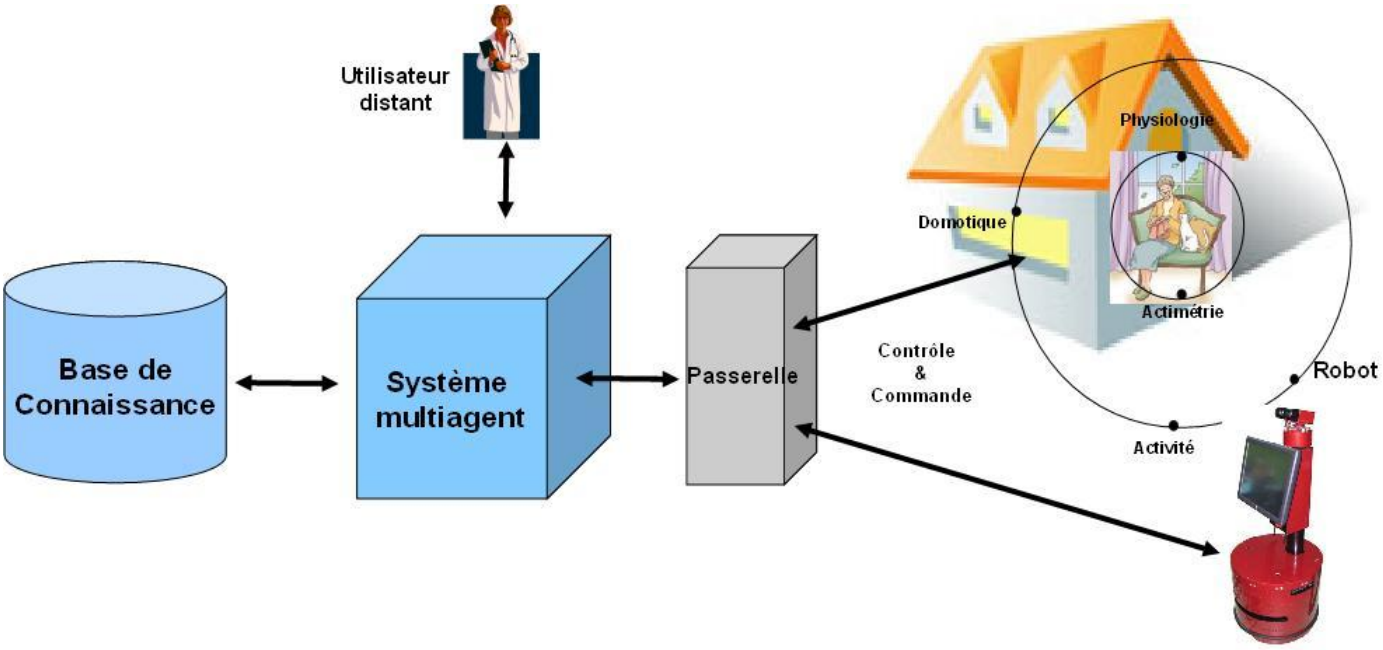

Figure III.2 : Schéma général de coopération robot-environnement dans le cadre de l'assistance à la personne

Utilisateurs

Les acteurs principaux qui seront les utilisateurs finaux du système sont le patient et sa famille, les producteurs de soins et les techniciens.

SMA

Le SMA-CRE raisonne à partir des informations qu'il reçoit de la base de connaissances et des capteurs pour faire émerger une solution à un effet souhaité émis. Le SMA offre un dynamisme et une flexibilité, qui permet d'appréhender les situations imprévues dans l'environnement. Le principe des coalitions d'agents dans un SMA est de permettre à des agents de se regrouper temporairement afin d'atteindre un objectif conjointement décidé et commun. La coalition offre ainsi un mécanisme d'adaptation qui apporte une réponse face à des situations diverses afin de répondre aux situations imprévues.

Environnement ambiant

L'environnement ambiant est constitué de capteurs, d'actionneurs et du robot en réseau. Ces objets sont capables de communiquer pour fournir des données (mesures) ou en produire une suite de commandes. Dans notre application, l'environnement est constitué surtout de capteurs destinés au service de télévigilance. Les capteurs infrarouges qui

détectent la présence d'une personne, les capteurs de passage de porte, les dalles actimétriques sensibles au passage d'une personne qui peuvent aussi détecter certaines postures (debout, allongé). On peut également citer les caméras orientables (ou non) de l'environnement, les lecteurs et tags RFID et enfin, les capteurs embarqués sur le robot (tels que les ultrasons, le laser, la caméra mobile, ou l'odomètre). Il est à noter que suivant son état, la personne âgée ou handicapée, peut porter des capteurs qui délivrent des informations physiologiques (ex. pouls, glycémie), actimétriques ou d'alarme (détection d'une chute).

Base de connaissances

Cette base de connaissances regroupe toutes les informations persistantes qui peuvent être utiles au SMA pour modéliser et exploiter les ressources de l'environnement ambiant et répondre aux demandes d'effets. Ces informations sont organisées en classes, représentant ainsi les principaux objets de l'application ; habitat, objet communicant, tâche et être vivant.
Des relations intra et inter membres des classes donnent la possibilité au SMA d'établir des liens entre ces informations. Cette base de connaissances est complétée par les informations dynamiques en provenance de l'environnement ambiant via la passerelle.

Passerelle

La passerelle est un module destiné à l'uniformisation des informations issues de l'environnement ambiant pour la perception des agents. Cette uniformisation est nécessaire à cause de l'hétérogénéité des protocoles des différents constructeurs. Ainsi, le SMA perçoit et agit sur l'environnement ambiant au travers de la passerelle sans se soucier du format des données perçues.

III.4 Scénarios

Pour illustrer les concepts et les solutions retenues dans ce chapitre, nous nous sommes appuyés sur deux scénarios. Un scénario ECLAIRAGE, l'effet recherché est un éclairage en un point donné d'une pièce. Un scénario LOCALISATION, plus proche de notre problématique, l'effet recherché est la localisation du robot dans l'habitat. Dans le scénario LOCALISATION, la précision du résultat et le niveau d'intrusion du système vis-à-vis de la personne sont pris en compte, alors que pour le scénario ECLAIRAGE, pour des raisons de simplification, nous ne les avons pas pris en compte.

III.4.1 Notion d'effet

Le SMA lance la recherche d'une solution lorsqu'une requête pour obtenir un effet particulier est formulée, l'effet peut être par exemple la recherche d'éclairage d'une zone ou la localisation d'un robot. Le SMA se configure de sorte à fournir une solution en fonction des objets communicants disponibles en recherchant la meilleure solution au sens des critères retenus. Notons qu'il ne s'agit pas de rechercher une solution optimale mais une solution qui s'en rapproche. L'adaptation au contexte est inhérente à la modélisation multiagent, renforcée par des mécanismes de formation de coalitions et de négociation.
D'un point de vue formel, un effet est décrit par un libellé (ECLAIRAGE, LOCALISATION,...), un niveau d'intrusion maximal toléré, le niveau de précision de l'effet souhaite, plus un ensemble de paramètres directement liés à l'effet souhaité (point d'une pièce à éclairer, entité à localiser, ...)

III.4.2 Scénario ECLAIRAGE

L'objectif est d'exploiter les objets communicants capables de générer de la lumière et qui soient à proximité d'une zone à éclairer. Les différents objets possédant leurs propres capacités d'éclairage, le but est de chercher une combinaison de ces capacités permettant d'obtenir l'effet recherché. L'influence des objets communicants dépend de leurs propres caractéristiques, de la distance et de la position par rapport au lieu à éclairer. Il est à noter que seuls les objets communicants disponibles participent à la recherche de solutions.
Par exemple, on aimerait éclairer une pièce avec un certain niveau de puissance. Cet effet peut être obtenu de différentes manières. Soit une lampe, qui se trouve à proximité du lieu à éclairer et qui possède directement cette puissance, soit plusieurs objets communicants capables de produire de la lumière (téléphone portable, voyant lumineux d'une free-box, etc.) qui, en combinant leur action, atteignent ce niveau en ce lieu. Tout le problème est de faire émerger une solution qui soit appropriée au contexte. L'adaptation au contexte dans ce scénario consiste à prendre en compte les informations disponibles, les contraintes liées aux capacités de chaque objet communicant et le lieu de la pièce où est désiré cet effet.

III.4.3 Scénario LOCALISATION

A son domicile, la chute d'un patient a enclenché une alarme de chute. Le robot, à la réception de ce signal doit se déplacer vers la personne. Pour cela, il a d'abord besoin de connaître sa propre position de façon exacte.
L'objectif est de localiser le robot dans l'habitat d'une personne. Certains des capteurs de l'environnement ambiant délivrent des informations de localisation plus ou moins précises. Le but est de chercher une combinaison de ces mesures pour répondre à l'effet demandé :

localiser le robot avec une précision dans un temps donné. Dans ce scénario, l'adaptation au contexte est plus complexe que dans le scénario précédent. En plus de prendre en compte les informations disponibles et les facteurs liés aux capacités de localiser de chaque objet communicant, nous avons ajouté les contraintes liées au service à la personne, le niveau d'intrusion et le degré d'urgence.

Nous développons dans la suite du chapitre, la base de connaissances et le SMA.

III.5 Base de connaissances

Une base de connaissances regroupe toutes les informations persistantes qui peuvent être utiles au SMA pour modéliser et exploiter les ressources de l'environnement ambiant et réaliser les services demandés. Nous avons identifié quatre catégories d'informations liées au domaine d'application : l'habitat pour définir la structure de l'environnement, les objets communicants pour connaître leurs caractéristiques et leurs conditions d'utilisation, les utilisateurs pour leur profil et les tâches et services que le système est susceptible de réaliser.
En outre, le système, pour répondre à une demande de service, a besoin d'établir des liens entre membres d'une même catégorie ; telle que par exemple, une relation topologique entre deux pièces de l'habitat. Des liens sont également nécessaires entre des membres de catégories différentes. Par exemple, pour exploiter une information fournie par un objet communicant, le système doit le localiser dans l'habitat.

III.5.1 Généralités sur les ontologies

L'origine du terme ontologie vient du grec "ontos" (ce qui existe, l'existant) et logos (le discours, l'étude). Étymologiquement, l'ontologie définit donc l'étude des propriétés générales de tout ce qui est. Initialement, ce terme qui est généralement utilisé en philosophie selon [Kivela2002] et [Smith2001] a évolué vers l'intelligence artificielle pour décrire formellement des modèles de domaines. La norme W3C[1] décrit les ontologies comme "une description et représentation d'un champ d'expertise. Les ontologies associent les concepts de base d'un domaine précis et les relations entre ces concepts d'une manière compréhensible par les machines". Nous retiendrons comme définition que l'ontologie est une description d'un domaine de connaissance particulier avec un ensemble de concepts et de relations entre ces concepts.
Les utilisations des ontologies sont nombreuses et variées mais les plus répandues sont classées, selon Uschold, en trois catégories : la communication, l'interopérabilité et l'ingénierie

1. W3C : Le World Wide Web Consortium est un organisme de standardisation chargé de promouvoir la compatibilité des technologies du World Wide Web.

des systèmes [Uschold1996]. En 2004, Le W3C classe principalement les cas d'utilisation des ontologies en six domaines (figure III.3).

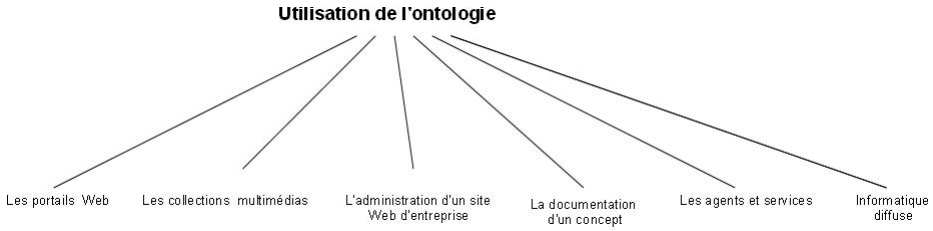

Figure III.3 : Cas d'utilisation de l'ontologie selon le W3C

Selon notre objectif dans cette thèse, les cas d'utilisation qui nous intéressent sont les agents et services ainsi que l'informatique diffuse. Dans le domaine des agents et services, les ontologies offrent aux agents la capacité de comprendre et d'intégrer des ressources d'informations diverses tandis que dans le domaine de l'informatique diffuse les ontologies sont utiles pour décrire les caractéristiques des appareils, les méthodes pour accéder à ces appareils, la politique d'utilisation de l'appareil établie par son propriétaire et les autres contraintes et conditions techniques qui affectent l'incorporation de l'appareil dans un réseau de type informatique diffuse.

En plus de permettre de structurer et relier les informations entre elles, il existe de nombreux outils logiciels, développés et mis à jour par la communauté, permettant de manipuler, d'exploiter et de faire évoluer ces ontologies. Nous avons opté pour la conception d'une ontologie adaptée au domaine de l'Assistance Ambiante, inspirée de celle développée par le laboratoire DOMUS [Fatiha2007b]. Dans ce projet, l'objectif est de permettre, l'observation et l'apprentissage des comportements d'un patient. Dans [Fatiha2007a], les auteurs décrivent une ontologie constituée de sept sous-ontologies :

- Habitat pour décrire la structure globale de la maison.
- Équipement [Fatiha2007b] qui regroupe les capteurs servant à la détection du comportement de la personne, les actionneurs et aussi l'ensemble des équipements ménagers.
- Personne, pour définir son profil notamment médical.
- Comportement, qui énumère les différents comportements du patient. Ces informations sont nécessaires pour établir par apprentissage les habitudes de vie de la personne.
- Tâche, qui sert à décrire aussi bien les activités du patient que celles des aidants.
- Décision, qui caractérise les différentes actions suivant chaque comportement pour assurer la sécurité du patient.
- Applications logicielles.

Nous nous sommes basés sur ce principe de raisonnement pour décrire notre base de connaissances dédiée à l'application d'Assistance Ambiante que nous appellerons "*Onto-*

logie pour l'Assistance Ambiante" ou "*Ontologie AA*". L'ontologie AA distingue l'aspect connaissance, de l'exploitation de ses connaissances. Elle ne regroupe que les connaissances persistantes. Une partie de ces connaissances est utilisée pour générer initialement le SMA tandis qu'une autre partie sera exploitée pour la réalisation des tâches par le SMA. Ainsi, cette solution ne préjuge pas de la manière avec laquelle ces connaissances seront exploitées.

On distingue généralement deux entités au sein d'une ontologie. La première, à objectif terminologique, définit la nature des éléments qui composent le domaine de l'ontologie en question, identique à la définition d'une classe en programmation orientée objet qui définit la nature des objets que l'on va manipuler par la suite. La seconde partie d'une ontologie explicite les relations, appelées propriétés, entre plusieurs instances de ces classes définies dans la partie terminologique. Ainsi, au sein d'une ontologie, les concepts sont définis les uns par rapport aux autres (modèle en graphe de l'organisation des connaissances), ce qui autorise un raisonnement et une manipulation de ces connaissances.

Dans une ontologie, on définit :

- Les concepts ou classes qui réunissent un groupe "d'individus" parce qu'ils ont des caractéristiques similaires.
- Les instances de classe qui définissent un "axiome d'individu" ou un "fait". Ce sont aussi les feuilles d'une classe. Exemple : L'instance d'un Patient dans une ontologie peut être un patient nommé Pierre ou un patient nommé Bob.
- Les propriétés définissent ce qu'on appelle une relation en ontologie. Pour définir une relation, il faut définir ce qu'on appelle un triplet `RDF` [2] (le domaine, la propriété, l'image). Par exemple, pour un capteur de son, une propriété APPARTIENT A peut avoir pour domaine une classe CAPTEUR SON et pour image une classe PIECE (chambre) : elle reliera ainsi des instances de la classe CAPTEUR SON à des instances de la classe PIECE. Il peut exister aussi des relations entre des attributs ou directement des instances de classes.

Figure III.4 : Triplet rdf

2. RDF : Resource Description Framework, Développé par le W3C, RDF est le langage de base du Web sémantique. est un modèle de graphe destiné à décrire de façon formelle les ressources Web et leurs métadonnées, de façon à permettre le traitement automatique de telles descriptions.

Les propriétés peuvent être symétriques, transitive, fonctionnelle, inversement fonctionnelle. Parmi toutes ces relations la transitivité est très utile dans notre ontologie parce qu'elle permet d'inférer plusieurs suites de relations entre des individus. Si on définit une propriété P comme transitive, alors pour tout x, y, z : P(x,y) et P(y,z) ⇒ P(x,z)

Sur l'exemple de la figure III.5, on a :
Appartient A(Capteur son, mur 21) et *Appartient A*(Mur 21, Pièce 1) ⇒ *Appartient A*(Capteur son, pièce 1).

Figure III.5 : Exemple de transitivité

III.5.2 Ontologie Assistance Ambiante

La base de connaissances contient toutes les informations relatives aux services qu'on veut rendre à la personne ainsi que celles nécessaires aux différentes actions que le robot pourrait exécuter. Elle comprend aussi à la fois la description des composants du domaine qui sont les personnes, l'habitation et l'ensemble des capteurs et actionneurs mais aussi les tâches et les services.

III.5.2.1 Propriétés d'une ontologie

Pour les besoins de notre application, nous avons distingué et dénommé trois types de propriétés ontologiques :

- Relation.
- Usage.
- Attribut.

III.5.2.1.1 Relation

La propriété ontologique *Relation* définit une relation logique, généralement d'appartenance, qui lie les concepts ou les individus de l'ontologie AA entre eux.

III.5.2.1.2 Usage

La propriété ontologique *Usage* définit l'utilisation possible d'un objet. Cette propriété spécifie si un agent peut utiliser cet objet pour participer ou non à un effet souhaité. Par exemple un détecteur de présence possède comme Usage Localisation. C'est à dire que ce capteur peut être exploité par les agents pour un effet souhaité de localisation.

III.5.2.1.3 Attribut

La propriété ontologique *Attribut* désigne la caractéristique d'un concept, d'un sous-concept ou encore d'un individu de l'ontologie AA. Par exemple, un capteur possède comme attributs : la grandeur mesurée et la précision de la mesure.
Nous avons structuré l'ensemble des informations persistantes en quatre concepts qui sont :

- Être Vivant.
- Habitat.
- Objet Communicant.
- Tâche.

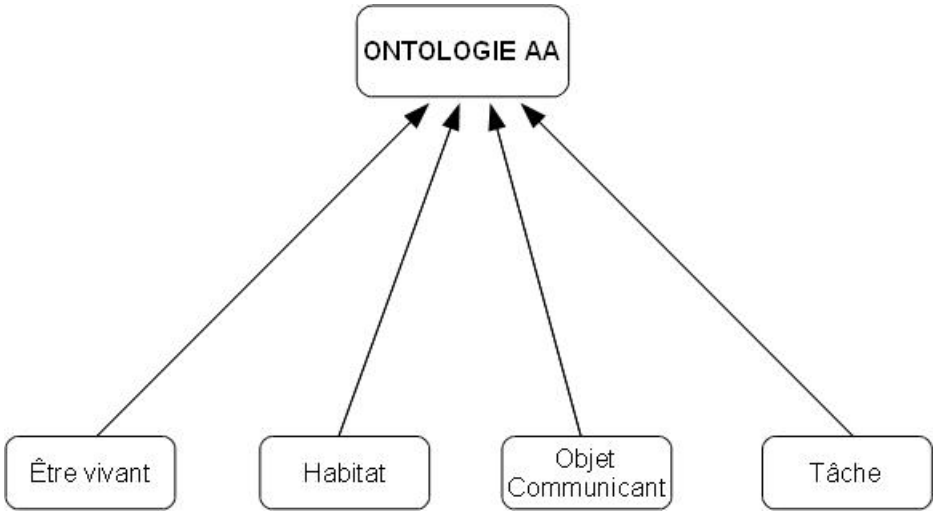

Figure III.6 : Concepts de l'ontologie AA

Le concept Être Vivant définit le profil des utilisateurs du système. Le concept Habitat décrit la structure de l'environnement dans lequel habite la personne pour laquelle le système est conçu. Le concept Objets Communicant décrit les objets constituant l'environnement ambiant, leurs caractéristiques et leurs conditions d'utilisation. Enfin le concept Tâche, décrit les tâches et services que le système est susceptible de réaliser.

Nous avons surtout développé les concepts Habitat et Objet Communicant car leur description exhaustive est indispensable à l'évaluation de notre architecture de coopération et des mécanismes d'adaptation. Nous commencerons donc par leur description.

III.5.2.2 Concept Habitat

Il existe un consensus général sur le fait que les différents types de modèles pour représenter un habitat ont des avantages et des limitations et qu'ils sont plus ou moins adaptés selon la mission à accomplir. Par exemple, les cartes métriques sont difficiles à construire

et à maintenir à cause de l'incohérence entre le mouvement du robot et la perception (si le modèle est initialisé par un plan d'architecte, cette étape se résume à une mise à jour). Elles sont moins adaptées pour les problèmes symboliques. Les cartes métriques sont en général très génériques. Elles peuvent a priori représenter tout type d'environnement. La construction des cartes métriques nécessite la gestion de la cohérence géométrique de l'ensemble de la représentation ; en particulier lorsque le robot retourne sur un lieu connu après avoir réalisé une boucle dans l'environnement (fermeture de la boucle). En effet, la fermeture de la boucle est l'un des défis majeurs de la construction des cartes métriques. Les cartes topologiques sont mieux adaptées pour représenter les grands environnements, pour ajouter un niveau symbolique ou pour communiquer avec l'homme. Mais ces cartes permettent seulement au robot de se localiser de manière globale, et de planifier la trajectoire de façon sous optimale. Elles se présentent sous forme de graphes, dont les sommets correspondent à des lieux, souvent associés à des informations perceptuelles (histogrammes de couleurs, images, données télémétriques, etc.), et dont les arêtes indiquent l'existence d'un chemin traversable par le robot, reliant les lieux associés aux deux sommets de l'arête. L'avantage principal des cartes topologiques est de s'abstraire des problèmes d'incertitudes dans le mouvement des robots : les incertitudes ne s'accumulent pas globalement, car le robot se contente de naviguer localement entre endroits (navigation topologique). Ces cartes peuvent présenter un découpage de l'espace qui facilite l'interaction avec l'homme, notamment si les lieux correspondent à des pièces ou à des couloirs. Par exemple, on peut imaginer de donner l'ordre au robot d'aller à la salle de séjour plutôt qu'aux coordonnées cartésiennes (x,y) ou bien encore "aller vers un marqueur".
Les modèles de primitives sont basés sur des parties distinctives de l'environnement que l'on peut facilement extraire via un type de capteur donné et qui admettent une description paramétrique (RFID). Notons qu'on parle généralement de balises quand il s'agit d'amers artificiels. Un amer géométrique pour la localisation doit vérifier plusieurs critères : pouvoir discriminant, domaine de visibilité important, stabilité, invariance et bonne adaptation à la métrologie. Et afin de maintenir une représentation cohérente de l'environnement, les cartes d'amers doivent de plus modéliser les incertitudes sur ces primitives géométriques. En général, ces incertitudes sont représentées par des distributions gaussiennes sur les paramètres géométriques de la primitive (positions cartésiennes ou polaires, longueurs, etc.). La représentation par amers fournit un cadre de travail mieux adapté pour résoudre le problème de la Cartographie et Localisation Simultanées (SLAM). Ces représentations s'avèrent souvent complémentaires. C'est le cas par exemple, des approches métriques et topologiques. En particulier, leur utilisation conjointe est susceptible de favoriser l'exploitation de la carte résultante pour les besoins de navigation du robot. C'est pourquoi les approches hybrides, qui combinent différents types de modèles élémentaires, se généralisent mais le maintien de la cohérence entre les représentations complémentaires reste difficile.

En considérant que dans le domaine applicatif qui nous intéresse, l'habitat des personnes en perte d'autonomie présente des caractéristiques très variables :

- Pièce de dimensions faibles au domicile, à importantes en institution ;
- pièce présentant un encombrement important ;
- environnement dynamique, de nombreux objets peuvent être déplacés (table, chaise) ;
- présence d'objets mobiles (robot aspirateur) ;
- présence d'êtres vivants (humain, animaux) ;
- présence d'objets non coopérants (meubles avec des pieds et un plateau).

Nous avons opté pour une stratégie de déplacement du robot topologique complétée par des comportements autonomes du robot. Ces comportements répondent à des situations locales de type évitement d'obstacles, passages difficiles (portes ...), accostage.

Nous avons donc adopté un modèle de l'environnement simple de type topologique, enrichi par des propriétés ontologiques de type *Attribut*. C'est un modèle qui est capable d'accepter une complexité croissante selon les besoins. Le modèle topologique de base donne les relations entre les pièces, les murs et des éléments caractéristiques comme les portes. La granularité peut être augmentée en déterminant des zones à l'intérieur des pièces à partir de la position des capteurs ou des marqueurs placés dans l'environnement. Si nécessaire et si l'information est disponible, on peut ajouter des propriétés ontologiques de type *Attributs* géométriques (position, orientation) à certains individus du modèle.

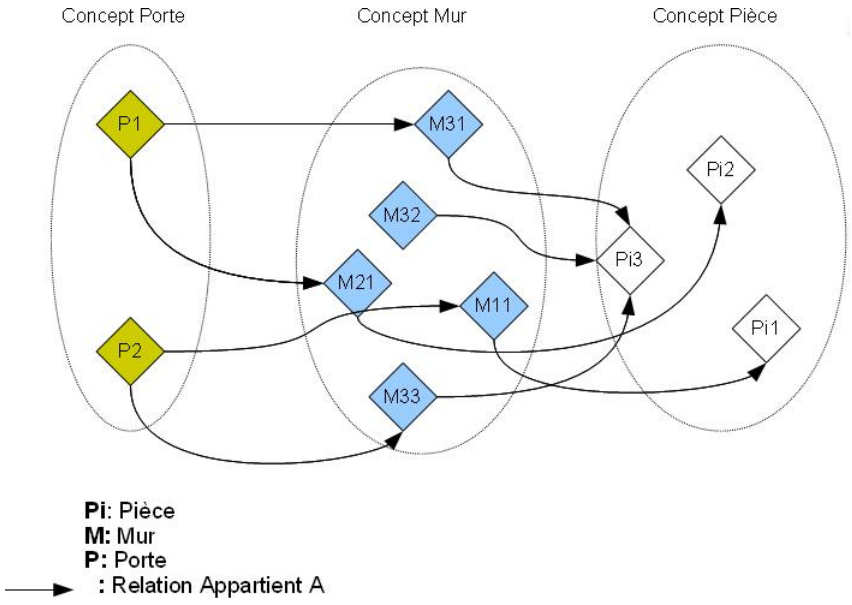

Figure III.7 : Concept Habitat

La figure III.7 représente une vue partielle du concept Habitat et ses différentes composantes. Une maison est constituée de pièces qui sont composées de murs. Certains murs sont percés par des ouvertures. Il existe des relations *Appartient A* entre les individus de

chaque sous-concept de Habitat.
Par exemple, dans le scénario de localisation, si on connaît l'identifiant des murs ou des portes grâce à des marqueurs, le robot peut déduire logiquement la pièce dans laquelle il se trouve grâce aux relations d'appartenances. De même, une Relation nommée Communique permet de déduire par exemple que pour aller à une Pièce 1, il faut passer par la Porte 2. Par exemple, il est possible d'inférer que la pièce 3 est en communication avec la pièce 2 par une porte commune.

III.5.2.3 Concept Objet Communicant

Le concept objet communicant réunit l'ensemble des capteurs et des actionneurs qui constituent l'habitat du patient de la personne. Une première propriété ontologique, définie précédemment, affectée à chacun de ces objets est l'*Usage*. Cette propriété est utilisée au niveau des agents pour déterminer s'ils peuvent participer à l'obtention d'un effet demandé. Les caractéristiques des objets communicants sont précisées par un ensemble de propriétés ontologiques *Attributs*. Cette connaissance est nécessaire à l'agent pour exploiter correctement cet objet. Par exemple, un capteur de type Détecteur Infrarouge possède plusieurs attributs :

- identifiant : pour différencier chaque objet communicant des autres, un identifiant est unique et est affecté à chacun d'eux. Comme il y a plusieurs types de capteurs, l'identifiant contient un champ identique pour caractériser le type de capteur et un autre champ qui est le numéro du capteur dans sa catégorie,
- portée : définit la distance maximale que le capteur peut couvrir pour détecter une présence. Par exemple 2 m,
- précision : La précision définit une zone d'intervalle de détection dans l'angle d'ouverture du capteur. Par exemple, l'angle d'ouverture peut varier entre 1 et 170° avec une précision de plus ou moins 3°.
- temps de réponse : désigne le temps écoulé entre la détection d'une nouvelle donnée captée par le capteur et la présentation de cette dernière. Par exemple 50 ms,
- position : définit la localisation du capteur dans la pièce,
- orientation : désigne la direction du cône de détection du capteur. Il est défini par un angle par rapport à un référentiel. Par exemple, le détecteur de présence est dirigé d'un angle de 30° par rapport à l'horizontal parallèle au long de la pièce.

Enfin la propriété de type *Relation* (figure III.8) permet d'établir des liens avec les autres concepts de l'ontologie AA. Par exemple, pour localiser les capteurs statiques de l'environnement, nous avons établi une relation d'appartenance entre chaque capteur et un élément de l'habitat.

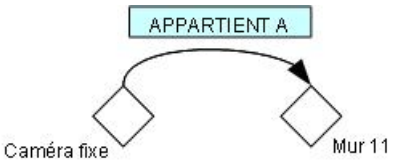

FIGURE III.8 : Exemple de relation

En résumé, quand une requête pour obtenir un effet souhaité est formulée, les agents du SMA interrogent l'ontologie. La propriété *Usage* leur permet d'identifier les objets communicants pertinents. Les *Attributs* précisent les caractéristiques d'utilisation et les *Relations* permettent d'inférer à une localisation d'un objet communicant.

III.5.2.4 Concept Tâche

Le concept Tâche (figure III.9) regroupe les différents services à la personne comme la stimulation cognitive, mais aussi les tâches nécessaires au déplacement du robot (localisation, navigation).

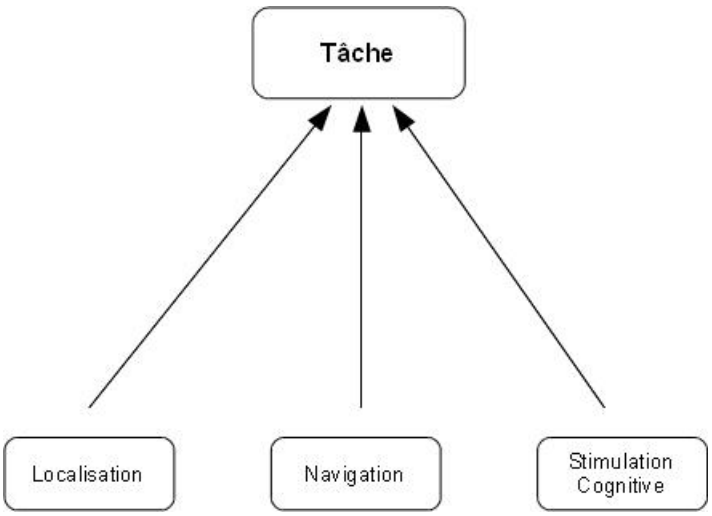

FIGURE III.9 : Concept Tâche

III.5.2.5 Concept Être Vivant

Le concept Être vivant (figure III.10) regroupe les différentes entités qui peuvent être présentes dans l'environnement ambiant, on y trouve les acteurs comme les producteurs de soins, les techniciens ou encore les proches. Dans ce même concept, on inclut la personne.

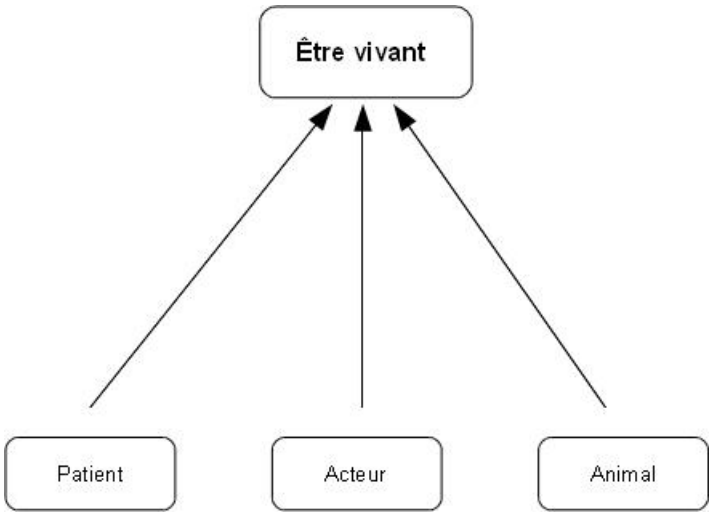

Figure III.10 : concept Être vivant

III.5.2.6 Distance topologique

La distance topologique (figure III.11) est définie comme étant le nombre de sauts "abstraits" entre deux individus de l'ontologie. Les sauts abstraits étant des relations telles que définies précédemment dans l'ontologie.
Si la structure de l'ontologie est définie par un graphe, la distance topologique est le nombre de nœuds qui séparent deux individus moins un.
Soient I_i et I_j deux noeuds quelconques du graphe et dist(I_i, I_j) la fonction qui calcule la distance topologique entre I_i et I_j. Si n est le nombre de nœuds séparant les deux individus i et j dist(I_i, I_j) = n - 1.

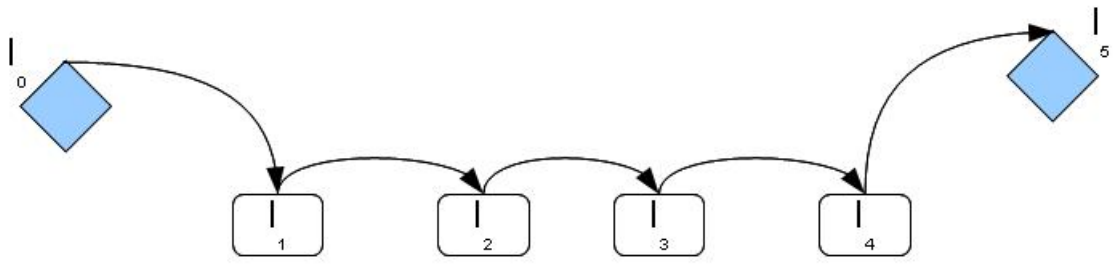

Figure III.11 : Exemple d'une distance topologique

Sur la figure III.11, dist(I_0, I_5) = 5. En effet, le chemin entre les individus I_0 et I_5 contient 6 noeuds soit 5 relations.

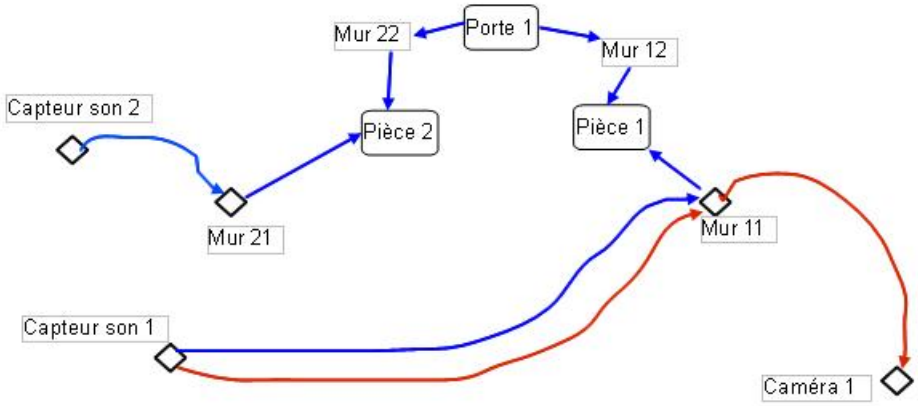

Figure III.12 : Exemple d'une distance topologique

Un autre exemple, sur la figure III.12 représente la distance topologique entre le capteur son 1 et la caméra 1 est de 2 (en rouge). Alors que la distance topologique entre le capteur son 2 et le capteur son 1 est de 7 (en bleu). La distance topologique est utilisée par les agents du SMA pour déterminer leur voisinage lors de la formation des coalitions.

III.6 Architecture du système multi-agent pour la coopération robot-environnement

Le Système Multi-Agent pour la Coopération Robot-Environnement (SMA-CRE) est constitué de plusieurs agents hybrides. Chaque agent encapsule un objet communicant tel qu'un capteur ou un actionneur.

Le système est composé d'agents que nous appellerons Agent Ambiant pour la Coopération Robot Environnement (AA-CRE).

III.6.1 Modèle d'Agent AA-CRE

Par définition, un AA-CRE est un agent autonome embarqué sur un objet communicant de l'environnement (figure III.13) ou d'un robot et qui peut interagir avec l'environnement réel et avec d'autres agents. Les objets communicants transmettent aux agents les données capturées via la passerelle depuis l'environnement physique.

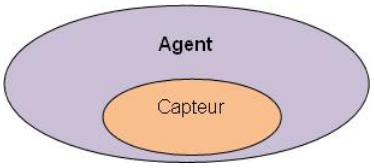

Figure III.13 : Modèle schématique d'un agent AA-CRE

Un AA-CRE est composé de trois principaux modules qui sont l'Entrée, la Sortie et le module de Décision. La figure III.14 montre l'ensemble de ses composantes.

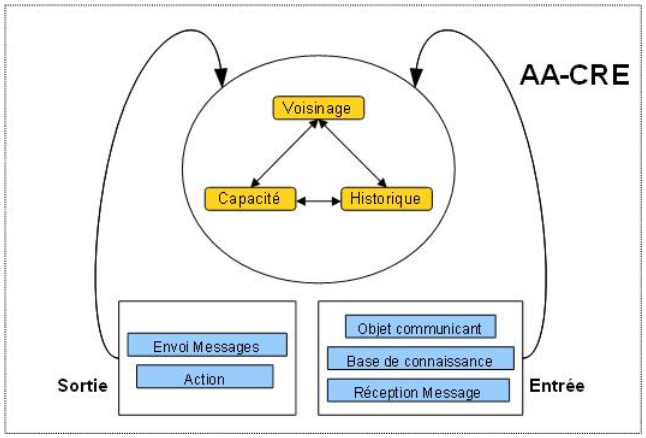

Figure III.14 : Modules composant un agent AA-CRE

Entrée

Le module *Entrée* regroupe les différentes propriétés de l'agent qui sont reprises depuis l'ontologie. C'est ce module qui fait l'acquisition ou la perception des données des objets communicants et de l'ontologie. Ce module a aussi pour rôle la perception de l'agent et la communication entre eux.

Sortie

Le module *Sortie* est destiné aux envois de messages entre les agents et aussi pour actionner l'objet communicant de type actionneur qu'un agent encapsule.

Décision

Le module *Décision* est essentiel car c'est lui qui assure son adaptativité et sa réactivité. Ce module est composé de trois paramètres qui sont le Voisinage, l'Historique et la Capacité.

Voisinage : Le voisinage définit la liste des agents qui sont voisins immédiats d'un agent à un instant donné. On appelle voisinage, tous les agents qui, à la réception

d'un effet souhaité, se trouvent à une certaine distance topologique (cf La notion de distance topologique) proche de l'agent. La proximité est définie par l'appartenance à une même pièce.

Historique : L'historique est un moyen qui permet aux agents de sauvegarder des informations essentielles a priori de leurs tâches précédentes. C'est la sauvegarde des données perçues jusqu'à un temps donné ou l'ensemble des expériences d'interaction qui ont été réalisées avec satisfaction.

Capacité : La capacité est la relation entre un ou plusieurs propriétés du capteur encapsulé par l'agent et un effet que le capteur peut produire grâce à ces propriétés.

III.6.2 Protocole de formation de coalitions

Le protocole SMA de Coopération Robot-Environnement qui est conçu dans ce travail est défini comme un ensemble de règles que les agents AA-CRE suivent pour faire émerger une solution. Le protocole est basé principalement sur la formation de coalitions. Le protocole de Coopération Robot-Environnement est composé de deux étapes distinctes qui sont : la formation de coalitions proprement dite qui consiste à former des groupes d'agents selon leur capacité. La deuxième étape est une phase de négociation entre chaque coalition qui s'est formée pour choisir la meilleure qui puisse satisfaire l'effet souhaité.

L'objectif des agents AA-CRE est de former des coalitions. Nous allons décrire le système dans lequel les agents évoluent. Une coalition est formée lorsque chacun de ses agents accepte d'en faire partie. Les agents doivent s'entendre sur les coalitions à former et valider leur formation. Lorsqu'un agent est intéressé par la formation d'une coalition, le protocole doit lui garantir que sa proposition sera effectivement transmise aux autres membres envisagés. Les autres agents membres doivent également être dans la capacité de lui répondre.

Dans la formation de coalitions, un agent peut être initiateur ou candidat ; par définition, un agent initiateur est celui intéressé par une coalition et qui est à l'origine de l'échange des messages avec les autres agents membres de la coalition tandis que les autres agents de la coalition, à qui l'agent initiateur propose la coalition sont appelés agents candidats.

Le protocole est basé sur des échanges de messages entre l'agent initiateur et les agents candidats. Puisqu'un agent initiateur ne peut forcer les agents candidats à former une coalition, le protocole de formation d'une coalition doit comporter un minimum de trois règles :

1. Les agents candidats doivent prendre connaissance de leur coalition.
2. Les agents candidats doivent indiquer à l'agent initiateur s'ils veulent former la coalition.
3. Les agents candidats n'interagissent qu'avec l'agent initiateur.

III.6.3 Algorithmes AA-CRE

L'algorithme de base d'un agent, par le biais du module Décision, utilise sa capacité et son voisinage pour la formation de coalitions et la sélection de la meilleure pour un initiateur. Les agents forment des coalitions suite à une demande externe pour la résolution d'un problème donné, c'est le point de départ pour les agents.

Le protocole SMA de coopération robot-Environnement est basé en trois séquences d'envoi de message :

1. L'agent initiateur envoie une proposition de formation de la coalition aux agents candidats.
2. Les agents candidats indiquent à l'agent initiateur s'ils sont intéressés par la formation de la coalition.
3. (a) Si tous les agents candidats sont intéressés par la formation de la coalition, la coalition est formée et l'agent initiateur envoie une confirmation de formation de la coalition. (b) S'il existe au moins un agent candidat non intéressé par la formation de la coalition, l'agent initiateur envoie une annulation de formation de la coalition.

Dans la suite sera détaillée la communication entre les agents. En effet, pour former des coalitions en fonction d'un effet souhaité, les agents doivent communiquer par l'intermédiaire d'un échange de messages.

Interaction entre les agents

Pour la formation de coalitions, considérant les dialogues entre les agents qu'il faut établir, trois principaux types de messages sont définis : Les messages de type Requête, les messages de type Réponse et enfin les messages de type Information.

Messages de type Requête

Le type requête regroupe deux sous-types de messages qui sont :

- *Initialisation :* sous-type de messages qui regroupe la requête de l'agent AI qui formule une requête à tous les agents du système (*InitEffect*) et la requête qu'une fois toutes les coalitions formées, chaque agent initiateur envoie un message pour initier la négociation (*InitNegociation*).
- *Coalition :* un message envoyé à la réception d'un effet souhaité pour initier une formation de coalitions (*InitCoal*) est de ce sous-type.

Messages de type Réponse

Le type Réponse regroupe trois sous-types de messages qui sont :

- *Accusé de Réception :* un message de confirmation (*ACKCoal*) ou de refus (*RefuseCoal*) d'appartenance à une coalition est de ce sous-type.
- *Réaction :* ce sous-type regroupe essentiellement deux messages qui sont *AcceptCoal* qui est un message envoyé par un agent qui accepte une proposition d'*InitCoal*

venant d'un initiateur. Le deuxième message est *ArgNeg* qui est envoyé par un agent pour répondre à une demande de négociation.

Chaque type de message contient la capacité d'un agent au moment de la formation de coalitions et la capacité de la coalition au moment de la négociation de la coalition gagnante.

Un agent candidat peut répondre favorablement à plusieurs initiateurs tandis qu'un initiateur ne pourra faire partie que d'une et une seule coalition.

Soit :
- C_i 1<i<n
- i : indice d'agent,
- n : nombre d'agents
- C_{coal}, la capacité d'une coalition. La notion capacité d'une coalition est identique à celle de l'agent mais se réfère à la "somme" des capacités de tous les agents qui forment une coalition.
- InitCoal : Demande de faire partie d'une coalition
- C_n^p : Nombre de coalitions maximales qu'un initiateur peut former (équivaux le nombre de combinaison possible avec :
  - n : Nombre de voisin d'un agent
  - p : Nombre d'agents dans la coalition à former

L'algorithme 5 suivant représente la formation de coalitions par un agent initiateur.

pour tout effet souhaité σ **faire**
  si $C_i < \sigma$ **alors**
    $C_{coal} = C_i$
  finsi
  pour $i = 0$ à $i \leq C_n^p$ **faire**
    tantque $C_{coal} < \sigma$ **faire**
      Envoi InitCoal aux voisins
      Attendre réponse
      $C_{coal} = C_{coal} + C_i$
    fin tantque
  fin pour
fin pour

Algorithme 5: Formation de coalitions d'un agent

L'objectif d'un initiateur est de former une coalition, cependant, comme les initiateurs n'ont pas de préférence vis-à-vis des candidats voisins, il est ainsi possible pour un initiateur de créer plusieurs coalitions, sachant que le nombre d'agents dans une coalition

est inférieure à n (n étant le nombre de candidats maximaux du voisinage), pour au final choisir la meilleure formée.

Quant aux agents qui reçoivent une demande de participer à une coalition, ils exécutent l'algorithme 6 ci-dessous. Si leur capacité est compatible avec l'effet souhaité, ils envoient une acceptation de faire partie de la coalition.

pour tout Message *InitCoal* **faire**
  si C_i tend vers σ **alors**
    Envoi AcceptCoal à l'initiateur
  sinon
    Envoi RefuseCoal
  finsi
fin pour

Algorithme 6: Traitement d'une requête de coalition d'un agent

Exemple Considérant le scénario des lampes, avec un effet souhaité d'éclairer une pièce avec un seuil donné, chaque lampe est encapsulée par un agent qu'on appellerait Agent Lampe. La pièce comprend 5 lampes A_1, A_2, A_3, A_4, A_5. Un Agent Interface AI existe aussi pour assurer l'interfaçage avec l'utilisateur et le système. C'est ce dernier agent qui envoie les requêtes d'effet souhaité. Un agent Lampe aura donc comme capacité la valeur de l'intensité lumineuse de sa lampe ainsi que la distance à laquelle se trouve la lampe. Ainsi, chaque agent à la réception de cet effet souhaité va chercher à répondre favorablement à la requête tout seul ou, dans le cas où sa capacité est insuffisante, lancer une invitation pour créer une coalition. L'objectif de la coalition étant de rassembler le nombre maximum et suffisant d'Agents Lampes qui atteint le seuil demandé.

Étape d'initialisation du SMA

L'étape d'initialisation et de création du SMA est effectuée par un module particulier d'initialisation. Elle consiste à déclencher un comportement de scrutation de l'environnement de chaque agent et consulter l'ontologie pour créer les agents qui vont eux-mêmes s'initialiser en chargeant localement, un ensemble de données à partir de cette ontologie et des informations de l'environnement physique.

Traitement d'une requête de l'Agent Interface par un agent A_i

Chaque agent détermine sa capacité en fonction de son environnement. Si la capacité d'un agent A_i à l'instant t est insuffisante pour répondre à l'effet souhaité requis par l'Agent Interface, il initie la formation de coalitions pour répondre à la requête en envoyant un message à tous les agents pour leur demander de se joindre à lui (figure III.15). Il leur envoie dans ce cas sa capacité et il sera le référent de cette coalition. Ce message d'initia-

lisation est un message de type requête et est nommé *InitCoal*.

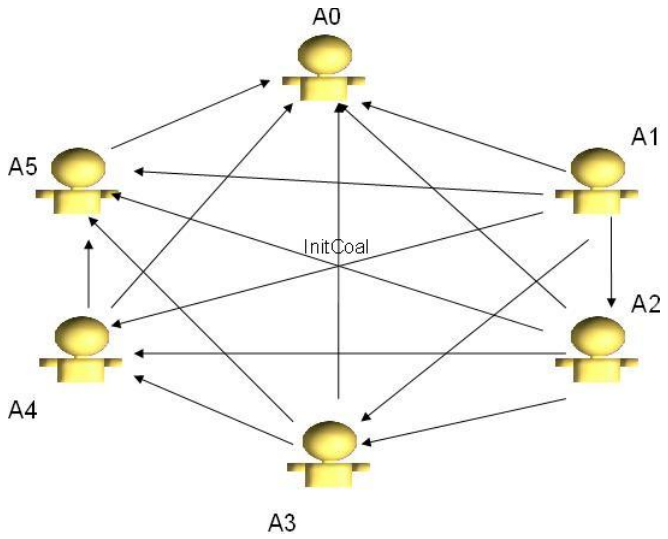

Figure III.15 : Envoi *InitCoal*

Dans le cas, où l'agent est capable à lui seul de donner une réponse satisfaisante à l'objectif recherché, il forme à lui seul une réponse à la requête de l'AI qui sera déjà une réponse possible puisqu'elle représente une capacité valable répondant aux contraintes données par l'AI.

Pour le cas des Lampes, si l'Agent Interface envoie une requête de la forme : Éclairer la pièce A avec un maximum de 90% d'une intensité lumineuse de 100 Lux. Un agent ayant une capacité de 30 Lux, va initier une coalition avec ses Agents Lampes voisins afin d'atteindre au maximum 90 Lux. Il envoie un message *InitCoal* contenant l'effet souhaité pour inviter ses voisins à faire partie d'une coalition dont il sera l'initiateur.

Traitement d'une demande de participation à une coalition en tant qu'agent candidat

Si un agent possède une compétence lui permettant d'avoir une capacité en lien avec la requête, il accepte de faire partie de la coalition et mémorise cette information localement. Il a accepté de former une coalition et envoie un message *AcceptCoal* puis s'attend à recevoir une confirmation de la part de l'initiateur avec l'identifiant de la coalition.
Pour des coalitions à quatre membres, avec un voisinage de cinq agents, un agent initiateur devra envoyer une demande *InitCoal* à trois candidats et en recevoir trois réponses.

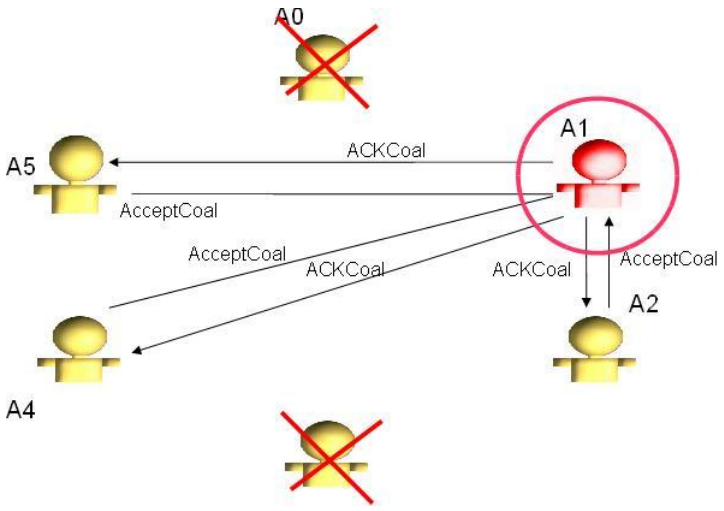

Figure III.16 : Formation d'une coalition pour l'initiateur A_1

Un Agent Lampe A_2 recevant un *InitCoal* de l'Agent Lampe A_1 vérifie si sa capacité, c'est à dire sa localisation par rapport au lieu à éclairer et son intensité lumineuse ne dépasse pas l'effet souhaité de 90 Lux. Dans ce cas, il envoie un message d'acceptation *AcceptCoal* qui contient sa capacité.

Traitement des réponses (suite aux demandes de former des coalitions) par les initiateurs

L'agent initiateur de coalitions accepte les réponses au fur et à mesure et cumule les capacités des agents qui ont répondu favorablement à son invitation en leur envoyant un message *ACKCoal*. Dès que le seuil est atteint, il clôture la coalition et envoie le nom de sa coalition à tous ses participants. Ces derniers vont mémoriser ce nom ainsi que l'agent référent de la coalition.

Dans le cas contraire, où le référent n'a plus besoin d'autres agents, dans ce cas, il refuse la proposition d'un participant à une coalition en lui envoyant un message *RefuseCoal*.

Quand l'Agent Lampe A_1 qui est l'initiateur de la coalition reçoit un *AcceptCoal*, il vérifie ajoute à sa capacité locale la capacité de l'agent A_2. C'est la capacité de sa coalition C_{coal1}. Et tant que cette capacité est inférieure à l'effet souhaité, il accepte tous les *AcceptCoal* qu'il reçoit et envoie un message de confirmation *ACKCoal*, sinon *RefuseCoal*.

Traitement des réponses de confirmation (de la part des initiateurs)

Un agent qui a reçu un *ACKCoal* d'un initiateur de coalitions mémorise le nom de la coalition et le nom de son référent.

Prise en compte du voisinage immédiat

Le voisinage immédiat décrit la liste de tous les autres agents qui se trouvent topologiquement proches. Les renseignements nécessaires à l'obtention de cette mesure proviennent de l'ontologie, plus précisément du concept habitat et du concept objet communicant. Une propriété de type *APPARTIENT A* relie chaque élément de ces deux concepts. Cette relation mesure l'éloignement topologique d'un capteur ou d'un actionneur par rapport à son emplacement.

Prise en compte du niveau d'intrusion

Le niveau d'intrusion a pour objectif de minimiser la gêne occasionnée sur le patient par les objets communicants et le robot. Ce paramètre permet de moduler les actions des capteurs ou des actionneurs de l'environnement qui sont gérés par les agents. Ainsi, le système fonctionne initialement avec un niveau d'intrusion 0, ce niveau minimal est le niveau de fonctionnement de base des objets communicants. Si au bout d'un temps donné (timeout), le SMA n'a pas réussi à former une coalition valable pour un effet souhaité recherché. Le niveau d'intrusion augmente d'un cran. Ce niveau d'intrusion à 1 permet de faire mouvoir les objets communicants qui peuvent le faire. Par exemple, la caméra mobile du robot.
L'augmentation de ce niveau d'intrusion relance le processus de formation de coalitions avec de nouvelles mesures parce que l'environnement de chaque agent aura changé. Enfin, le niveau d'intrusion maximal qui est le niveau 2 est le niveau qui permet de déplacer le robot dans l'habitat du patient. Encore une fois, le processus de formation de coalitions est relancé.

A un moment donné, les agents ne vont plus pouvoir former de coalitions, ils vont entrer dans un processus de négociation pour confronter les coalitions afin que la "meilleure" puisse être proposée à l'AI (Agent Interface). C'est l'optimum de Pareto (cf. paragraphe III.2.5.3).

Pour l'exemple des Agents Lampes. Ce niveau d'intrusion est matérialisé par le relâchement du niveau de précision de l'effet souhaité. C'est à dire qu'au lieu de rechercher initialement une intensité lumineuse maximale de 90% de 100 Lux, cette requête est relâchée avec seulement 70% du seuil. Car, si le SMA n'a pas pu former de coalition, cela signifie qu'aucun des agents n'a assez de capacité pour respecter la demande de l'Agent Interface et que malgré leurs demandes de former des coalitions, les différentes capacités de coalitions C_{coal} sont aussi insuffisantes. Ainsi, si aucune coalition n'a pas pu se former après le premier seuil, le système se relance automatiquement en diminuant le seuil d'intensité lumineuse.

Négociation

Une fois toutes les coalitions possibles formées la négociation est la phase pendant laquelle chaque initiateur de coalition négocie avec leurs semblables pour savoir quelle est la meilleure coalition possible de toutes celles qui se sont formées.
La négociation se base sur un critère dépendant directement de l'effet souhaité, mais aussi sur la capacité de coalition de chaque agent initiateur. Chaque agent initiateur envoie donc une invitation à la négociation par le message *InitNegociation* qui contient principalement C_{coal}. A la réception de ce message, un agent initiateur compare la capacité de coalition qu'il a reçu à la sienne, si l'agent récepteur fait parti d'une coalition qui à une capacité inférieure à celle reçue, sa coalition ne sera plus considérée, dans le cas contraire, c'est une coalition gagnante jusqu'à la réception d'une nouveau *InitNegociation*.

III.6.4 Comportements et contrôle interne d'un agent AA-CRE

III.6.4.1 Les comportements

Pour qu'un agent puisse prendre des décisions et respecter le protocole, un agent AA-CRE exécute les comportements adéquats et se met dans un état correspondant au comportement adopté. La figure suivante (figure III.17) montre les différents comportements d'un agent AA-CRE.

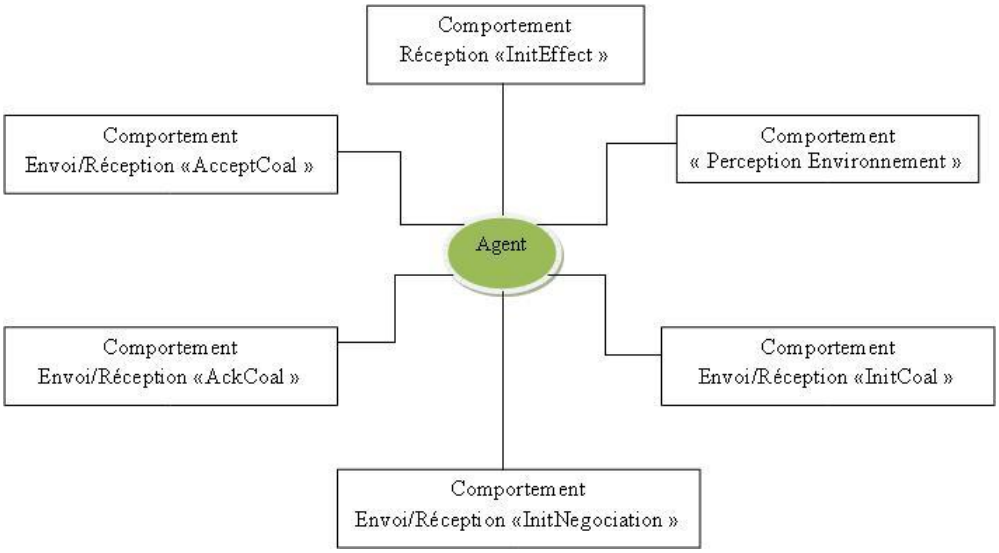

Figure III.17 : Schéma descriptif des comportements de chaque agent

Perception Environnement

Ce comportement de type cyclique permet à l'agent de percevoir, l'état de son environnement, à travers la passerelle et l'ontologie. C'est pourquoi, il existe deux méthodes de perception de l'environnement, l'interrogation de la base de connaissances et le capture des trames issues des capteurs.

Réception *InitEffect*

L'effet souhaité doit être communiqué aux agents via un message de la part de l'utilisateur ou de l'Agent Interface. Une fois que les agents sont créés et prêts pour résoudre un problème, ils attendent un effet souhaité en se mettant en attente d'un message *InitEffect*, une fois cet effet reçu ils exécutent le comportement dit "Réception *InitEffect*". C'est un comportement en parallèle au comportement "Perception Environnement".
Comme il est possible qu'un agent puisse traiter plusieurs effets souhaités simultanément, ce comportement est de type "Cyclique". Par exemple, on souhaite localiser un robot dans une et à un moment donnée une alarme de chute du patient se déclenche suite à un évanouissement.
La gestion de plusieurs *InitEffect* se fait cependant par la mise en place de niveau de priorité d'urgence dans chaque effet souhaité.

Envoi/Réception *InitCoal*

Suite a la réception d'un *InitEffect*, chaque agent teste sa capacité à résoudre le problème, si cette capacité est suffisante, il se lance directement dans la résolution de la requête. Mais comme dans la majorité des cas, les agents ont des capacités limités, ils doivent se coaliser avec d'autres agents afin de former des coalitions dont l'objectif est la résolution du problème en commun. Chaque agent exécute ainsi le comportement "Envoi *InitCoal*". Parallèlement à ce comportement, l'agent devra s'attendre à la réception d'un *InitCoal* de la part des autres agents, c'est le comportement "Réception *InitCoal*". Une fois qu'un agent aura envoyé sa demande de coalition, il se met en attente en exécutant le comportement "Réception *InitCoal*".

Envoi/Réception *AcceptCoal*

Dans le cas où un agent reçoit une demande de faire parti d'une coalition, pour exprimer son avis favorable il répondra par une acceptation, c'est le comportement "Envoi *AcceptCoal*". Comme l'agent en question peut aussi être un initiateur, il devra attendre les acceptations de ses candidats, il exécute alors le comportement "Réception *AcceptCoal*". A la réception d'un *AcceptCoal*, c'est à dire, la réponse favorable d'un agent pour faire partie de la coalition. L'agent initiateur envoie des *ACKCoal* aux agents qui ont répondu favorablement à l'initialisation de la coalition et si la capacité de cette dernière n'est pas encore suffisante pour l'effet souhaité.

Envoi/Réception *AckCoal*

Pour chaque initiateur recevant une acceptation de la part des autres agents voulant faire parti de sa coalition il répondra par une confirmation.

Envoi/Réception *InitNegociation*

Ce comportement, initie la phase de négociation entre chaque initiateur de coalition une fois que toutes les coalitions possibles ont été formées par le système. Les actions suivantes sont effectuées par chaque agent initiateur :

1. Calcul de la mesure récente pour la coalition sélectionnée de chaque agent initiateur.
2. Envoi d'un message *InitNegociation* contenant la mesure récente de la coalition aux initiateurs.

Si, toutes les coalitions ont échouées, essentiellement, cela signifie que l'environnement de chaque agent n'est pas à jour. D'où, le système augment le niveau d'intrusion I et relance de nouveau le processus de formation de coalitions.

En résumé, de ces échanges, il en découle les trois principaux actions suivantes :

1. Formation de toutes les coalitions possibles pour chaque référent.
2. Sélection de la meilleure coalition en fonction de la précision. La précision étant un attribut de l'agent issu des caractéristiques techniques du capteur qu'il encapsule. C'est la précision de détection du capteur.
3. Déclenchement du comportement *InitNegociation*.

Contrôle de l'état interne d'un agent

Chaque agent A_i peut être dans l'un des états suivants, à la réception d'un *InitEffect* :

- *État 0 :* État de veille, c'est l'état pendant lequel, un agent n'est pas dans une tâche de coalitions. Il ne fait que percevoir l'environnement.
- *État 1 :* Formation de coalitions en cours.
- *État 2 :* Négociation en cours.
- *État 3 :* transmission des résultats.

Des primitives sont utilisées par chacun des agents pour mettre en œuvre les comportements des agents et permettre l'interaction avec l'ontologie d'une part, et la passerelle d'autre part. Les principaux primitives des agents AA-CRE concernent leur module Décision et leur module Sortie.

Écoute des trames

C'est une fonction qui fait partie du comportement Perception de l'environnement, elle a pour rôle de réceptionner une trame de l'environnement et d'extraire l'identifiant de l'agent à l'initialisation. Un agent capture une trame en fonction du champ Identifiant de la trame. Pour qu'une trame puisse être destinée à un agent, il faut que ce champ Identifiant correspond au type de l'agent.

Accès à l'Ontologie et mise à jour de la capacité

La deuxième fonction du comportement perception de l'environnement est l'accès à la

base connaissance. L'identifiant acquis par l'agent pendant l'écoute des trames lui servira de clé pour l'accès à l'ontologie. Cette primitive permet à un agent de mettre à jour sa capacité. Une fois que l'agent de type détecteur de présence a acquis un identifiant, il accède à l'ontologie et prend tous les propriétés ou attributs qui le concernent pour mettre à jour sa capacité. Parmi ses attributs, on peut citer dans le cas du scénario Localisation : la précision, la localisation précise de la caméra dans l'environnement, son angle d'ouverture, etc ...

Mise à jour de l'historique

Le premier traitement de la requête de l'environnement reçu sert à mettre à jour la capacité de l'agent et les autres trames de données qui suivent seront stockés localement dans son Historique du module de Décision pour une utilisation ultérieure.

Recherche du voisinage

Pour pouvoir répondre rapidement à un effet souhaité, un agent initie une coalition avec des agents voisins. Pour se faire, l'agent utilise la distance topologique pour évaluer la proximité. Dans le scénario Localisation, un agent de type caméra fixe connaît sa distance topologique par rapport aux autres agents capteurs en parcourant le graphe formé par les propriétés ontologiques de type relation qui établissent des liens entre les différentes instances de l'ontologie.

III.6.5 Diagramme d'états des comportements des agents

L'ensemble des comportements peut être divisé en 2 niveaux de comportement, comportements de niveau 1 et comportements de niveau 2.

III.6.5.1 Comportement de niveau 1 d'un agent AA-CRE

Le comportement de niveau 1 de l'agent (figure III.18) est obligatoire, il représente le traitement minimal qui doit être appliqué à une requête reçue. Lors de la réception d'une trame de l'environnement, l'agent doit récupérer l'identifiant du capteur qui lui est associé afin qu'il puisse accéder à l'ontologie et mettre à jour sa capacité. La requête *InitEffect*, informe l'agent du besoin de l'utilisateur. Initialement, les agents sont tous à l'état 0, à la réception des trames de l'environnement leur état ne change pas, la trame en question n'est que la transmission des données capturées, elle contient essentiellement l'identifiant de l'agent. En effet, cela permet par la suite que chaque agent ne reçoive que les trames transmises par le capteur qu'il encapsule.
Dès qu'une trame de type *InitEffect* est transmise, chaque agent bascule de l'état 0 vers l'état 1, sans pour autant arrêter la perception de l'environnement, qui reste possible grâce

au parallélisme des comportements. La figure III.18 illustre les deux premiers états avec l'algorithme 7 ci-dessous :

```
pour tout Reception(Requete, A) faire
  si Contenu requête == TrameEnvironnement alors
    si nombre trame == 1 alors
      Décomposition trame()
      Acces ontologie()
    sinon
      Sauvegarde de la trame
    finsi
  sinon
    Execute comportement InitCoal
  finsi
fin pour
```

Algorithme 7: Comportement de niveau 1 d'un agent AA-CRE

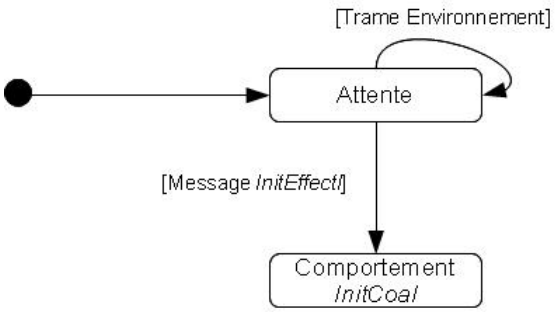

Figure III.18 : Diagramme d'état de transition de l'état 0 vers l'état 1

III.6.5.2 Comportements de niveau 2 d'un agent AA-CRE

Les comportements du second niveau (figure III.19) englobent le processus de formation de coalitions et la négociation. Pour la formation de coalitions, le premier comportement à exécuter est l'envoi de *InitCoal* suite à la réception d'un *InitEffect*. L'exécution d'un *InitCoal* consiste à envoyer un message au voisinage de l'agent contenant principalement la capacité de l'agent. Tous les agents qui reçoivent ce message acceptent ou refusent de faire partie de cette coalition. Les agent qui acceptent doivent alors exécuter le comportement *AcceptCoal* puis envoyer un message d'acceptation. Les initiateurs qui reçoivent une acceptation répondent par une confirmation.

Les algorithmes 8 et 9 ci-dessous montrent les comportements au niveau 2 ; spécialement, les comportements de la formation de coalitions en complément du diagramme de

transition précédent.

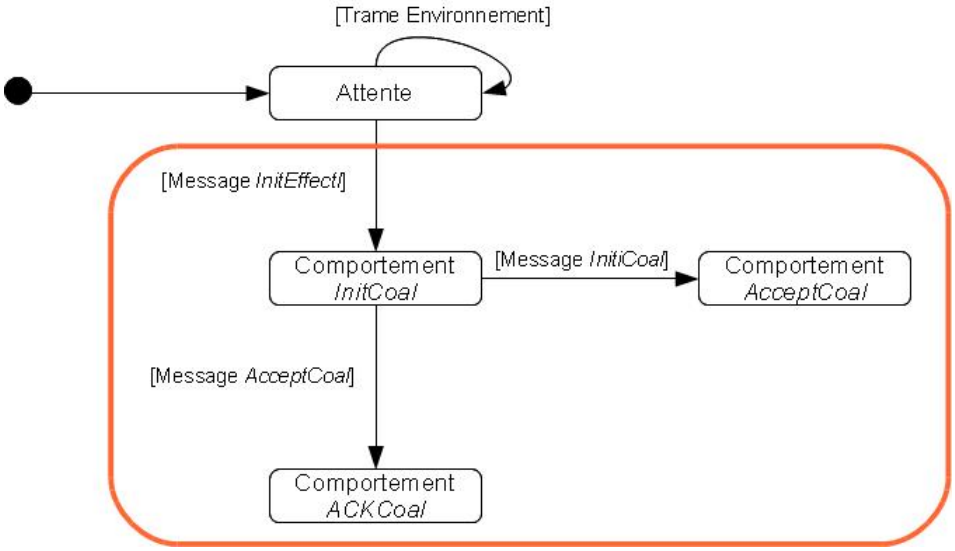

FIGURE III.19 : Diagramme d'état de transition vers état 1

Recherche voisinage(A_i)
Envoi(InitCoal, voisinage)
Attente()
si Reception(AcceptCoal, $voisin_j$) **alors**
  Executer comportement Envoi ACKCoal
  si Reception(InitCoal, $Initiateur_k$) **alors**
    Executer comportement Envoi AcceptCoal
  finsi
finsi

Algorithme 8: Comportement Envoi/Reception InitCoal

si Message recu = InitCoal **alors**
  Envoi(AcceptCoal), Emetteur
finsi

Algorithme 9: Comportement Envoi AccepCoal

si Message recu = AcceptCoal **alors**
  Envoi(ACKCoal), Emetteur
finsi

Algorithme 10: Comportement Envoi ACKCoal

Pour la négociation (figure III.20), une fois que chaque initiateur a formé sa meilleure coalition, une seule est gagnante. Pour cela, les agents entrent dans le processus de négociation qui englobe, entre autres, le comportement *InitNegociation* permettant d'initier une négociation en vue de sélectionner la coalition avec la meilleure capacité (algorithme 11).

```
si formation coalition == OK alors
  Envoi(C_coal, initiateur)
  si Réception(InitNegociaition, Initiateur) alors
    si capacité coalition > capacité coalition envoyée alors
      Emetteur ← Perdu
      si C_Initiateur > C_i alors
        Envoi déploiement aux membres de la coalition
      sinon
        continue
      finsi
    sinon
      si capacité coalition < capacité coalition envoyée alors
        Récepteur ← Perdu
        Executer comportement EndNegociation,
      sinon
        Choix()
      finsi
    finsi
  finsi
finsi
```

Algorithme 11: Comportement InitNegociation à l'État 3 :

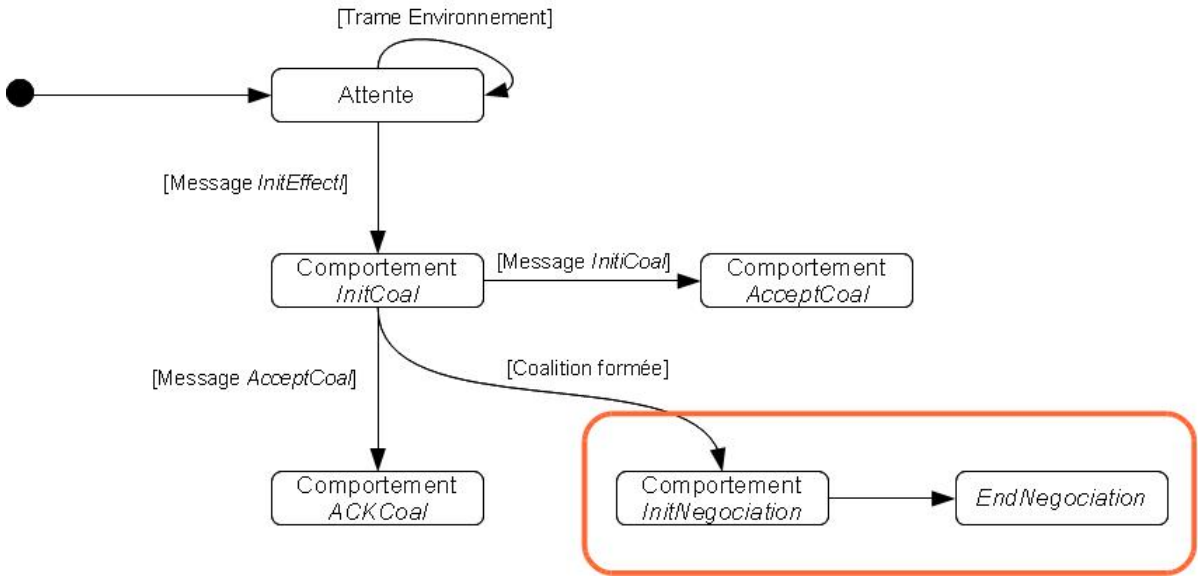

FIGURE III.20 : Diagramme d'état de transition des états 1-3

Le comportement *EndNegociation* est un comportement qui s'execute une fois q'une coalition gagnante a émergé. C'est un comportement qui permet le déploiement de la coalition.

Pour illuster le fonctionnement global du système de Coopération Robot-Environnement. Nous allons décrire ci-dessous les détails des interactions entre les agents AA-CRE. Considérant que dans le scénario Localisation, le système est composé de détecteurs de présence, une caméra fixe, une caméra mobile du robot et des marqueurs qui peuvent être identifiés par les caméras on aura donc les types d'agents suivants :

- Un $AgentDetecteurPresence$ de type détecteurs de présences.
- Un $AgentCameraFixe$ de type caméra fixe.
- Un $AgentCameraMobile$ de type caméra mobile.
- Un $AgentInterface$.

La représentation graphique des interactions entre les agents et l'utilisateur éventuel du système est illustrée dans le diagramme des séquences suivant :

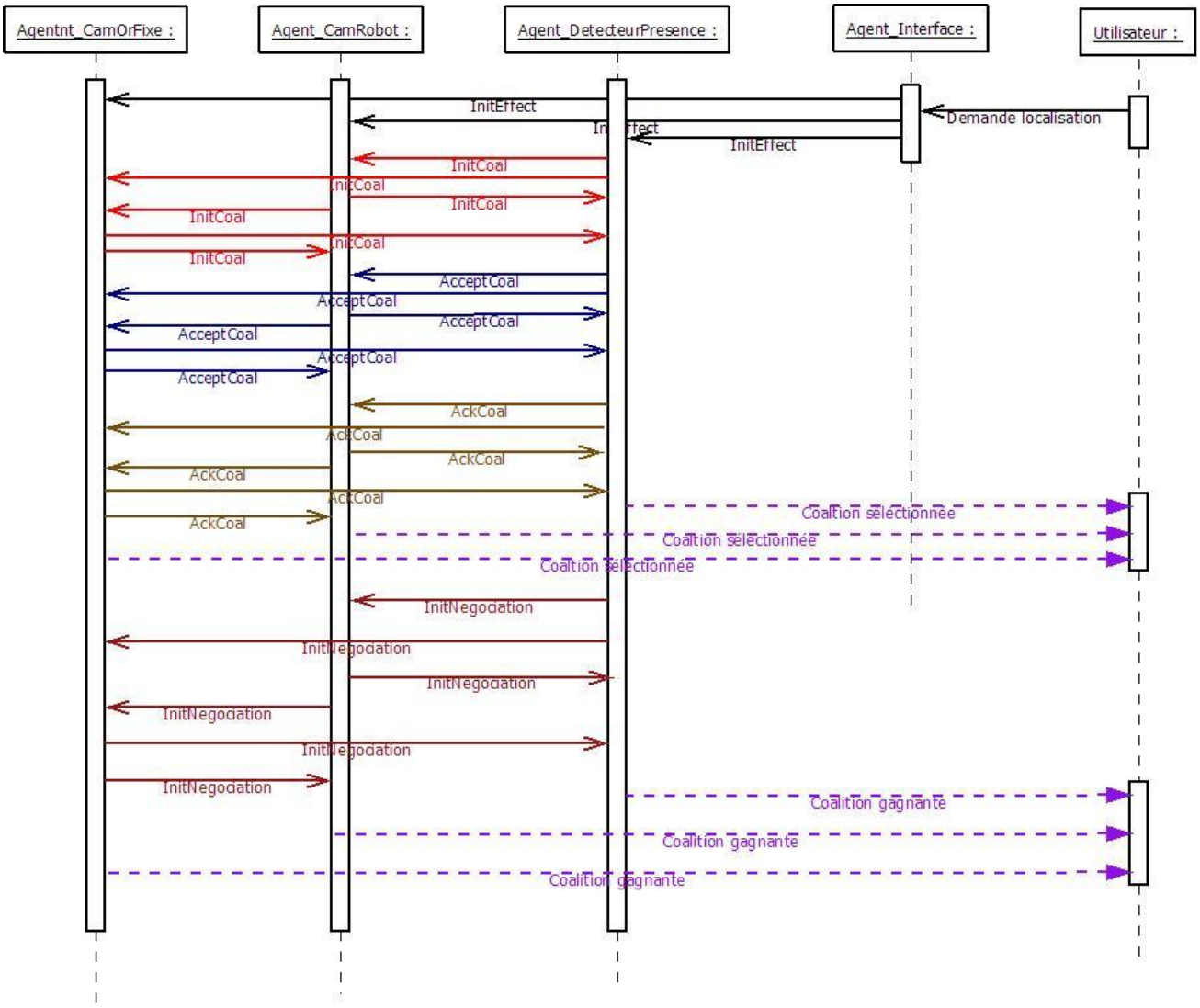

Figure III.21 : Diagramme des séquences

Le diagramme de séquence sur la figure III.21 ci-dessus décrit l'échange de messages entre les différents agents pour la formation d'une coalition gagnante.

1- Tout d'abord les agents à leur création s'initialisent en fonction de l'état de l'environnement en utilisant les deux primitives du comportement de perception de l'environnement :

1. Chaque agent reçoit un premier trame de données venant des capteurs physiques à travers la Passerelle. Comme chaque trame contient l'identifiant du capteur et ses mesures, chaque agent initialise son identifiant et fait un premier sauvegarde des mesures qu'il reçoit, dans son historique.
2. Puis chaque agent fait une recherche dans la base de connaissances en recherchant, le sous-concept correspondant à son type et à l'individu à qui il correspond grâce

à l'identifiant qu'il a acquis de la passerelle. Ce sous-concept et cet individu appartiennent au concept Objet Communicant.

3. A travers ce sous concept, sont initialisées plusieurs attributs physiques de l'agent. Par exemple, sa précision, sa portée, ses localisations dans une pièce. L'appartenance à une pièce précise étant connue grâce à une inférence par une transitivité. Par exemple pour un Agent de type Détecteur de Présence, on aura la transitivité suivante : AgentDetecteurPresence[*Appartient A*(AgentDetecteurPresence, mur 21) et *Appartient A*(Mur 21, Pièce 1)] implique *Appartient A*(AgentDetecteurPresence, pièce 1).

2- Une fois que chaque agent est initialisé, ils connaissent tous leur propre capacité C_i qui sont la combinaison de tous les attributs qu'ils ont acquis lors de l'écoute des trames et de la consultation de l'ontologie. Ils sont donc dans l'État 0 qui est l'état de Veille des agents. Ils sont prêts à recevoir une requête.

3- Une requête *InitEffect* (messages en noir) a été envoyée à tous les agents. Le message contient une requête de LOCALISATION du robot avec une précision P soit E(Localisation, P).

4- S'apercevant que chacun des agents tout seul ne peut résoudre la requête à cause de leur capacité limité. Les agents AgentDetecteurPresence, AgentCameraFixe et AgentCameraMobile initient chacun des formations de coalitions en envoyant un message *InitCoal* (messages en rouge) à leurs voisins immédiats ou topologiquement proche. Chaque message *InitCoal* contient, l'effet souhaité et la capacité de l'agent initiateur de coalitions *InitCoal*(E, C_i). Ici nous supposerons que tous les capteurs se trouvent tous dans une seule pièce, ce qui implique qu'ils sont tous voisins immédiats.

5- A l'issue des demandes de formations de coalition, chaque agent accepte ici de faire partie de chaque coalition, ils envoient un AcceptCoal (message en bleu). Ce message contient leur capacité *AcceptCoal*(C_i), en particulier leur précision.

6- Chaque agent initiateur confirme l'appartenance de chaque candidat par l'envoi d'une réponse *ACKCoal* (message marron). L'acceptation d'un agent candidat se fait par la vérification de la précision de cet agent par l'agent initiateur. Si la précision de l'agent est adéquate à celle de la requête. Ce message contient l'identifiant de la coalition et l'identifiant de l'agent référent A_i de la coalition notée $Coal_i$, $ACKCoal(A_i, Coal_i)$ à revoir pas d'identifiant de coalition. A ce stade, trois coalitions ont été formées avec trois initiateurs qui sont AgentDetecteurPresence, AgentCameraFixe, AgentCameraMobile. Chaque agent référent de coalitions calcule la capacité globale de sa coalition à chaque réception de message *AcceptCoal*. Comme il ne faut valider qu'une seule coalition, les agents vont entrer dans la phase suivante qui est la phase de négociation qui correspond à l'État 2

7- La phase de négociation est caractérisée par l'envoi de messages *InitNegociation* par chaque initiateur (message rouge). Ce message contient l'identifiant de la coalition et la capacité de la coalition.

8- Après cette séquence et qu'une coalition aura gagné, l'agent référent envoie un message de déploiement à ses candidats pour réaliser l'effet souhaité selon les capacités de chaque agent. Par contre, si aucune coalition n'est sortie gagnante, le SMA augmente le niveau de l'intrusion en bougeant la caméra mobile qui est celle du robot et le système se relance automatiquement. Donc, de nouvelles mesures sont acquises et la formation de coalitions recommence.
A la fin de la négociation, l'agent initiateur vérifie si le nombre de mesures est suffisant pour faire le calcul de localisation. Sinon, le système continue de tourner. Chaque initiateur reforme de nouvelles coalitions à partir de nouvelles mesures qu'ils auront acquis entre temps ou en ayant augmenté le niveau d'intrusion du système.

Ce scénario montre le fonctionnement d'un SMA pour la coopération robot-environnement, qui, grâce à un mécanisme de formation de coalitions est capable de façon adaptative, de répondre favorablement à une demande d'effet souhaité de l'utilisateur. L'aspect adaptatif est présent à deux niveaux, le premier point vis-à-vis des disponibilités données capteurs ainsi que leur précision et le second point par rapport au niveau d'intrusion qui s'ajuste en fonction de la gravité de la situation.

III.7 Conclusion

L'architecture de coopération robot-environnement proposée dans ce chapitre est constituée d'un SMA, d'un environnement ambiant et d'une interface utilisateur qui interagit avec le SMA. L'environnement ambiant regroupe l'environnement physique contenant des objets communicants, un robot, une passerelle et une base de connaissances.

La constitution d'une base de connaissances sous la forme d'une ontologie est une contribution importante pour le domaine de l'assistance ambiante. En effet, cela amorce la constitution d'un corpus de connaissances dédié au domaine et qui peut être, à terme, partagé et alimenté via le web de façon collaborative. De plus, de nombreux outils génériques pour l'interrogation et la mise à jour de tels entrepôts sont disponibles.

Dans le cadre de ce travail, l'ontologie a été employée pour représenter les informations persistantes qu'un habitat de type domotique possède ; telles que les caractéristiques des objets communicants, la topologie de l'habitat et les données associées à la personne.

Le niveau d'abstraction choisi est lié au besoin de l'application. Cependant, la puissance des ontologies se situe justement dans la représentation des concepts sous-forme hiérarchisée, permettant ainsi de faire évoluer la base de connaissances et d'augmenter le niveau de détails de la représentation sans remettre en cause l'existant. Par exemple,

l'ajout de données métriques permettraient d'améliorer la précision de la représentation topologique. De plus, le mode d'interrogation de l'ontologie par les agents peut être amélioré par l'emploi d'un moteur d'inférence en diversifiant le type de requêtes que les agents peuvent adresser à l'ontologie.

Pour répondre à la problématique de l'adaptation en fonction du contexte et du respect du niveau de gêne de l'utilisateur, la solution proposée est basée sur un mécanisme de formation de coalitions d'agents (nommés AA-CRE). Ces coalitions se constituent pour répondre à un besoin d'obtention d'un effet particulier dans le système (effet souhaité).

Chaque objet de l'environnement ambiant est encapsulé dans un agent AA-CRE. Rappelons que les objets embarqués sur le robot sont pris en compte de la même manière. Le protocole de Coopération Robot-Environnement devient alors un protocole de coopération entre les différents agents composant le SMA. L'un des intérêts des protocoles basés sur la formation de coalitions réside dans la souplesse avec laquelle les coalitions se forment. Ces dernières peuvent se défaire dynamiquement et se réorganiser avec des règles définies localement aux agents.
L'obligation de résultat et le respect d'un niveau d'intrusion en fonction de l'urgence de la situation, sont les critères les plus importants considérés au niveau des règles de constitution des coalitions. Ils sont aussi employés lors de la réorganisation des agents pour la recherche d'un effet souhaité. Le critère d'obligation de résultat est employé en priorité, tandis que le niveau d'intrusion n'est pris en compte (augmenté) que s'il est nécessaire d'acquérir de nouvelles données et donc d'actionner des capteurs (caméra par exemple) susceptible de provoquer une gêne pour la personne. Comme hypothèse de travail, nous avons considéré que les agents acceptent toujours (selon leur capacité) de participer à des coalitions. Ce choix étant fonction de leur voisinage immédiat, il serait pertinent d'améliorer ce critère en ajoutant des informations sur les affinités entre les agents. Par exemple, il serait possible de tenir compte de coalitions passées qui ont été satisfaisantes ou bien de mettre en place des stratégies de "couplage" d'agents en fonction du fait qu'il y ait une potentialisation ou non de leur association.

Le prochain chapitre illustre et montre la faisabilité de ce protocole sur des exemples de localisation d'un robot et d'éclairage d'une zone de l'environnement. Il exposera des résultats qui confirment la pertinence de tels protocoles pour les applications d'assistance ambiante.

Chapitre IV

Mise en œuvre et évaluations

Objectifs du chapitre

Ce chapitre décrit la mise en œuvre des différents composants de l'architecture de Coopération Robot-Environnement, les choix des implementations effectuées, l'évaluation du protocole multi-agent pour la formation des coalitions et enfin l'évaluation dans le contexte applicatif réel de la localisation du robot.
Comme vu dans le chapitre précédent, l'architecture de coopération robot-environnement est constitué des composants suivants (figure IV.1) :

- le SMA,
- la base de connaissances,
- la passerelle et
- l'environnement ambiant.

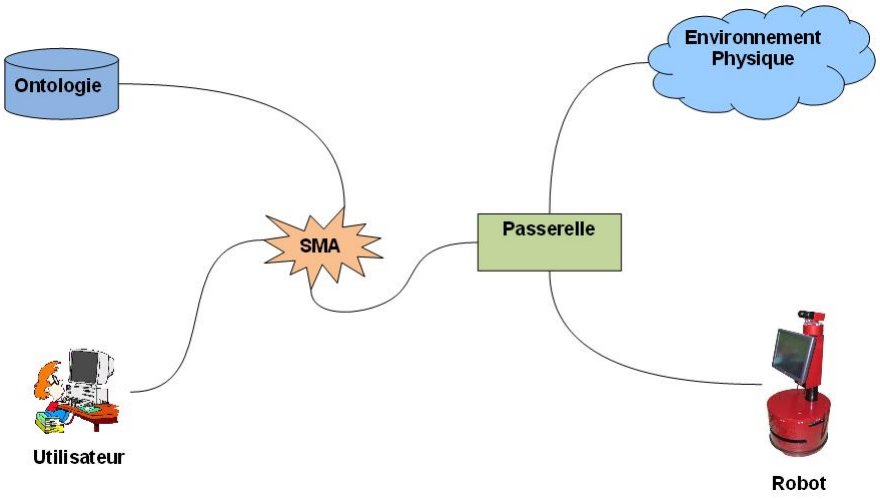

Figure IV.1 : Architecture globale du système

Pour l'implémentation du SMA, la plateforme retenue permet non seulement la communication asynchrone entre les agents mais aussi d'implanter plusieurs comportements au niveau des agents. L'accès des agents aux informations de l'ontologie est réalisé par l'intermédiaire d'outils permettant des interrogations de type "requête" qui offrent une souplesse dans les échanges. Il a été développé une passerelle (adaptateurs de protocoles) permettant de gérer l'hétérogénéité des communications entre les SMA et l'environnement ambiant. En ce qui concerne l'environnement ambiant, un algorithme de détection de marqueurs visuels de type Datamatrix soit à partir de caméras fixes, soit à partir de caméras orientables permet de compléter la gamme de capteurs de la plateforme expérimentale.

Le chapitre est organisé en trois parties. La première partie décrit la plateforme expérimentale. La deuxième partie, les réalisations tant au niveau du dispositif de détection des marqueurs et de la passerelle que du SMA et de l'ontologie. Et enfin la troisième partie illustre les évaluations du protocole de coopération agents d'une part, et d'autre part de l'ensemble de l'architecture pour la localisation du robot.

IV.1 Environnement ambiant

Afin de respecter les hypothèses de départ définies au premier chapitre, soit l'utilisation de réseaux de capteurs hétérogènes, la plateforme expérimentale intègre des capteurs de domotiques LEGRAND et des capteurs dédiés développés au laboratoire. L'environnement est composé d'une pièce de type habitat.

IV.1.1 Description de la plateforme expérimentale

L'environnement ambiant est équipé d'un ensemble de capteurs et du robot. Les capteurs présents dans l'environnement physique sont de trois types :

- des détecteurs de présence,
- des caméras fixées aux murs,
- une caméra orientable sur le robot,
- des marqueurs visuels.

Les dimensions de l'habitat sont 9.40 m x 6.34 m. La figure IV.2 indique la position des différents capteurs et marqueurs.

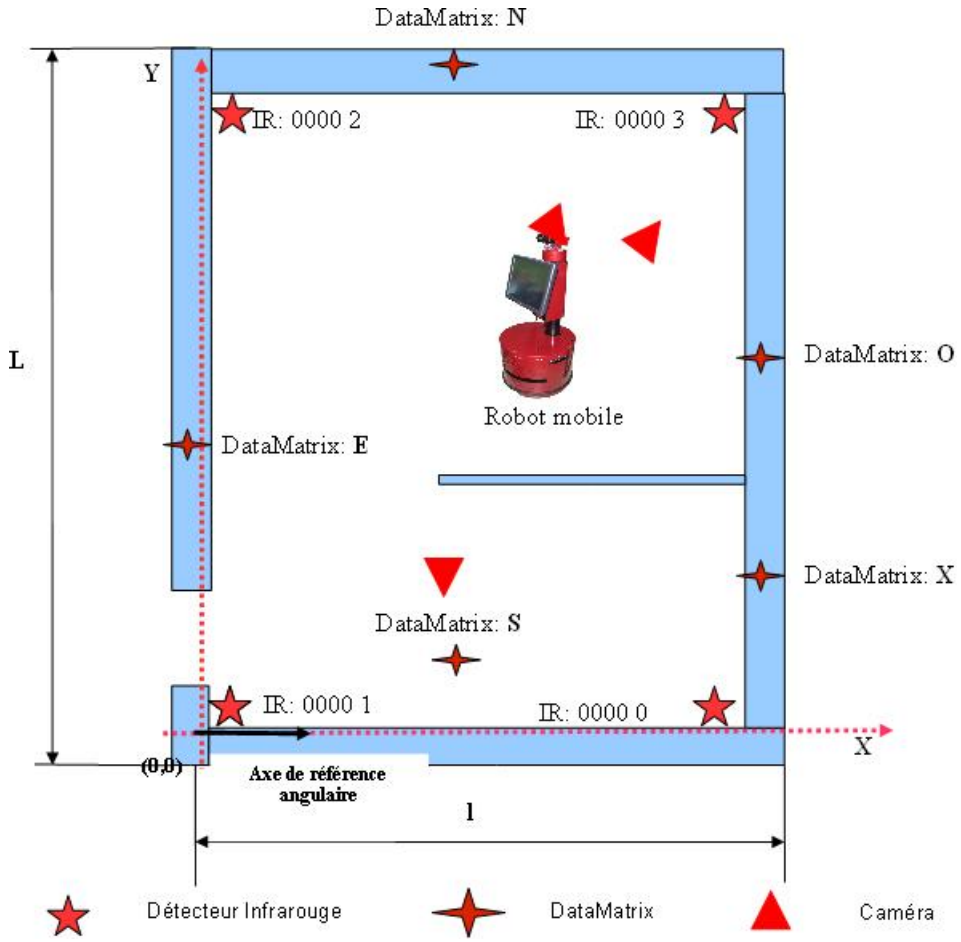

FIGURE IV.2 : Placement des capteurs et marqueurs et des repères

L'origine O du repère (x, y) de l'habitat est située en bas à gauche. L'orientation et l'angle mesuré de chaque capteur sont définis par rapport à l'axe des x. Les angles sont donnés dans le sens trigonométrique. Il est à remarquer que l'orientation de la caméra mobile est variable.

IV.1.2 Caractéristiques des capteurs

Le tableau IV.1 résume les caractéristiques de chaque capteur ainsi que leur position précise dans la pièce tel qu'ils sont décrits dans l'ontologie, dans le concept Objet Communicant. Les valeurs de l'orientation et de la position données sont des exemples pour chaque type de capteur.

Capteurs	Détecteurs de présence	Caméra fixe	Caméra orientable
Portée (m)	3	4	4
Précision (degré)	±45	±5	±5
Angle d'ouverture (degré)	90	±55/2	±55/2
Orientation (degré)	135	225	θ_{robot}
Position (x, y) (m)	(6.34, 0)	(3, 2.20)	(X_{robot}, Y_{robot})

Table IV.1 : Caractéristiques des capteurs

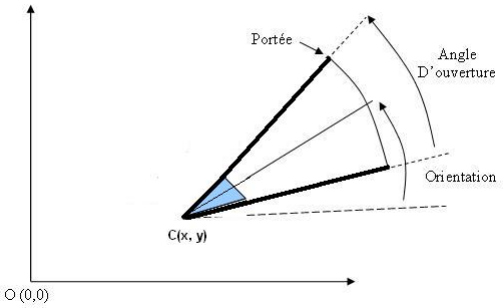

Figure IV.3 : Caractéristiques d'un capteur

IV.2 Réalisation : choix et état d'avancement

IV.2.1 Les marqueurs

Les marqueurs visuels sont des éléments destinés à être reconnus par des machines électroniques. Dans notre cas, les marqueurs sont détectés par des caméras fixes ou orientables, placées dans l'habitat ou montées sur le robot. Les marqueurs placés dans l'habitat sont détectés par la caméra du robot et ceux placés sur le robot sont détectés par les caméras de l'habitat. Nous avons choisi des marqueurs de type datamatrix. Le code datamatrix est une symbologie code-barre bidimensionnelle permettant de représenter une quantité importante d'informations sur une surface réduite. La datamatrix peut comprendre jusqu'à 2335 caractères alphanumériques ou 3116 caractères numériques. Sa détection utilise des méthodes de vision artificielle. L'intérêt du choix des datamatrix est la facilité de leur mise en œuvre et le fait qu'il existe actuellement plusieurs librairies de code disponibles pour leur utilisation. Les datamatrix peuvent se présenter sous forme carré ou rectangulaire (figure IV.4).

Figure IV.4 : Différents types de datamatrix

A l'exemple de la figure IV.5, chaque datamatrix de l'environnement identifie un élément de l'habitat. Le marqueur et son identifiant sont répertoriés dans l'ontologie et les datamatrix du robot ont le même identifiant.

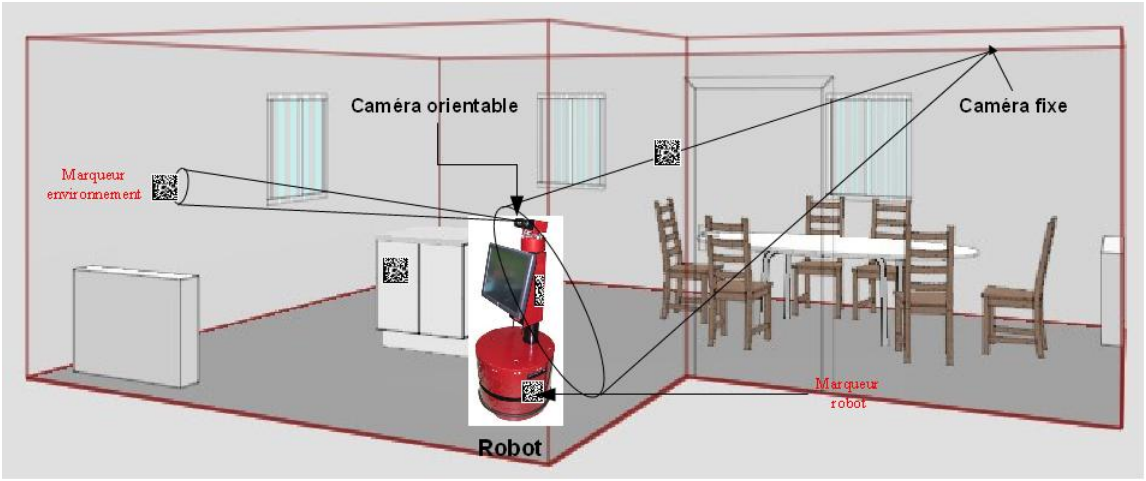

Figure IV.5 : Méthode de localisation multicapteur avec les datamatrix

IV.2.1.1 Détection de marqueurs avec la caméra mobile du robot

Dans le cadre du stage Master de Maxime Jubert, une application permettant de reconnaître les datamatrix a été développée. Le principe étant de récupérer le flux vidéo de la caméra présente sur le robot puis d'analyser les images de ce flux pour y localiser une datamatrix. Ensuite, la caméra s'oriente pour centrer la datamatrix sur son plan image. Enfin, la position angulaire de la caméra est mesurée, l'orientation de la caméra et la valeur de l'identifiant de la datamatrix sont transmises au SMA.
La figure IV.6 représente l'algorithme simplifié de détection de datamatrix par une caméra.

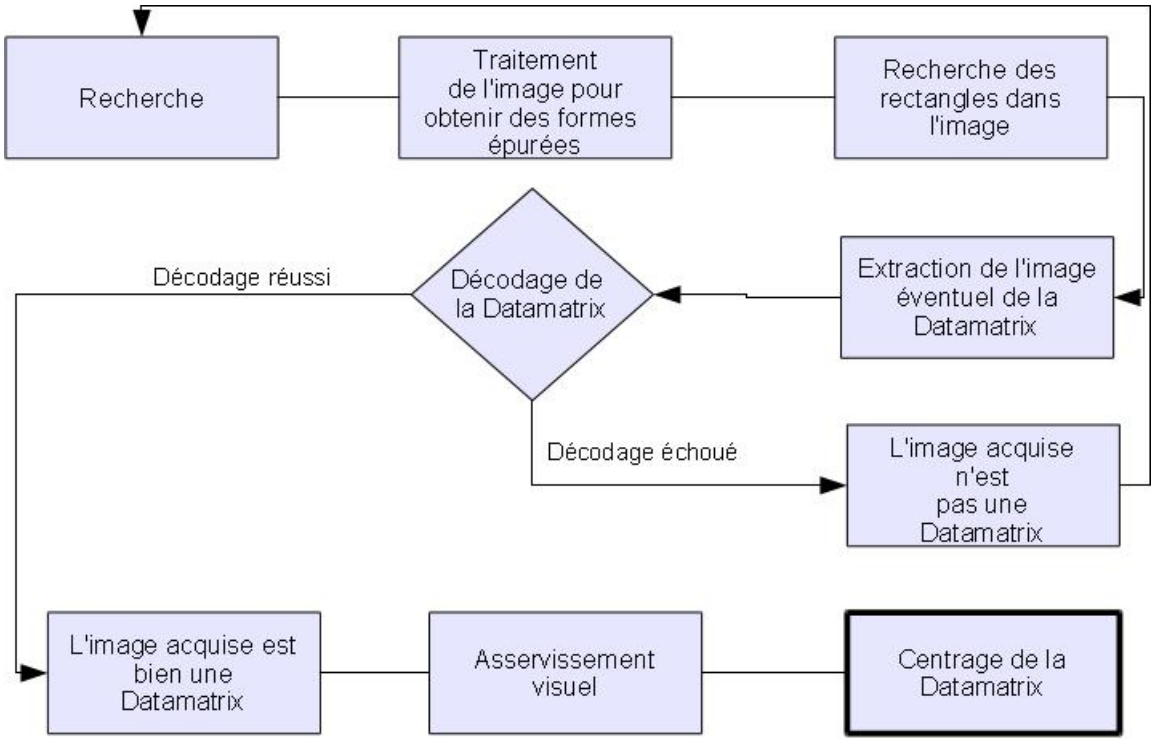

Figure IV.6 : Algorithme d'identification d'une Datamatrix

Pour l'évaluation, nous avons limité la recherche de marqueurs au plan horizontal passant par la caméra.

IV.2.1.2 Détection de marqueurs avec une caméra fixe

La détection de marqueurs par une caméra orientable fixe se base sur le même principe que la caméra mobile. C'est à dire, traitement de l'image acquise par la caméra, puis recherche des formes types datamatrix pour enfin décoder l'identifiant. La différence vient du fait que, comme la caméra est fixe, elle ne peut analyser que les marqueurs se trouvant dans son champ de vision. C'est pourquoi plusieurs datamatrix ont été disposées sur le pourtour du mat du robot afin que des caméras fixes de l'environnement puissent le détecter.

IV.2.2 Passerelle

La passerelle est un module destiné à adapter les protocoles d'échange des objets communicants de l'environnement ambiant avec celui du SMA. Cette adaptation est imposée par l'hétérogénéité des protocoles utilisés par les différents constructeurs de capteurs et d'actionneurs. Ainsi, le SMA perçoit et agit sur l'environnement ambiant au travers de la passerelle sans se soucier du format des données échangées. La passerelle assure les échanges entre les capteurs domotiques de type LEGRAND (protocole USB et proto-

cole NITOO), les caméras fixes (protocole TCP/IP filaire), la caméra mobile du robot (protocole TCP/IP, Wifi) et le SMA.

IV.2.2.1 Fonctionnement général de la passerelle

La passerelle est une interface entre les capteurs et le SMA, la figure IV.7 représente son algorithme général de fonctionnement. Son rôle est de recevoir les données des capteurs (via la passerelle NITOO pour les capteurs Legrand ou via un logiciel tiers pour les caméras fixes et la caméra mobile). Ces données sont ensuite traitées par notre passerelle pour être adaptées au format du SMA, puis stocker dans des fichiers XML et envoyer sous forme de requête aux AA-CRE.
La passerelle permet de recevoir des données via divers medium comme :

- le port Série,
- le port USB,
- le port TCP.

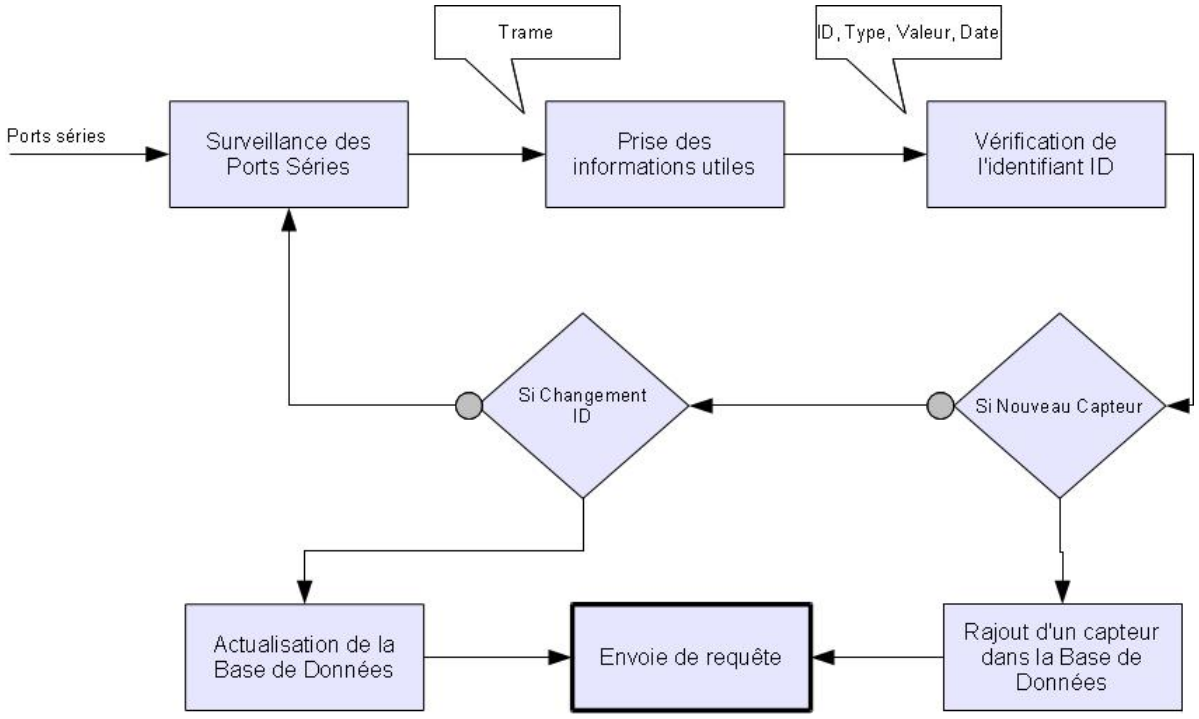

Figure IV.7 : Algorithme de fonctionnement de la passerelle

L'ensemble des trames arrivent par l'intermédiaire des ports séries. Une surveillance de trames par des "Listeners" est lancée en permanence et en parallèle. La passerelle dispose de trois listeners, un par type de périphériques (Port série, USB ou via requête TCP/IP). Ces listeners fonctionnent tous de manière analogue, ils espionnent le système pour détecter l'insertion ou le retrait d'un nouveau périphérique ou bien l'ouverture ou la fermeture d'un nouveau client et gère les données reçues. Dans un premier temps les

listeners espionnent les différents ports séries ou usb. Si un périphérique est connecté ou déconnecté le programme en est informé via l'usage des `WMI` [1].
Une fois que la connexion ou la déconnexion d'un périphérique est détectée, le listener questionne le système sur l'identité du périphérique et crée, dans le cas d'une connexion, un objet modélisant le périphérique (ou supprime cet objet dans le cas d'une déconnexion). Par exemple, connecter une clé USB à l'ordinateur créera un évènement Windows particulier qui sera détecté par le WMI. La passerelle créera alors une instance de l'objet "USB" qui s'occupera de récupérer les données émises de cette clé. Ensuite, lors de sa déconnexion, un évènement Windows particulier sera détecté par le WMI. La passerelle détruira alors l'instance de l'objet "USB" qui s'occupait de récupérer les données émises de la clé USB.

Pour la gestion des réceptions et émissions de trames, une primitive appelée TCPListener est lancée pour écouter un/des port(s) prédéfini(s) à l'avance. Cette primitive reçoit les données émises par le périphérique et lance leur traitement. Elle assure aussi la fermeture du flux d'information émanant d'un périphérique lors de son retrait. Le traitement des trames reçues se décompose en trois grandes fonctions :

- L'envoi de requête vers le SMA,
- le découpage des trames reçues pour en extraire les informations utiles,
- l'analyse des trames et la gestion de la base de données des capteurs.

Une fois la réception des informations utiles terminée, la passerelle envoie des requêtes TCP/IP contenant les informations reçues vers le SMA. Les requêtes envoyées et encapsulées dans des trames TCP se présentent toujours sous le même format suivant (figure IV.8) :

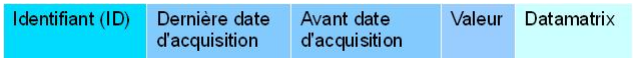

Figure IV.8 : Formats des requêtes envoyées aux AA-CRE

L'identifiant (ID) correspond à un identifiant qui est unique à chaque capteur et est généré automatiquement par la passerelle. Les dates d'acquisitions correspondent aux deux dernières dates auxquelles les capteurs ont acquis des mesures. Le champ valeur contient la mesure effectuée à la dernière date d'acquisition et enfin le dernier champ datamatrix représente la valeur de la datamatrix identifiée pour les marqueurs.

La base de données des capteurs est surtout utilisée pour permettre à la passerelle d'identifier et de gérer dynamiquement l'ajout de nouveaux capteurs dans l'environnement. Cette base de données des capteurs a été réalisée en XML pour son avantage d'être

1. WMI : Windows Management Instrumentation qui est un outil software permettant d'avoir connaissance des interactions de Windows avec le monde extérieur via des requêtes.

structuré et facilement manipulable par différentes plateformes. La figure IV.9 représente le format de chaque fichier XML selon le type de capteurs.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Capteurs>
     <DetecteurPresence id = "0001">
          <valeur> 0 </valeur>
          <date1> 11/05/2011 18:01:35:56 </date1>
          <date2> 11/05/2011 18:01:35:56 </date2>
     </DetecteurPresence>
<DetecteurPresence id = "0002" >
          <valeur> 1 </valeur>
          <date1> 11/05/2011 18:01:35:56 </date1>
          <date2> 11/05/2011 18:01:35:56 </date2>
     </DetecteurPresence>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Capteurs>
     <PassagePorte id = "1000">
          <valeur> 0 </valeur>
          <date1> 11/05/2011 18:01:35:56 </date1>
          <date2> 11/05/2011 18:01:35:56 </date2>
     </PassagePorte>
</Capteurs>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Capteurs>
     <CameraOrientableFixe id = "2001">
          <valeur> 0 </valeur>
          <date1> 11/05/2011 18:01:35:56 </date1>
          <date2> 11/05/2011 18:01:35:56 </date2>
          <rotation> 3,14 </rotation>
          <deplacementX> 0 </deplacementX>
          <deplacementY> 0 </deplacementY>

     </CameraOrientableFixe>
</Capteurs>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Capteurs>
     <CameraOrientableMobile id = "3001">
          <valeur> 0 </valeur>
          <date1> 11/05/2011 18:01:35:56 </date1>
          <date2> 11/05/2011 18:01:35:56 </date2>
          <rotation> 3,14 </rotation>
          <deplacementX> 1 </deplacementX>
          <deplacementY> -0,5 </deplacementY>

     </CameraOrientableMobile>
</Capteurs>
```

Figure IV.9 : Format des fichiers XML de la Base de données des capteurs

IV.2.2.2 Protocole propriétaire LEGRAND

Les détecteurs de présence déployés dans l'environnement physique sont de type LEGRAND. La gestion des différents capteurs et actionneurs de LEGRAND se fait spécifiquement par un protocole propriétaire qui est le protocole NITOO. Ce protocole définit un format spécifique des trames émises par les capteurs et les actionneurs de types LEGRAND.

Le dispositif mis en œuvre par LEGRAND pour le déploiement de leurs produits domotiques est un système à base de communication par radio-fréquence et `CPL`[2]. Le principe étant qu'il y a une interface radio qu'on appelle "*Passerelle LEGRAND*" qui assure la communication entre des produits à base de radio-fréquence et des produits comme des actionneurs qui utilisent le CPL. Cette passerelle LEGRAND reçoit les messages grâce à un appareil appelé "Interface Radio d'évolutivité" (figure IV.10) qui permet la communication des produits radio et courant porteur. C'est-à-dire, l'émission et la réception des trames. Un PC de contrôle est relié à la Passerelle Legrand par une liaison USB pour la gestion des appareils domotiques et l'espionnage des trames.

2. CPL : Le terme "Courants Porteurs en Ligne" (CPL) se réfère à une technique permettant le transfert d'informations numériques en passant par les lignes électriques.

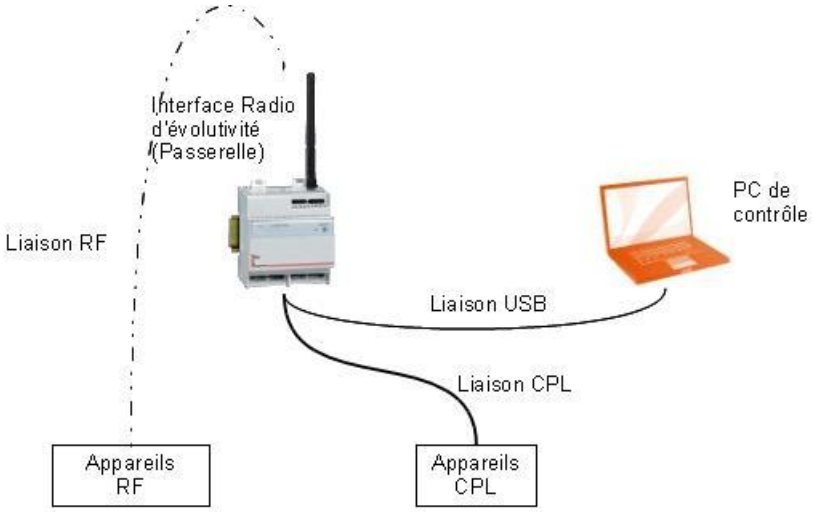

Figure IV.10 : Passerelle Legrand

IV.2.3 SMA et Ontologie

IV.2.3.1 SMA

IV.2.3.1.1 La plateforme JADE

JADE (Java Agent Development Framework) est une plateforme multi-agents développée en Java par CSELT (Groupe de recherche de Gruppo Telecom, Italie). Elle fournit un environnement de développement et d'exécution des systèmes multi-agents, compatible avec les standards FIPA [Huang2009]. JADE comprend deux composantes de base : la plateforme d'agents compatible FIPA et un package logiciel pour le développement des agents Java. Elle fournit les facilités suivantes :

1. Une plateforme agent répartie : la plateforme peut être distribuée (partagée) entre plusieurs hôtes (hosts) connectés via RMI de Java, de telle façon qu'une seule application Java, et par conséquent une seule " Machine Virtuelle Java " est exécutée sur chaque hôte.
2. Un certain nombre de DF (Facilitateurs d'Annuaire) compatibles FIPA : qui peuvent être activés quand on lance la plateforme pour exécuter les applications multi-domaines.
3. Une interface utilisateur graphique (GUI) : pour gérer plusieurs agents et conteneurs d'agents d'un hôte éloigné.
4. Des outils de Déboguage : pour faciliter la mise au point d'applications.
5. Un transport efficace des messages entre les agents : à l'intérieur de la même plateforme. Cette conversion est transparente aux programmeurs intéressés uniquement aux objets Java.

6. Une bibliothèque de protocoles : compatibles aux standards FIPA et prêts à être employés pour régir l'interaction inter-agent.

IV.2.3.1.2 Utilisation de JADE pour le développement de l'architecture proposée

JADE utilise l'abstraction de comportement pour modéliser les tâches qu'un agent peut exécuter. De point de vue de la programmation concurrente, un agent est un objet actif, ayant un thread de contrôle. JADE utilise un modèle de programmation concurrente "un thread par agent" pour éviter une augmentation du nombre de threads d'exécution exigés sur la plateforme d'agents. Ceci signifie que, pendant que les agents différents s'exécutent dans un environnement multi-threads préemptive. Ainsi, deux comportements d'un même agent sont planifiés d'une façon coopérative.

Du point de vue implantation, un agent dans JADE est une instance de la classe Java définie par le programmeur. Cette classe elle-même est une extension de la classe Agent de base (incluse dans une librairie jade.core de Jade). Cela implique l'héritage de l'ensemble de méthodes de base pour implémenter le comportement personnalisé de l'agent. L'implémentation d'un agent est multitâche, où les tâches (appelées comportements) sont exécutées en concurrence. Chaque fonctionnalité fournie par un agent doit être implémentée en un ou plusieurs comportements.

Le tableau IV.2 dresse la liste des trois différents types d'agents qui ont été développés suivant les types de capteurs qui se trouvent dans l'environnement.

Capteur	**Agent AA-CRE**
Détecteur de présence (Capteur infrarouge)	$AgentDetecteurPresence$
Caméra fixe	$AgentCameraFixe$
Caméra mobile	$AgentCameraMobile$

Table IV.2 : Différents agents AA-CRE

La figure IV.11 représente le diagramme de classe de l'implémentation des différents agents AA-CRE. Les trois types agents héritent d'un agent générique PrototypeAgentAA qui, lui même est l'extension de la classe Agent de Jade.

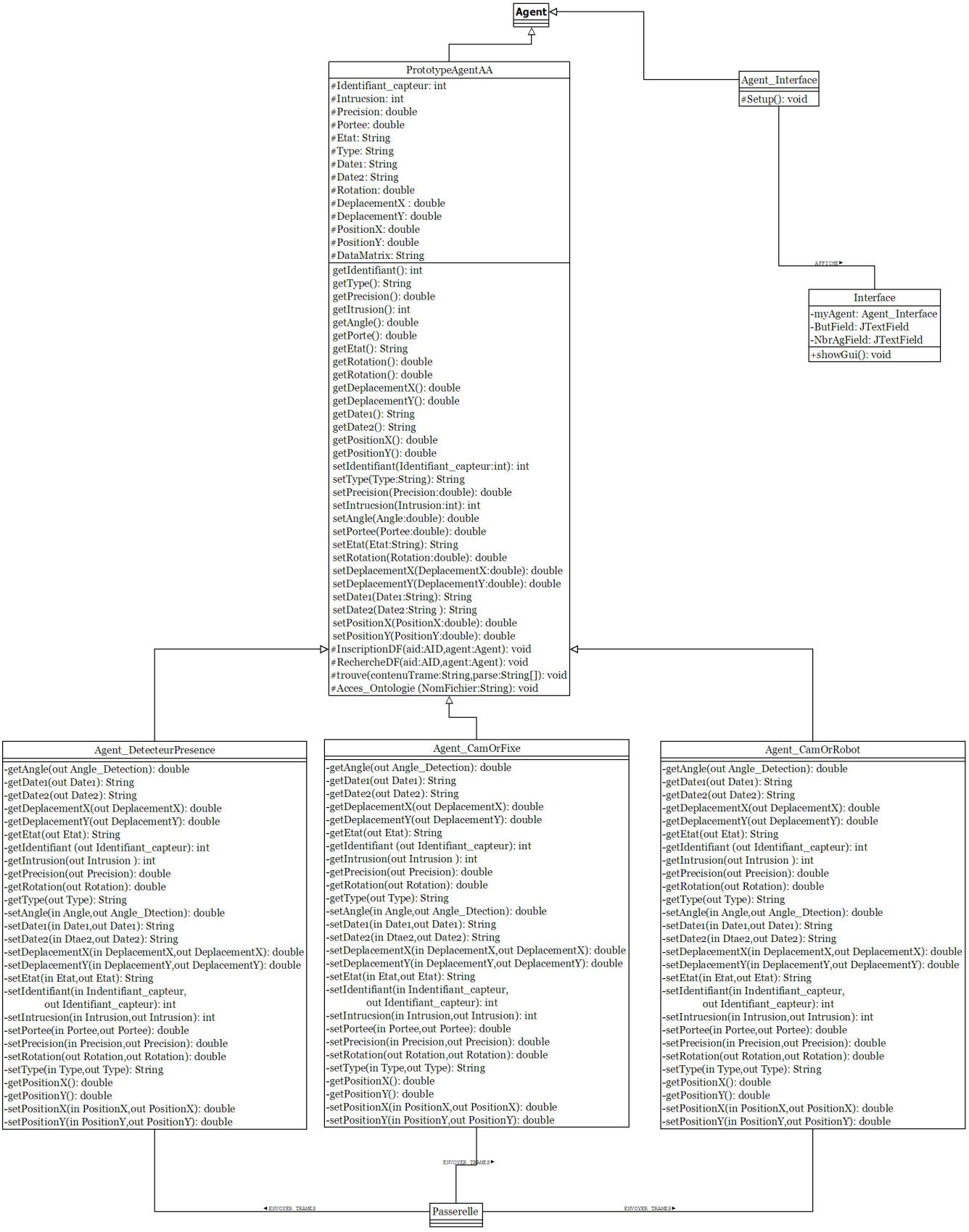

Figure IV.11 : Diagramme de classes des agents AA-CRE

IV.2.3.1.3 Comportements des agents

JADE offre trois types de comportements qui sont :

– les Behaviour Simples,

– les Behaviour Planifiés

– les Behaviour Composés

Chaque type de comportement est constitué de deux ou trois sous-comportements différents (figure IV.12). Les différents comportements des agents implantés pour la formation de coalitions sont de type Behavior Simple.

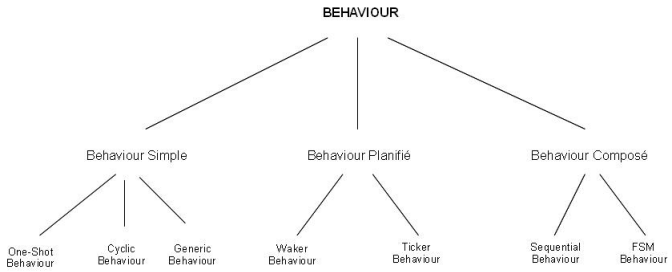

Figure IV.12 : Différents comportements dans JADE

La figure IV.13 représente les sous comportements les plus courants dans JADE. Le comportement One Shot Behaviour a la particularité d'exécuter sa tâche une et une seule fois puis il se termine. La classe OneShotBehaviour implémente une méthode classique *done()* dans laquelle sont exécutés les différents traitements inhérents au comportement. La classe CyclicBehaviour implémente la méthode *done()* et exécute sa tâche d'une manière répétitive. Le WakerBehaviour est implémenté de façon à exécuter une méthode *onWake()* après une période passée comme argument au constructeur. Enfin, le TickerBehaviour est implémenté pour qu'il exécute sa tâche périodiquement par une méthode nommée *onTick()*.

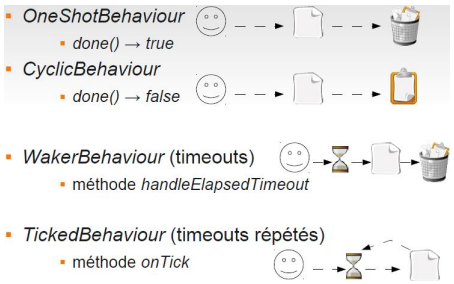

Figure IV.13 : Sous comportements les plus courants dans JADE

Le tableau IV.3 cite les comportements de chaque agent AA-CRE vu dans le chapitre Architecture Informatique avec leur type de comportement correspondant développé sur JADE.

Comportement	Type de comportement JADE
SendInitCoalBehaviour	OneShotBehaviour
SendInitNegociationBehaviour	OneShotBehaviour
SendAckCoalBehaviour	OneShotBehaviour
SendAcceptCoalBehaviour	OneShotBehaviour
ReceptionInitNegociationBehaviour	CycliBehaviour
ReceptionAckCoalBehaviour	CycliBehaviour
ReceptionInitCoalBehaviour	CycliBehaviour
ReceptionAcceptCoalBehaviour	CycliBehaviour
PerceptEnvironmentBehaviour	CycliBehaviour
ReceptionInitEffectBehaviour	CycliBehaviour

Table IV.3 : Types de comportements dans chaque agent AA-CRE

Ci-dessous est représentée une spécification partielle des comportements d'un agent AA-CRE. Les comportements pour les envoies de messages sont de type OneShotBehaviour qui ne s'enclenchent qu'une seule fois pour assurer l'envoi unique d'un message de type requête tandis que, pour assurer la réception permanente des messages destinés aux agents, les comportements pour les réceptions de messages qui conditionnent l'évolution des coopérations entre chaque agent sont de types CyclicBehaviour.

```
private class ReceptionAcceptCoalBehaviour extends CyclicBehaviour {
    public void action() {
        /*** Traitement **/
        // Message AcceptCoal à // réceptionner
        ACLMessage AcceptCoal = receive(MessageTemplate.MatchPerformative(
        ACLMessage.INFORM_REF));
        /*** Traitement **/
        } );
    }
}

private class SendInitCoalBehaviour extends OneShotBehaviour {
    public void action() {
            /*** Traitement **/
            Initcoal.addReceiver(agents[i]); // Préparation du message InitCoal à envoyer
            }
        send(Initcoal);
        }
    }
```

IV.2.3.2 Ontologie pour l'assistance ambiante

La modélisation d'un domaine sous forme d'ontologie nécessite un langage pour spécifier les concepts du domaine. Il existe de nombreux langages de représentation d'ontologies (XOL [Horrocks2002], OIL [Fensel2001], DAML-OIL [Horrocks2002]). Actuellement, les langages du Web Sémantique [Berners-Lee2001] s'imposent comme la référence pour la modélisation de connaissance dans les environnements ubiquitaires [Wang2004] [Ranganathan2003] [Chen2004]. La figure IV.14 présente un modèle en couche qui montre le lien entre les différentes plate-formes logicielles. RDF [Lassila2002] permet une représentation des ressources et de leurs relations. RDF Schema [Brickley2004] permet de définir des concepts et les relations entre ces concepts sous forme de classes et de propriétés. Finalement OWL, constitue une extension sémantique à RDF Schema offrant un vocabulaire plus riche pour la description des classes et des propriétés. Nous nous intéressons particulièrement au langage OWL qui est le langage de référence pour la définition d'ontologies pour le Web Sémantique.

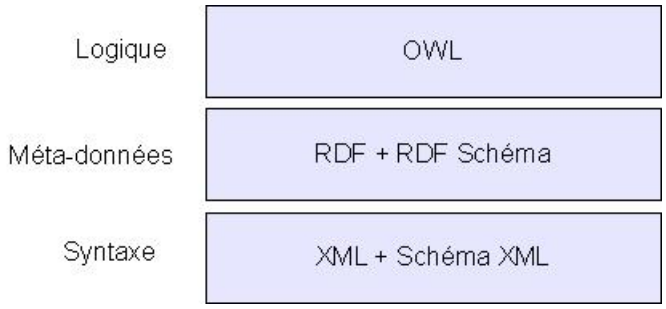

Figure IV.14 : Représentation schématique des caractéristiques de xml, rdf et owl

RDF, OWL et XML constituent les trois couches de base du Web Sémantique (figure IV.14) : XML est le support de sérialisation sur lequel s'appuient RDF et OWL pour définir les structures de données et les relations logiques qui les lient [Lapique2006].

IV.2.3.2.1 Éditeur d'ontologies Protégé

Protégé [Stanford2005] est un éditeur d'ontologies distribué en open source de l'université en informatique médicale de Stanford. Nous avons choisi d'utiliser Protégé pour définir la base de données ontologique car c'est un outil puissant et flexible spécialement dédié à OWL. En effet, il permet de formaliser les concepts primitifs en utilisant les logiques de descriptions. Et enfin c'est aussi un éditeur hautement extensible, capable de manipuler des formats très divers. Le support d'OWL, comme de nombreux autres formats, est possible dans Protégé.

OWL propose trois types de propriétés :

- Les propriétés qui relient des instances, définies sous le terme de Object Property dans Protégé.
- Les propriétés qui relient une instance avec une valeur, définies sous le terme de ***Datatype Property*** dans Protege.
- Les annotations qui permettent d'associer/décrire des méta-données aux classes, propriétés et instances.

Le paragraphe suivant décrit les instances de l'ontologie, appelées "ăindividusă", de la plateforme expérimentale.

IV.2.3.2.2 Description des sous ontologies

Habitat L'habitat est formé d'une grande pièce séparée partiellement par une cloison (figure IV.15).

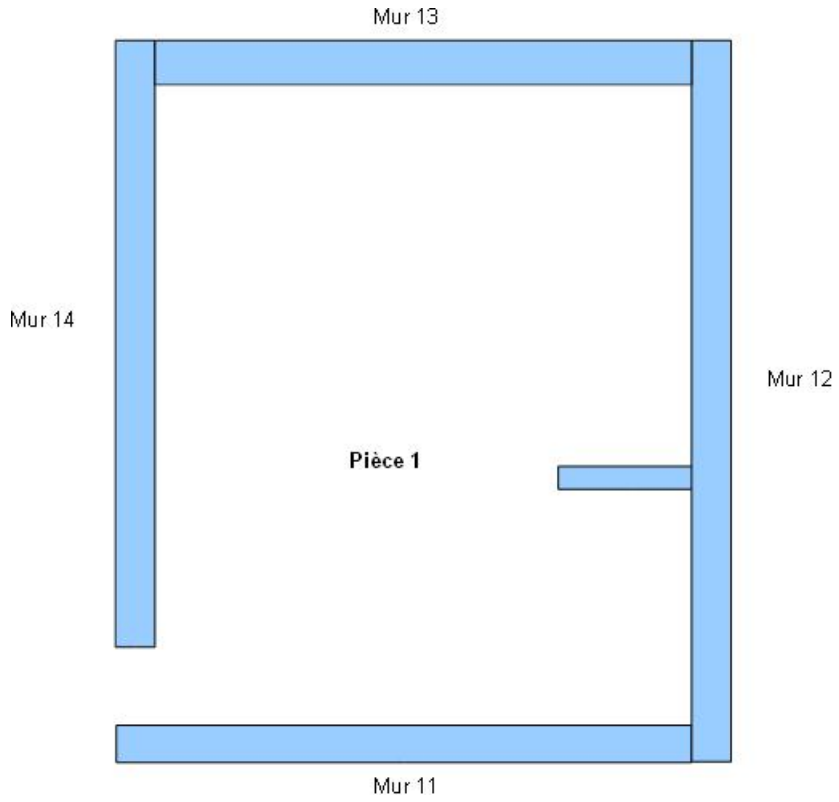

Figure IV.15 : Habitat

Comme vu dans le chapitre Architecture Informatique (Cf. chapitre III), les murs qui appartiennent à une pièce sont liés par une relation d'appartenance. Cette relation est une propriété de type Object Property qui représente le lien entre deux individus. La figure IV.16 représente un exemple partiel du code relatif à la relation d'appartenance entre un mur et la pièce de l'habitat. C'est un code owl généré automatiquement par Protégé.

```
<owl:Thing rdf:ID="Mur14">
          <OntologyAA:AppartientA_Mur-Piece rdf:resource="#Piece1"/>
          <rdf:type rdf:resource="#Mur"/>
             <owl:sameAs rdf:resource="#Mur13"/>
          <owl:sameAs rdf:resource="#Mur12"/>
          <owl:sameAs rdf:resource="#Mur13"/>
</owl:Thing>
```

Figure IV.16 : Représentation owl d'une instance d'un mur de l'habitat

Capteurs et marqueurs visuels Les figures IV.17 et IV.18 ci-dessous représentent respectivement des parties de code owl des trois types de capteurs qu'on trouve dans l'environnement ainsi que les marqueurs visuels de type Datamatrix. Les caractéristiques des capteurs sont représentées par des relations de type attribut.

```
<OntologyAA:DetecteurPresence rdf:ID="DetecteurPresence_4">
   <OntologyAA:PorteeMaximale rdf:datatype="http://www.w3.org/2001/XMLSchema#float'
   >2.0</OntologyAA:PorteeMaximale>
   <OntologyAA:PositionX rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
   >6.34</OntologyAA:PositionX>
   <OntologyAA:PositionY rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
   >9.4</OntologyAA:PositionY>
   <OntologyAA:PlagePrecision rdf:datatype="http://www.w3.org/2001/XMLSchema#float'
   >45.0</OntologyAA:PlagePrecision>
   <OntologyAA:AngleOuverture rdf:datatype="http://www.w3.org/2001/XMLSchema#float'
   >90.0</OntologyAA:AngleOuverture>
   <OntologyAA:Orientation rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
   >225.0</OntologyAA:Orientation>
   <OntologyAA:Identifiant rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >0000 3</OntologyAA:Identifiant>
</OntologyAA:DetecteurPresence>
```

Figure IV.17 : Représentation owl d'une instance des capteurs de type détecteur de présence

```
<OntologyAA:CameraFixe rdf:ID="CameraFixe_1">
   <OntologyAA:PositionY rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
   >7.0</OntologyAA:PositionY>
   <OntologyAA:PositionX rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
   >4.64</OntologyAA:PositionX>
   <OntologyAA:PlagePrecision rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
   >5.0</OntologyAA:PlagePrecision>
   <OntologyAA:Orientation rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
   >225.0</OntologyAA:Orientation>
   <OntologyAA:Identifiant rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >3000 1</OntologyAA:Identifiant>
   <OntologyAA:AngleOuverture rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
   >27.5</OntologyAA:AngleOuverture>
</OntologyAA:CameraFixe>

<OntologyAA:CameraRobot rdf:ID="CameraRobot_1">
   <OntologyAA:Identifiant rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >2000 1</OntologyAA:Identifiant>
   <OntologyAA:AngleOuverture rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
   >27.5</OntologyAA:AngleOuverture>
   <OntologyAA:PlagePrecision rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
   >5.0</OntologyAA:PlagePrecision>
</OntologyAA:CameraRobot>
```

Figure IV.18 : Représentation owl d'une instance des capteurs de type caméra fixe et caméra orientable du robot

```
<OntologyAA:MarqueurVisuel rdf:ID="Datamatrix_1">
    <OntologyAA:Identifiant rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >N</OntologyAA:Identifiant>
    <OntologyAA:PositionX rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >3.17</OntologyAA:PositionX>
    <OntologyAA:PositionY rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >9.4</OntologyAA:PositionY>
 </OntologyAA:MarqueurVisuel>
```

Figure IV.19 : Représentation owl d'une instance d'une datamatrix

Appartenance d'un capteur et d'un marqueur visuel à un mur d'une pièce L'appartenance d'un capteur ou d'un marqueur visuel à un mur de la pièce se fait aussi par des propriétés de type Object Property. La figure IV.20 donne une vue partielle du code d'une caméra fixe qui appartient au Mur12 de l'habitat. Il est à remarquer qu'en réalité, chaque caméra fixe est disposée sur un trépied par soucis de commodité au lieu de les fixer directement au mur (figure IV.21). Cependant, la relation d'appartenance topologique n'est prise en compte que quand on raisonne sur un habitat avec au minimum deux pièces.

```
<OntologyAA:MarqueurVisuel rdf:ID="Datamatrix_1">
   <OntologyAA:AppartientA_Capteur-Mur rdf:resource="#Mur13"/>
   <OntologyAA:Identifiant rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   ></OntologyAA:Identifiant>
   <OntologyAA:Identifiant rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >N</OntologyAA:Identifiant>
   <OntologyAA:PositionX rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
   >3.17</OntologyAA:PositionX>
   <OntologyAA:PositionY rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
   >9.4</OntologyAA:PositionY>
 </OntologyAA:MarqueurVisuel>
```

Figure IV.20 : Représentation owl d'd'un marqueur visuel et son appartenance à un mur

```
<OntologyAA:CameraFixe rdf:ID="CameraFixe_1">
        ........
<OntologyAA:AppartientA_Capteur-Mur>
      <OntologyAA:Mur rdf:ID="Mur12">
        ........
</OntologyAA:AppartientA_Mur-Piece>
           <rdf:type rdf:resource="#Mur"/>
         </owl:Thing>
```

Figure IV.21 : Représentation owl d'un capteur et son appartenance à un mur

IV.2.3.3 Interrogation de l'ontologie par le SMA

En l'état actuel, les échanges entre le SMA et l'ontologie sont en partie simulées. Le principe étant qu'un comportement spécifique de l'agent assure l'interrogation de l'ontologie par des requêtes de type `SPARQL`[3]. C'est une solution qui a été implantée pour permettre d'évaluer l'ensemble de l'architecture mais pourrait être améliorée. La figure

3. SPARQL : SPARQL Protocol and RDF Query Language est un langage de requête et un protocole qui permet de rechercher, d'ajouter, de modifier ou de supprimer des données RDF disponible.

IV.22 représente une partie du code de ce comportement de type OneShotBehaviour et qui est exécuté à l'initialisation de l'agent.

```
class BehaviourInterrogationOntologyAA extends OneShotBehaviour {

    private String resourceType  = null;
    private Agent a;

    public BehaviourInterrogationOntologyAA(Agent _a, String _resourceType) {
      super(_a);
      a = _a;
      resourceType = _resourceType;
    }

    public void action() {
        agentIndividual.removeAll(mem.createProperty("resource"));
                ...............
        send(Message.createQueryMessage(a,"AgentCapteur", "SELECT ?x WHERE {?x rdf:type ont:"+resourceType+"}"));
    }
```

Figure IV.22 : Représentation partielle du comportement de l'interrogation de l'ontologie

IV.3 Évaluations

IV.3.1 SMA

Afin de valider notre protocole de coopération d'agents, nous l'avons comparé à un autre protocole connu qui est le Contract Net. C'est un travail qui a été réalisé dans le cadre du stage de Master de M. Lyes SEFIANE. Nous décrivons dans un premier temps le protocole Contract Net qu'on a implémenté dans le même contexte de problématique que la formation de coalitions, puis nous présentons une comparaison expérimentale des deux protocoles.

IV.3.1.1 Le Contract Net Protocol

Le protocole Réseau Contractuel (Contract Net en anglais) a été la première approche utilisée dans les SMA pour résoudre les problèmes d'allocation de tâches. Proposé par Smith en 1980 [Smith1980], il est basé sur une métaphore organisationnelle, c'est à dire que les agents coordonnent leur travail en se basant sur l'établissement de contrats. Dans ce protocole on distingue deux types d'agents, un agent gestionnaire et un agent contractant. L'agent qui doit exécuter une tâche est le gestionnaire. Il décompose chaque tâche en plusieurs sous tâches possibles, puis il annonce chaque sous tâche à un réseau d'agents qui sont les contractants, en envoyant une proposition. Les agents contractants qui ont les ressources appropriées pour accomplir la tâche répondent au gestionnaire en lui envoyant leur soumission avec leur capacité à résoudre le problème. Le gestionnaire analyse toutes les soumissions reçues puis en fonction du résultat de cette analyse attribue la tâche au meilleur contractant.

IV.3.1.2 Formation de coalitions à base de Contract Net Protocol

Lorsqu'un gestionnaire propose une tache à résoudre, un contractant peut accepter ou refuser la proposition. Dans le cas d'un refus, une soumission est envoyée au gestionnaire. Dans le cas contraire, il envoie une demande de formation de coalition à son voisinage, l'algorithme de formation de coalitions est le même que celui implémenté pour la coopération robot-environnement. Une fois que tous les contractants ont formé leur propre coalition, ils retournent la capacité de chacune de leur dernière coalition au gestionnaire.

IV.3.1.3 Expérimentations et résultats

Les expérimentations et résultats concernant le protocole de formations de coalitions présentés dans ce paragraphe ont été obtenus avec le scénario "ÉCLAIRAGE" décrit au chapitre III.
Notre objectif ici est de comparer notre protocole de formation de coalitions avec un autre basé sur les Contract Net. On a testé les deux protocoles sur différents critères et nos expérimentations ont été effectuées sur un système d'agents dont la cardinalité varie. Les résultats sont déclinés en trois catégories :

- Le nombre de coalitions formées pour un initiateur avec une cardinalité supérieure ou égale à 2,
- la comparaison du temps de réponse de chaque protocole,
- le nombre de messages échangés lors de la formation de coalitions.

Nous avons d'abord analysé le nombre de coalitions formées par chaque protocole (figure IV.23).

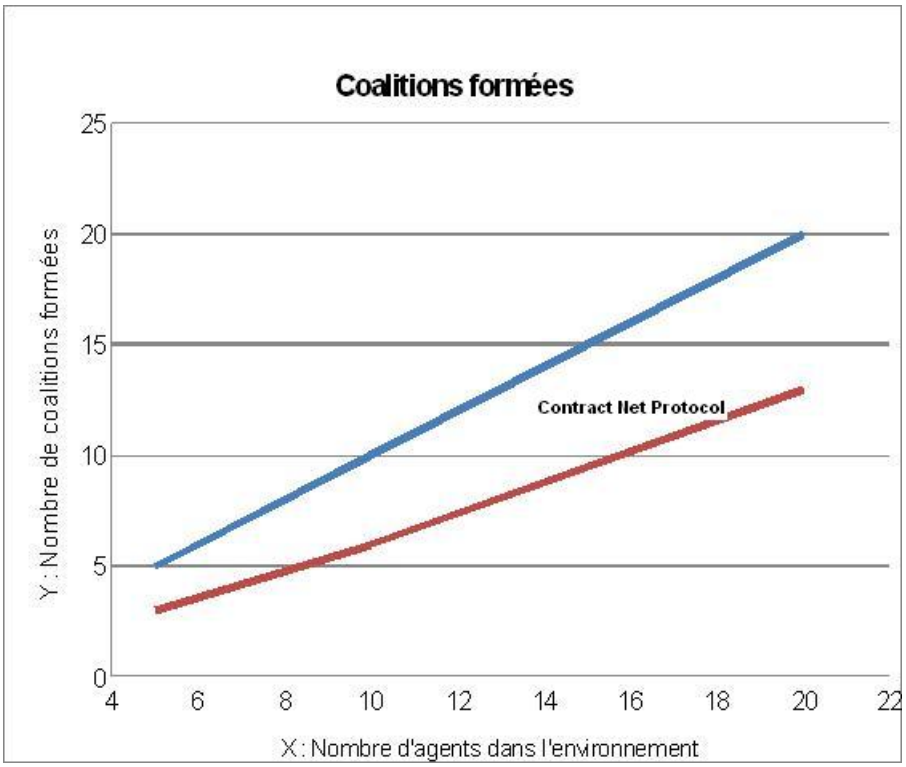

Figure IV.23 : Nombre de coalitions formées par rapport aux agents dans l'environnement

La stratégie que nous privilégions dans notre approche est d'obtenir un maximum de coalitions répondant aux critères de sélection que nous nous sommes fixés. Ces critères sont mis en jeu au cours des deux étapes de formation des coalitions et de négociation. L'objectif est de disposer de plusieurs solutions pour répondre à la requête ainsi nous augmentons les chances de garantir un résultat. Le nombre de coalitions dans notre approche est toujours égal aux nombres d'initiateurs, par exemple pour cinq agents dans l'environnement, on obtiendra cinq coalitions formées. On voit que sur le nombre de coalitions, le protocole Contract Net est moins performant.

Dans un deuxième temps, nous avons cherché à estimer le temps de réponse de chaque protocole. La figure IV.24 illustre les résultats obtenus. La formation de coalitions est plus rapide avec le protocole Contract Net que notre approche.

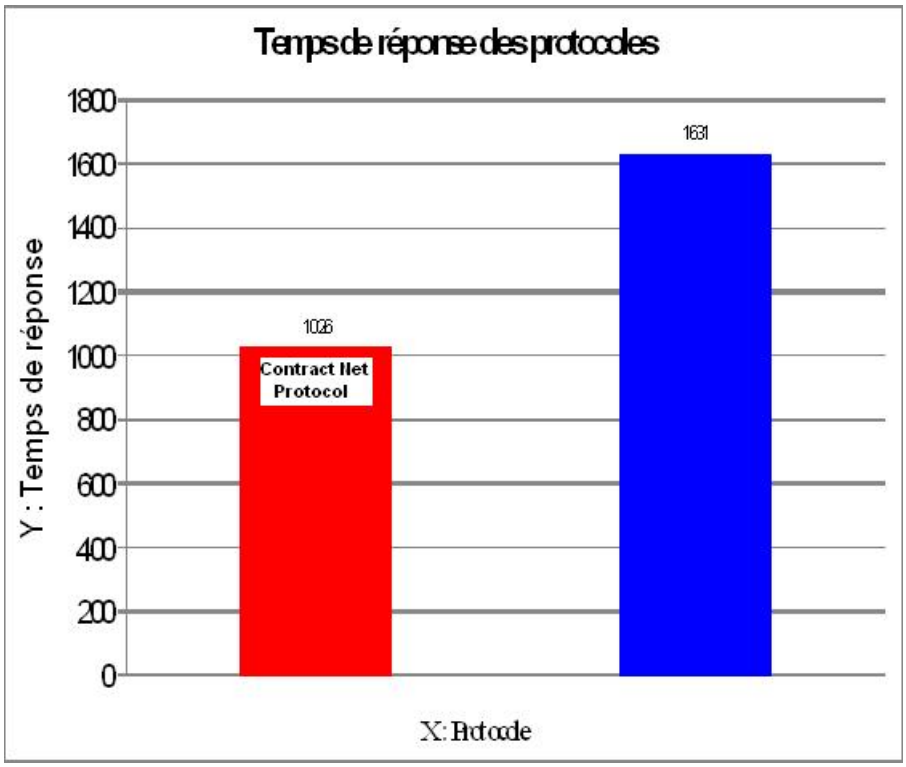

Figure IV.24 : Temps de réponse

En effet, le fait que le nombre de coalitions qui peuvent se former est inférieur aux nombres d'initiateurs a un effet direct sur le temps de réponse. Il impacte aussi le nombre de messages échangés (figure IV.25). Il faut noter que nous avons recherché la faisabilité sans contraintes temps réel d'où des temps de réponse de plusieurs secondes. Il est possible d'améliorer sensiblement ce critère. En effet, il peut y avoir des solutions qui permettraient à la formation de coalition d'être plus rapide, comme améliorer les critères de choix d'une ou des coalitions en introduisant l'aspect temps réel dans les choix. Comme la plateforme JADE peut gérer plusieurs machines en même temps, il serait aussi possible de distribuer les ressources machines que les agents consomment en déployant les agents sur des machines dédiées spécialement à chacun de leur type.

Le nombre de messages échangés (figure IV.25) augmente dans les deux protocoles en fonction du nombre des agents qui forment des coalitions. La stratégie choisie explique que notre protocole génère plus d'échanges. De plus, dans notre approche, un agent initiateur a formé une coalition si et seulement si les agents ont accepté d'en faire partie puis que l'initiateur le leur a confirmé, ce qui engendre un accroissement des échanges par rapport au protocole Contract Net.

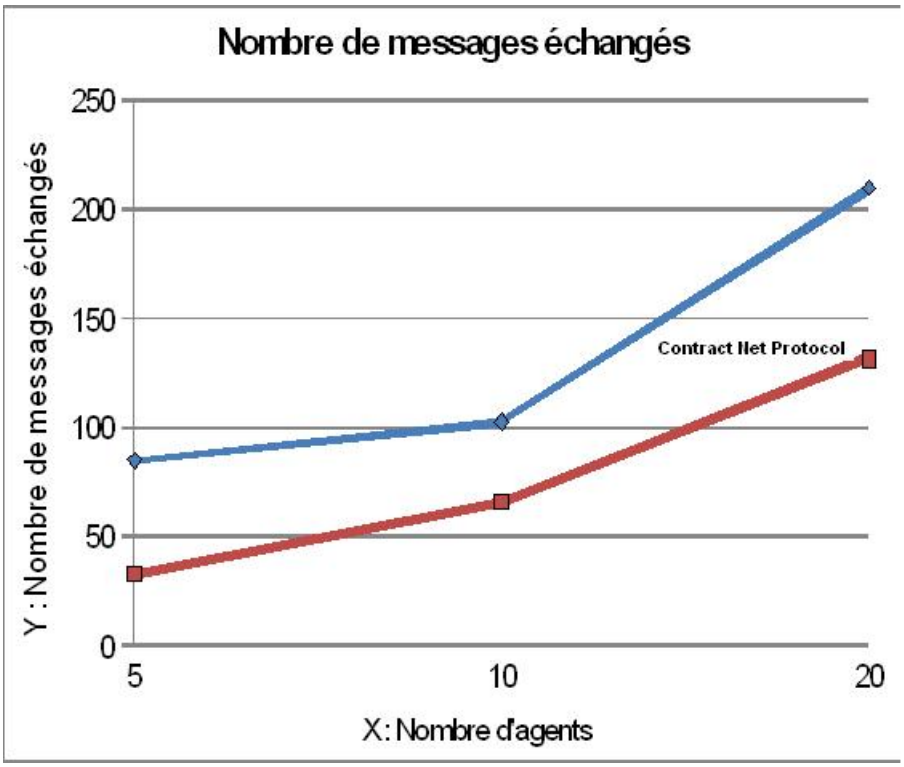

Figure IV.25 : Nombre de messages échangés

La courbe du nombre de messages échangés pour notre protocole suit la même allure que celle du Contract Net. Par contre, notre protocole accuse un nombre plus élevé de messages échangés du fait de la stratégie des agents qui continuent toujours de coopérer jusqu'à l'obtention d'un résultat pouvant répondre à l'effet souhaité. Mais aussi parce que contrairement au protocole Contract Net, pour éviter que le système ne se bloque, la stratégie de former d'abord des coalitions minimales dont la cardinalité s'accroît progressivement en fonction de la solution recherchée augmente sensiblement le nombre de messages échangés.

IV.3.2 Localisation par coopération multi-agent

L'objectif de cette évaluation est de valider le fonctionnement de l'ensemble du système coopération robot-environnement pour la localisation du robot. Les résultats des tests ci-dessous sont dans le cas où le robot est donc détecté par les capteurs de l'environnement et dans le cas ou la caméra du robot détecte des marqueurs de l'environnement.

IV.3.2.1 Objectifs des évaluations

L'évaluation a permis de réaliser en situation réelle :

– Le test l'intégration des constituants de l'environnement ambiant composé de capteurs domotiques de type capteur de présence, de capteurs goniométriques basés sur

une caméra et la détection de marqueurs visuels de type datamatrix et du robot,
- le test des protocoles d'échanges entre la passerelle et l'environnement ambiant, et la passerelle et le SMA (Système Multiagent),
- le test des mécanismes d'adaptation, coalition, négociation et la validité des critères,
- le test de la méthode de localisation.

Les tableaux IV.4, IV.5, IV.6, IV.7, suivants rappellent les coordonnées et les caractéristiques des capteurs, des marqueurs et de la pièce.

Détecteur de présence	**Identifiant**	**Position (m)**	**Orientation (°)**	**Précision(°)**
0E D6 01	0000 0	(6.34, 0)	135	±45
0E 80 41	0000 1	(0, 0)	45	±45
07 FB 51	0000 2	(0, 9.4)	315	±45
08 08 E1	0000 3	(6.34, 9.4)	225	±45

Table IV.4 : Coordonnées des détecteurs infrarouges et précision

Identifiant Datamatrix	**Position (m)**
N	(3.17, 9.4)
S	(3.17, 1.1)
E	(0, 4.7)
O	(6.34, 2.35)
X	(6.34, 4.7)

Table IV.5 : Coordonnées des datamatrix del'environnement

Caméra	**Identifiant**	**Position**	**Orientation**	**Précision**	**Champ de vision**
Caméra Fixe 1 (CF1)	3000 1	(4.64, 7)	225	±45	±55/2
Caméra Fixe 2 (CF2)	3000 2	(3, 2.2)	90	±45	±55/2
Caméra robot (CR)	2000 1	(X_{robot}, Y_{robot})	θ_{robot}	±45	±55/2

Table IV.6 : Coordonnées des caméras

Dimension	**Valeur (en m)**
Longueur	9.4
Largeur	6.34

Table IV.7 : Dimension de la pièce

IV.3.2.2 Protocole expérimental

Le protocole expérimental des scénarios des tests est le suivant :

1. Placer le robot à un point précis de la pièce. Cette position exacte du robot sera comparée à la position calculée par l'algorithme de localisation après que les agents auront formé une coalition.
2. Lancer le SMA qui attend les mesures des différents capteurs.
3. Lancer la requête de localisation aux SMA.
4. Les agents AA-CRE vont former des coalitions et négocier pour proposer une solution.
5. Une fois qu'une coalition gagnante a été formée, on prend la capacité de chaque agent pour les rentrer dans l'algorithme de localisation.

IV.3.2.3 Résultats expérimentaux

IV.3.2.3.1 Détection du robot seulement par les capteurs de l'environnement

Pour les premiers essais de localisation, les mesures provenant de la caméra du robot ont été exclues afin de tester le fonctionnement global de l'architecture et examiner la réponse des mécanismes d'adaptation.
La position (x_R, y_R) du robot à partir de la connaissance de la mesure (λ_i) des capteurs de l'environnement, de leurs coordonnées et de leur orientation (figure IV.26).

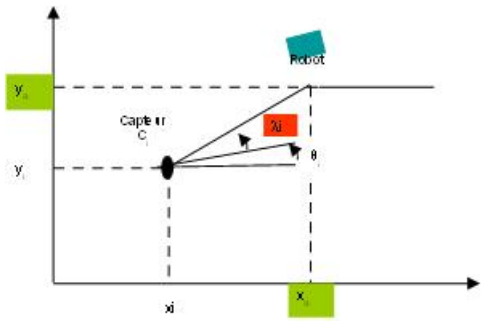

Figure IV.26 : Le robot est détecté par un capteur de l'environnement, noté C

Situation 1

Le tableau IV.8 présente les coalitions gagnantes selon que l'on autorise des coalitions de deux ou trois agents capteurs. La première colonne donne les agents membres de la coalition gagnante et les informations utilisées pour le calcul de localisation. Par exemple, la première ligne nous informe que la coalition est constituée du capteur de présence IR2 dont les coordonnées sont (0, 9.4)m et l'orientation 315° par rapport au repère de

l'habitat. Comme la précision de ce capteur est $\pm 45°$ et qu'il est situé dans un coin de la pièce, l'information qu'il délivre se limite à "présence du robot dans cette pièce". Sa mesure est en fait son orientation. Le deuxième membre de la coalition est une caméra fixe dont les coordonnées sont (4.64, 7) m et la mesure est 2.6°.

Coalition gagnante	**Mesure λ en °**
$[IR000002, CF1]$	$[315, 2.6]$
$[IR000001, IR000002, CF1]$	$[Pasdemesure, 315, 2.6]$

Table IV.8 : situation 1

La deuxième ligne du tableau correspond au résultat d'une coalition à trois membres. Comme les critères pour établir des coalitions puis ensuite déterminer la coalition gagnante sont différents, il se peut qu'un agent capteur retenu dans une coalition lauréate ne soit pas capable de délivrer une mesure. C'est le cas du capteur de présence IR1. Le choix des critères entre les phases de coalition et de négociation n'est pas trivial et demande une réflexion supplémentaire pour établir un compromis entre le nombre de coalitions pour augmenter les chances d'obtenir une solution et leur pertinence.

La figure IV.27 donne le résultat de localisation du robot en utilisant la coalition à deux agents. C1 représente le capteur de présence et C2 la caméra. La position théorique du robot est représentée avec son incertitude par une ellipse. Le pavage bleu représente les pavés acceptables, le pavage jaune les pavés ambigus.

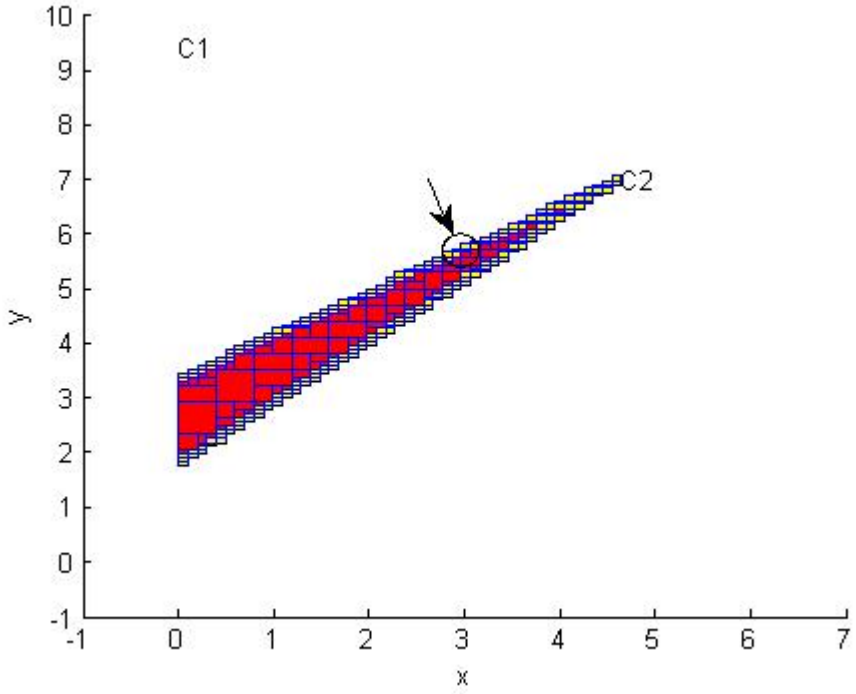

Figure IV.27 : Localisation pour la situation 1

La position réelle du robot est (3, 5.71) ±0.2 m.

Situation 2

Le tableau IV.9 suivant traduit une autre situation.

Coalition gagnante	**Mesure λ en degrés**
$[IR00000, CF2]$	$[135, -3.5]$
$[IR00000, IR00002, CF1]$	$[135, -7.8, -3.5]$

Table IV.9 : Situation 2

La figure IV.28 donne le résultat de localisation du robot en utilisant la coalition à trois agents. C1 et C2 représentent deux caméras fixes et IR0, le capteur de présence. La position théorique du robot est représentée avec son incertitude par une ellipse. Le pavage bleu représente les pavés acceptables, le pavage jaune les pavés ambigus.

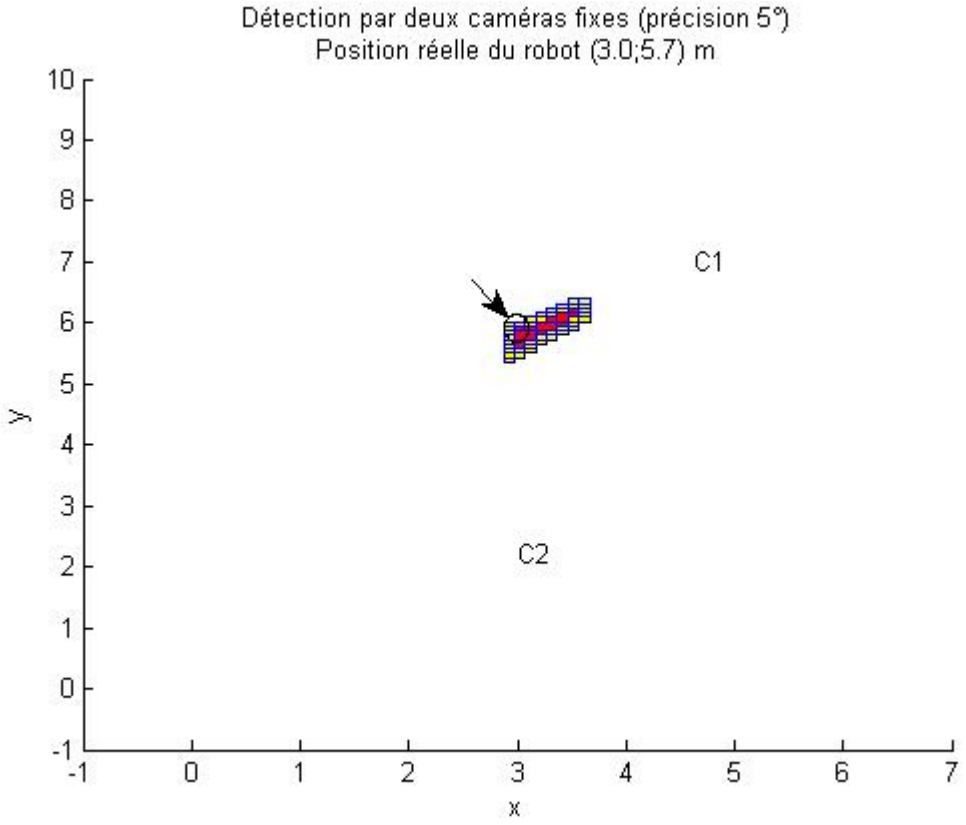

Figure IV.28 : Localisation pour la situation 2

La position réelle du robot est $(3, 5.71)$ ± 0.2 m. On remarque que la position de localisation du robot est connue avec une meilleure précision par le fait que les mesures effectuées avec les caméras sont précises, ce qui était prévisible.

IV.3.2.3.2 Détection du robot par les capteurs de l'environnement et la caméra du robot

Situation 3

Dans ce cas, la caméra orientable du robot peut détecter plusieurs marqueurs visuels placés dans l'environnement. La coalition à trois agents de la première ligne du tableau IV.10 est composée de deux caméras fixes C1 et C2 ainsi que de la caméra du robot qui a détecté le marqueur Datamatrix N dont les coordonnées sont (3.17, 9.4) m dans le repère de l'habitat.

Coalition gagnante	Mesure λ en degrés	Remarques
$[CR, CF1, CF2]$	$[3.1, 24.6, 4.4]$	Datamatrix N (3.17, 9.4)
$[IR0, CR, CF1, CF2]$	$[135, 3.1, 159.6, -4.4]$	Datamatrix N (3.17, 9.4)

Table IV.10 : Situation 3

La figure IV.29 donne la position du robot en utilisant seulement les mesures issues des deux caméras fixes C1 et C2. La position réelle du robot est (3, 3.2) ±0.2 m.

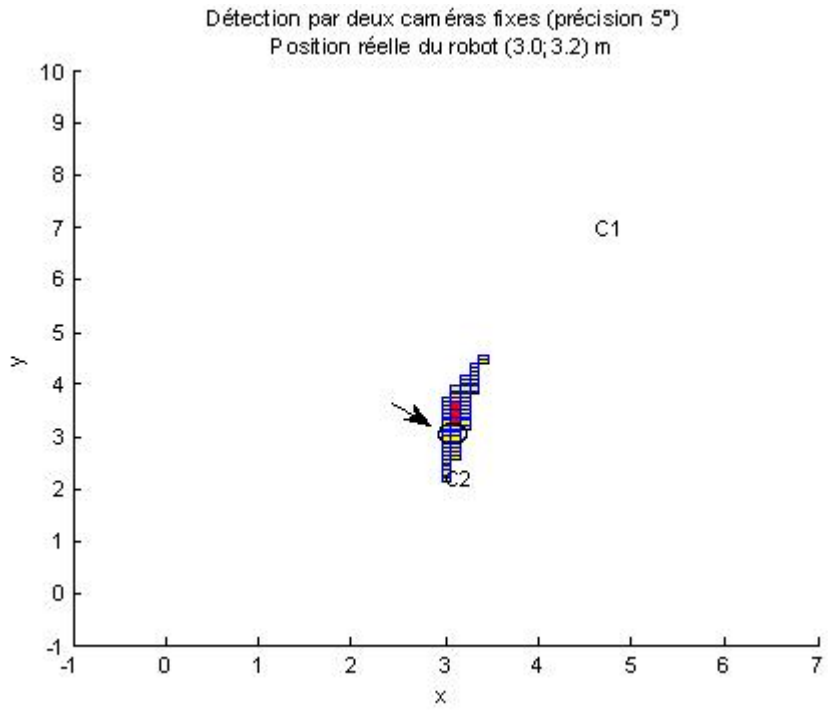

FIGURE IV.29 : Localisation (x, y) pour la situation 3 avec deux agents

Cependant le fait qu'une troisième mesure ait pour origine le robot permet de calculer en plus l'orientation de celui-ci. La figure IV.30 donne la position du robot en utilisant la coalition à trois agents. C1 et C2 représente les deux caméras fixes et M3 le marqueur visuel détecté par la caméra du robot.

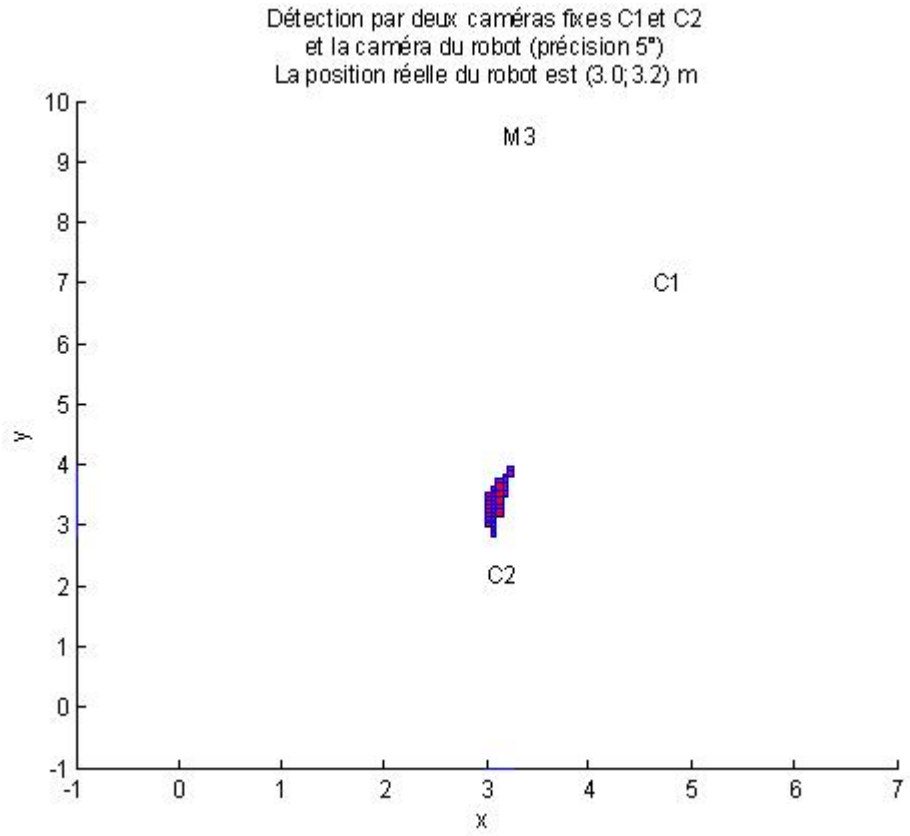

Figure IV.30 : Localisation (x, y) pour la situation 3 avec trois agents

D'une part la précision de la position est améliorée par cette troisième mesure et d'autre part l'orientation du robot peut être calculée. La figure IV.31 représente l'orientation du robot en fonction de la coordonnée en y de celui-ci.

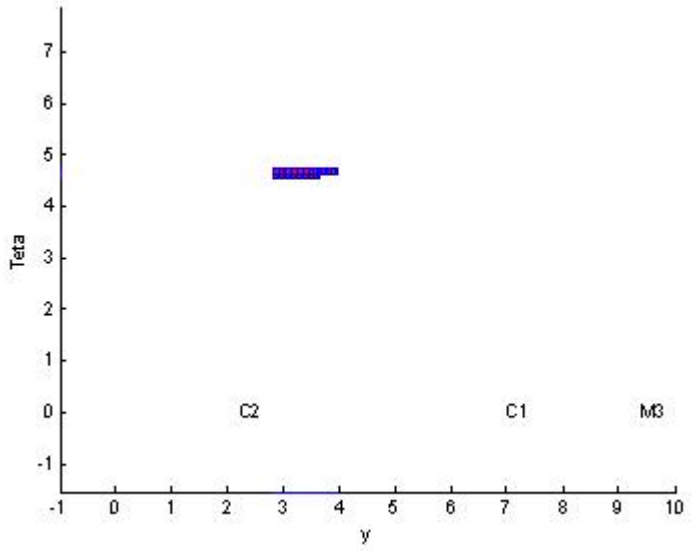

Figure IV.31 : Localisation (y, θ) pour la situation 3 avec trois agents

Situation 4

Dans ce cas, afin de limiter la gêne occasionnée à la personne, le degré d'intrusion fixé au départ n'autorisait pas le mouvement de la caméra du robot. Dans un premier temps le mécanisme d'adaptation n'a pu aboutir à une coalition lauréate, aucune mesure récente n'étant disponible. Dans un deuxième temps, le mouvement de la caméra du robot a été autorisé. Ce qui a permis d'obtenir une coalition lauréate (tableau suivant).

Coalition gagnante	**Mesure λ en °**	**Remarques**
$[CR, CF1, CF2]$	$[93.7, 3.6, -4.3]$	Datamatrix O (6.34, 2.35)

Table IV.11 : Situation 4

La position réelle du robot est (3, 5.71) ±0.2 m. Si on compare la figure IV.32 qui donne la position du robot en utilisant seulement les mesures issues des deux caméras fixes C1 et C2 et la figure IV.33 qui utilise la troisième mesure issue du robot, la précision de localisation n'est pas améliorée dans ce cas, mais on dispose de l'orientation du robot.

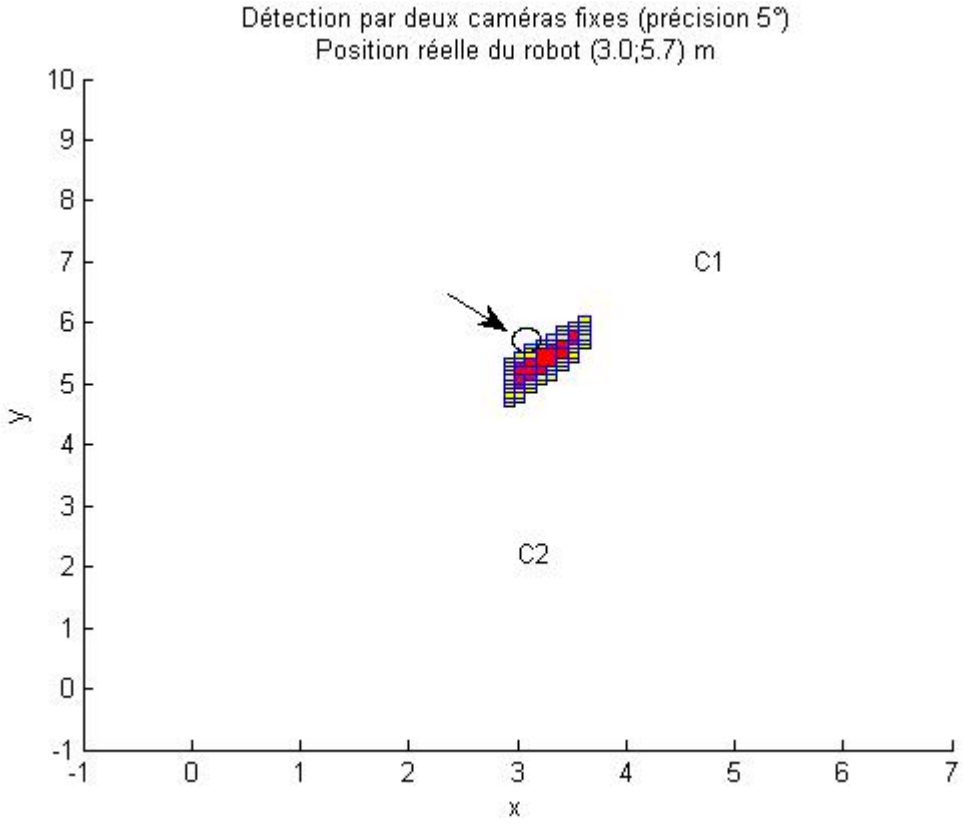

Figure IV.32 : Localisation (x, y) pour la situation 4 avec deux agents

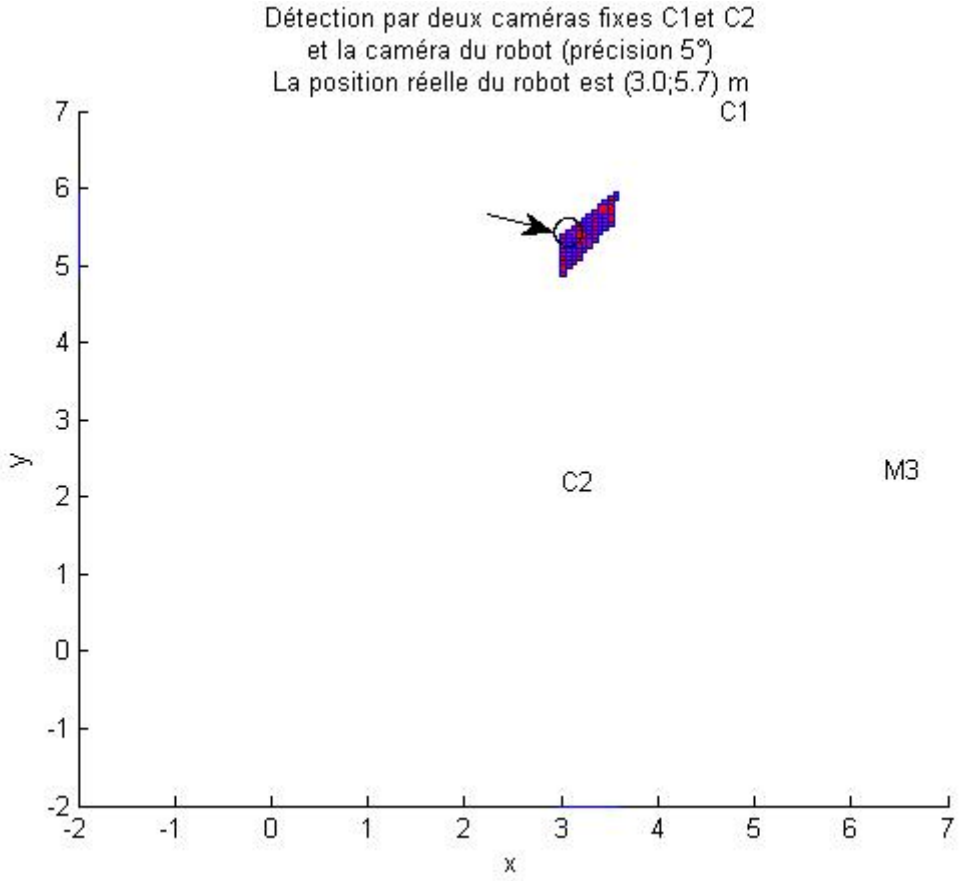

Figure IV.33 : Localisation (x, y) pour la situation 4 avec trois agents

IV.4 Conclusion

Ce chapitre a présenté la mise en œuvre d'une architecture de Coopération Robot-Environnement composée d'un SMA, d'une ontologie et d'un environnement de capteurs/actionners. En accord avec l'objectif de la thèse, l'évaluation est réalisée plus pour illustrer la faisabilité de l'approche que pour évaluer ses performances. La description des différents modules développés montre la coherence du systéme. Les tests sur le protocole employé au niveau des agents AA-CRE du SMA ont permis de vérifier la réponse aux contraintes posées au Chapitre I, à savoir, une obligation de résultat et une capacité d'adaptation du système. Deux scénarios, "éclairage" et "localisation", ont permis d'une part, de confronter notre système à ce qui est observé, en termes de mesures et d'informations obtenues, et d'autre part, de comparer les performances du protocole employé dans le SMA à un protocole bien connu, basé sur les réseaux contractuels. Les résultats montrent que les courbes associées à notre système suivent la même allure qu'avec un protocole à base de réseaux contractuels, en termes de nombre de messages échangés et de temps de réponse. Toutefois, notre système est plus gourmand en temps. Cela s'explique par le

fait qu'il continue de se réorganiser jusqu'à trouver une solution (même dégradée), alors qu'avec les réseaux contractuels, le système peut échouer et donc, peut ne pas proposer de solution.

Les points forts de notre protocole sont :
- Pour respecter l'effet souhaité avec l'obligation de résultat, plusieurs coalitions possibles se forment et la meilleure est choisie pour être déployée,
- les coalitions d'agents qui se forment, s'adaptent en fonction des ressources des capteurs qui sont disponibles dans l'environnement,
- tant que les agents n'ont pas trouvé de solutions qui respectent l'effet souhaité, les agents continuent de s'organiser.

La deuxième évaluation présentée dans ce chapitre teste l'ensemble de l'architecture. La confrontation des résultats de la méthode de localisation avec les mesures obtenues est concluante. L'écart entre la solution calculée et la position réelle du robot est faible, sachant que les coordonnées réelles du robot sont mesurées avec un mètre dépliant et que les coordonnées des capteurs de l'environnement (en particulier, leur orientation) ont été mesurées de façon imprécise.

La mise en œuvre et l'implémentation de notre architecture de Coopération Robot-Environnement ont permis de valider la cohérence de l'ensemble, mais aussi le fonctionnement unitaire des différents composants du système. Le principal point d'amélioration est le temps de réponse du système qui passe par une étude plus fine des divers domaines impliqués pour chacun des composants, avant un choix d'une technologie.

Conclusion générale et perspectives

La rencontre de l'intelligence ambiante et de la robotique d'assistance a donné naissance à ce que nous avons appelé robotique ambiante d'assistance car destinée à assister une personne en perte d'autonomie. Cette problématique s'appuie sur l'existence de réseaux d'objets communicants présents dans l'environnement de la personne pour décliner un ensemble de services et de téléservices destinés à faciliter la vie quotidienne de cette personne et de son entourage. L'idée générale est d'exploiter ce réseau d'objets communicants pour fiabiliser, améliorer, étendre les capacités d'un robot mobile dans sa participation aux services rendus à la personne. La complexité de la problématique nous a amené à nous focaliser sur la localisation du robot qui reste une question essentielle à résoudre en robotique mobile. L'objectif est de pouvoir localiser avec certitude le robot dans une zone de l'habitat.

En termes d'usage, nous avons opté pour un scénario de télévigilance défini en partenariat avec le SAMU-92 qui dépend de l'AP-HP (Assistance Publique- Hôpitaux de Paris) au cours de la phase de spécification du projet ANR TecSan QuoVadis (2008-2011). Dans ce scénario, le robot participe à la levée de doute. La réussite de la mission dépend de la capacité du robot à se déplacer d'un point de départ vers une destination de façon fiable, et ce quelles que soient les situations rencontrées. Pour effectuer ce déplacement, le robot a besoin de se localiser régulièrement.

En termes de problématique scientifique, les travaux de la thèse ont pour ambition de contribuer à fiabiliser la fonction localisation dans l'habitat standard d'une personne, domicile ou institution, grâce à la coopération entre le robot et les objets communicants présents dans l'environnement tels que les capteurs ou les actionneurs. Pour garantir l'autonomie du robot sur une période importante, il est nécessaire de le doter de mécanismes d'adaptation lui permettant de répondre à des situations diverses, pas forcément prévues par le concepteur.

Les trois questions principales auxquelles nous avons cherché à apporter des éléments de réponse sont : La localisation du robot en environnement intérieur par coopération avec un environnement ambiant. Comment obtenir un résultat de localisation garanti, sous certaines hypothèses à définir, en présence d'un réseau de capteurs hétérogènes dont le nombre et les caractéristiques sont variables. L'idée est que la connaissance fiable de sa localisation, même si celle-ci est dégradée (imprécise), permettra au robot d'adapter sa

stratégie de déplacement en fonction de la situation. L'adaptation au contexte. Quel(s) mécanisme(s) d'adaptation pour gérer : i) la variété et la disponibilité des capteurs, ii) les contraintes liées au service rendu à la personneă(degré d'intrusion, degré d'urgence). Une architecture informatique capable d'accueillir ce(s) mécanisme(s) d'adaptation.

Contribution à la localisation en environnement ambiant La méthode de localisation basée sur le principe de la multiangulation. La résolution du problème est obtenue par une approche ensembliste, plus précisément l'analyse par intervalles. Comme la dimension du vecteur d'état est inférieure ou égale à trois, les calculs sont relativement simples et a priori rapide (la simulation étant réalisée sous Matlab, une évaluation réaliste devra être effectuée pour estimer le temps moyen d'exécution). Le modèle simple retenu pour les capteurs donne la possibilité d'intégrer une grande variété de capteurs, du plus fruste au plus complexe, si celui-ci délivre une mesure d'angle. Accessoirement, l'algorithme de calcul de la fonction intervalle [arctg a] est original et peut être utile à d'autres travaux. La méthode est capable de s'adapter à plusieurs facteurs :

- un nombre variable de mesures ;
- des mesures hétérogènes au sens métrologique du terme (précision, portée) ;
- une connaissance statistique limitée sur ces mesures ;
- Généralement l'erreur de mesure n'est connue que sous la forme d'une tolérance ;
- un placement des capteurs/marqueurs dans le repère de l'environnement et dans celui du robot ; item une dispersion plus ou moins importante des capteurs et des marqueurs dans l'environnement.

Un résultat de localisation, même partiel, peut être obtenu dès qu'une donnée mesurée est exploitable.

Les résultats de simulation montrent l'intérêt de cette approche. Le choix du paramètre epsilon permet régler le compromis entre la précision, par la réduction du pavage ambigu, et le temps de calcul. Les résultats expérimentaux confirment l'efficacité de cet algorithme.

Des améliorations peuvent être apportées sur plusieurs points. Tout d'abord, les mesures aberrantes qui violent de l'hypothèse d'erreur bornée ne sont pas traitées. Il existe des pistes dans la littérature pour y remédier. Par ailleurs, il a été considéré, pour simplifier la présentation de l'approche, que la localisation des capteurs et des marqueurs de l'environnement était connue. Mais il est facile d'introduire cette incertitude dans l'algorithme, bien entendu au détriment de la précision de localisation.

Contribution à l'adaptation à des situations variées L'adaptation est nécessaire afin de pallier le caractère dynamique des environnements ambiants. La mobilité des utilisateurs et des périphériques qui entrent et quittent un environnement ambiant se traduit par une disponibilité dynamique des informations que peuvent fournir les différents capteurs et actionneurs de l'environnement. Les réponses que nous proposons repose à la fois sur le choix d'une architecture distribuées mais aussi de mécanismes de coalition et

de négociation. Nous avons proposé une architecture de coopération robot-environnement constitué d'un SMA, d'un environnement ambiant et d'une interface utilisateur qui interagit avec le SMA. L'environnement ambiant est constitué d'un environnement physique qui regroupe des objets communicants, d'un robot, d'une passerelle et d'une base de connaissance.

La base de donnée ontologique qui a été développée dans le cadre de cette est une solution motivée par la structure hiérarchisée d'une ontologie et sa généricité par rapport au cadre de l'assistance ambiante. L'ontologie a assuré la généricité de toutes les informations persistantes qu'un habitat de type domotique pourrait comprendre ainsi que les différents caractéristiques que chaque objet communicant pourraient avoir. Toutefois, des améliorations sont encore possibles, à savoir l'intégration d'informations métriques ajoutées aux données topologiques de l'habitat et l'utilisation d'un moteur d'inférence intelligent qui améliorerait l'interrogation de l'ontologie par les agents.

Pour répondre à la problématique de l'adaptation en fonction du contexte et du respect du niveau de gêne de l'utilisateur, la solution proposée est basée sur un mécanisme de formation de coalitions d'agents AA-CRE. Ces coalitions se constituent pour répondre à un besoin d'obtention d'effet souhaité. Chaque objet de l'environnement ambiant est encapsulé dans un agent AA-CRE. L'obligation de résultat et le respect d'un niveau d'intrusion en fonction de l'urgence de la situation, sont les critères les plus importants considérés au niveau des règles de constitution des coalitions. Ils sont aussi employés lors de la réorganisation des agents pour la recherche d'un effet souhaité. Le critère d'obligation de résultat est employé en priorité, tandis que le niveau d'intrusion n'est pris en compte que s'il est nécessaire d'acquérir de nouvelles données et donc d'actionner des capteurs susceptible de provoquer une gêne pour la personne. Comme hypothèse de travail, nous avons considéré que les agents acceptent toujours de participer à des coalitions. Ce choix étant fonction de leur voisinage immédiat, il serait pertinent d'améliorer ce critère en ajoutant des informations sur les affinités entre les agents. Par exemple, il serait possible de tenir compte de coalitions passées qui ont été satisfaisantes ou bien de mettre en place des stratégies de "couplage" d'agents en fonction du fait qu'il y ait une potentialisation ou non leur association.

Les points forts de notre protocole sont :

1. Pour respecter l'effet souhaité avec l'obligation de résultat, plusieurs coalitions possibles se forment et la meilleure est choisie pour être déployée,
2. les coalitions d'agents qui se forment s'adaptent en fonction des ressources des capteurs qui sont disponibles dans l'environnement,
3. tant que les agents n'ont pas trouvé de solutions qui respectent l'effet souhaité, les agents continuent de s'organiser.

Perspectives à court terme :
Au niveau du protocole de coopération Robot-Environnement, la formation de coalitions se base sur les critères du voisinage, de la précision qui est la capacité de l'agent et les mesures récentes à la phase de négociation. Une perspective à court terme est d'améliorer la formation des coalitions à l'initialisation. En effet, le choix de critères qui permettrait d'affiner la sélection des agents candidats et de réduire le nombre de messages échangés (amélioration du temps de réponse du système) est crucial et mérite une plus ample réflexion.

Au niveau de l'ontologie, on peut augmenter le niveau de granularité pour rassembler et affiner les différents renseignements et informations que le domaine de l'Assistance Ambiante offre sans remettre en cause l'existant.

Au niveau de l'ensemble de l'architecture, il est nécessaire d'évaluer le système dans un plus grand habitat présentant une variété plus élevée de capteurs et d'actionneurs.

Perspectives à moyen terme :
Actuellement, il est impossible de vérifier la validité du résultat d'une réponse à un effet souhaité. Une possibilité pour pallier à ce problème est de faire une optimisation multi-critères pendant la formation des coalitions et la négociation. L'optimisation permettrait d'obtenir un résultat fiable à la sortie d'une coalition gagnante.

Il serait interessant de doter les agents de comportements plus complexes basés sur les états mentaux de groupe et les réseaux de dépendance sociale. La connaissance par un agent des comportements des autres agents peut lui permettre de mieux raisonner afin de se rapprocher de ses propres buts en fonction de ce qu'il connaît des autres.

Actuellement, l'algorithme de localisation est lancée une fois qu'une coalition est formée. Au niveau de la localisation, il est possible d'intégrer directement des comportements inhérents aux mesures de chaque capteur à chaque agent.

Au niveau de la localisation, l'algorithme doit être complété pour mieux gérer les cas où plus de trois mesures sont disponibles, et traiter les valeur aberrantes.

Perspective à long terme
Comme perspective à long terme, il serait envisageable de proposer une architecture orientée service. Globalement, à l'identique de la communauté "middleware" vu au chapitre I, le principe étant de placer une couche intermédiaire entre les agents et les objets communicants. Cette couche servira à la création dynamique de services que les objets com-

municants avec le robot pourraient offrir. De façon générale, le modèle permettrait à un système de répondre à n'importe quel effet souhaité de type Assistance Ambiante de plus haut niveau que celui traité dans la thèse (localisation).

Annexes

Annexe A

Codes sources

Prototype d'un Agent AA

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.io.Serializable;
import java.net.InetAddress;
import java.net.Socket;
import java.net.UnknownHostException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Vector;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import javax.swing.ImageIcon;
import javax.swing.JOptionPane;
import com.sun.org.apache.xerces.internal.impl.dv.xs.DateDV;
import jade.core.AID;
import jade.core.behaviours.CyclicBehaviour;
import jade.core.behaviours.OneShotBehaviour;
import jade.core.behaviours.ParallelBehaviour;
import jade.core.behaviours.WakerBehaviour;
import jade.domain.DFService;
import jade.domain.FIPAException;
import jade.domain.FIPAAgentManagement.DFAgentDescription;
import jade.domain.FIPAAgentManagement.SearchConstraints;
import jade.domain.FIPAAgentManagement.ServiceDescription;
import jade.lang.acl.ACLMessage;
import jade.lang.acl.MessageTemplate;
import jade.lang.acl.UnreadableException;
public class test2 extends PrototypeAgentAA {
private static int Compte = 0; /** Compteur incrémentable **/
private  int informe = 0;
private Vector<String> DatePermute = new Vector<String>();
private Vector<String> NomPermute = new Vector<String>();
private Vector<Double> PrecisPermute = new Vector<Double>();
private Vector<String> TypePermute = new Vector<String>();
private Vector<String> Initiateurs = new Vector<String>();
private Vector<String> NomNeg = new Vector<String>();// Nom a la négociation
private Vector<Integer> PrecisNeg = new Vector<Integer>();// Precisison a la négociation
private Vector<String> L2 = new Vector<String>(); /** Vecteur intermediaire pour les dates **/
```

```
private Vector<String> L = new Vector<String>(); /** Vecteur intermediaire pour les Noms **/
private Vector<Double> L3 = new Vector<Double>(); /** Vecteur intermediaire pour les Precisions **/
private Vector<String> egal = new Vector<String>(); /**EGAUX DANS LA NEGOCIATION**/
private Vector<String> L4 = new Vector<String>();
private int MaxCombine;
private int N = 0,compt = 0;
private int Q = 0;
private int Membre = 0; /** Pour les accepte coal **/
private double P = 0.0, M;
private Vector<String>  Nomsup = new Vector<String>();
private Vector<Integer>  Precissup = new Vector<Integer>();
private Vector<String>  Nomeg = new Vector<String>();
private Vector<Integer>  Preciseg = new Vector<Integer>();
private int somme = 0;
private Vector<String> vect = new Vector<String>();
private Vector<String> vectDate = new Vector<String>();/** Enregistre des mesures des capteurs**/
private Vector<Double> vectPrecis = new Vector<Double>();/** Enregistre les précisions des agents **/
private Vector<String> vectTypes = new Vector<String>();
private int MembreCoalition = 2;
private Vector<String> egal2 = new Vector<String>();
private Vector<String> Nomsup2 = new Vector<String>();
private Vector<Integer> NbrMesure = new Vector<Integer>();
private int NbrRelance = 0;
boolean Iam;
private double somme2;
boolean Iam2 = true, Iam3 = true;
@Override
double getAngle() {
// TODO Auto-generated method stub
return Angle_Detection;
// ......  get et set
protected void setup(){

Verifie = Pattern.compile("00000.\\+");
System.out.println("Agent " + getLocalName()
+ " De Type Detecteur de Presence Prêt ! " + MembreCoalition );
NbrAgent = NbrAgent + 1; /**Incrémentatation des nombres d'agents crée **/
try {
Acces_Ontologie("OWL\\DetecteurPresence.txt");
System.out.println(" Precicion = " + getPrecision());
} catch (NumberFormatException e) {
// TODO Auto-generated catch block
e.printStackTrace();
} catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
setType("DetecteurPresence");
this.InscriptionDF(getAID(),this); /**Inscription dans une page jaune **/
informe = Compte;
Compte =  Compte + 1;
ParallelBehaviour s = new ParallelBehaviour();
s.addSubBehaviour(new CyclicBehaviour() {
@Override
public void action() {
        // TODO Auto-generated method stub
ACLMessage msg = receive(MessageTemplate.MatchPerformative(ACLMessage.REQUEST));
if(msg != null){
contenu_trame = msg.getContent();
```

```
matcher = Verifie.matcher(contenu_trame);
if(matcher.find()){
String Parse[] = contenu_trame.split("\\+");
if (getIdentifiant() == Integer.valueOf(Parse[0]).intValue()){
trouve(contenu_trame, Parse);
System.out.println(getLocalName() + "  " +  Parse[0] );
}
else {
if(getIdentifiant() == -1){
if(informe == Integer.valueOf(Parse[0]).intValue()){
setIdentifiant(Integer.valueOf(Parse[0]).intValue());
trouve(contenu_trame,Parse);
if(getIdentifiant() == 000000){
PositionX = 6.34;
PositionY = 0.0;
teta = 45;
System.out.println(getLocalName() + "  " +  Parse[0] +" Mon ID "
+ getIdentifiant() + PositionX + " " + PositionY);
}
else {
if(getIdentifiant() == 000001){
        PositionX = 0.0;
        PositionY = 0.0;
        teta = 315;
        System.out.println(getLocalName() + "  " +  Parse[0] +" Mon ID "
    + getIdentifiant() + PositionX + " " + PositionY);
}
else {

if(getIdentifiant() == 000002){
        PositionX = 0.0;
        PositionY = 9.4;
        teta = 215;
        System.out.println(getLocalName()
    + "  " +  Parse[0] +" Mon ID " + getIdentifiant() + PositionX + " " + PositionY);
}
        else {
                if(getIdentifiant() == 000003 ){
                PositionX = 6.34;
                PositionY = 9.4;
                teta = 135;
                System.out.println(getLocalName() + "  " +  Parse[0] +
        " Mon ID " + getIdentifiant() + PositionX + " " + PositionY);}}}}}}}}}
else {
                block();}}
});
s.addSubBehaviour(new CyclicBehaviour() {

@Override
public void action() {
// TODO Auto-generated method stub
ACLMessage msg = receive(MessageTemplate.MatchPerformative(ACLMessage.INFORM));
if(msg != null){
        System.out.println("Agent " + getLocalName()+ " A Reçu un Init Effect " );
        Effet_Reçu = msg.getContent();
        addBehaviour(new SendInitCoal());
}
else {
        block();
```

```
}}});
s.addSubBehaviour(new CyclicBehaviour() {
@Override
public void action() {
        // TODO Auto-generated method stub
        ACLMessage msg = receive(MessageTemplate.MatchPerformative(ACLMessage.FAILURE));
        if(msg != null){
                relance();
                doWait(5000);
                addBehaviour(new SendInitCoal());
        }else {
                block();
        }
}
});
addBehaviour(s);
}
private class SendInitCoal extends OneShotBehaviour {

@Override
public void action() {
ACLMessage Initcoal = new ACLMessage(ACLMessage.CFP);
Capacite AA = new  Capacite(getType(),getEtat(),getLocalite(),getAngle()
,getItrusion(),getPrecision(),getDate1(),getDate2(),getPorte());

try {
        Initcoal.setContentObject((Serializable) AA); /** Objet à envoyer **/
        RechercheDF(getAID(),myAgent); /** Faire une recherche des agents existants **/
        for (int i=0; i<agents.length; i++){
                if(!(myAgent.getAID().equals(agents[i])))
                Initcoal.addReceiver(agents[i]);
        }
} catch (IOException e1) {
        // TODO Auto-generated catch block
        e1.printStackTrace();
} catch (FIPAException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
}
        send(Initcoal);
        System.out.println(" L'Agent " + getLocalName() + " Envoie un Init Coal");
        //doWait(50000);
        addBehaviour(new RecInitCoal());
}
}
private class RecInitCoal extends CyclicBehaviour {
@Override
public void action() {

ACLMessage msg2 = receive(MessageTemplate.MatchPerformative(ACLMessage.CFP));

if (msg2 != null) {

System.out.println(getLocalName() + " A Reçu un Init coal de la part "
+ msg2.getSender().getLocalName());
try {
        Initiateurs.add(msg2.getSender().getLocalName());
        System.out.println("Agent "+ getLocalName()
    +" A Reçu "+((Capacite) msg2.getContentObject()).getPrecision()
```

```
    +" de "+ msg2.getSender().getLocalName());
        vectPrecis.addElement(((Capacite) msg2.getContentObject()).getPrecision());
        vectDate.addElement(((Capacite) msg2.getContentObject()).getDate1());
        vect.add(msg2.getSender().getLocalName());
        vectTypes.add(((Capacite) msg2.getContentObject()).getType());
        if(Iam2 == true){
        Iam2 = false;
        addBehaviour(new WakerBehaviour(myAgent, 10000) {
            public void onWake() {
                System.out.println(getLocalName() + " avec ------> " + Initiateurs);
                        addBehaviour(new envoiAcceptCoal());
            }
                  } );}
} catch (UnreadableException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }}
   else {
                block();}}}
private class envoiAcceptCoal extends OneShotBehaviour {

@Override
public void action() {
// TODO Auto-generated method stub
ACLMessage Accept = new ACLMessage(ACLMessage.INFORM_REF);
Accept.setContent(" J'accept de faire parti de ta coalition ");
for(int i = 0; i < Initiateurs.size(); i++){
Accept.addReceiver(new  AID(Initiateurs.elementAt(i), AID.ISLOCALNAME));
}
System.out.println(" L'agent " + getLocalName()
+ " A Envoyé un ACCEPT Coal aux Initiateurs " + Initiateurs + " NBR AGENT " + NbrAgent);
send(Accept);
addBehaviour(new receAcceptCoal());}
}
private class receAcceptCoal extends CyclicBehaviour {
@Override
public void action() {
// TODO Auto-generated method stub
ACLMessage Acept = receive(MessageTemplate.MatchPerformative(ACLMessage.INFORM_REF));

if (Acept != null) {
if (MembreCoalition == 2){
        if(Iam3 == true){
                Iam3 = false;
addBehaviour(new ACK_coal(Acept.getSender().getLocalName()));
                addBehaviour(new WakerBehaviour(myAgent, 10000) {
                public void onWake() {

            MaxCombine = NbrCombine(vectDate.size(), (MembreCoalition - 1));
                NombreCombine(vectDate.size(), (MembreCoalition - 1),0,L,L2,L3,L4,
        vect,vectDate,vectPrecis,vectTypes,vectDate.size());}} );}}
//}
else {
if ((MembreCoalition > 2) && (MembreCoalition < (NbrAgent + 1))){
                if(Iam3 == true){
                        Iam3 = false;
addBehaviour(new ACK_coal(Acept.getSender().getLocalName()));
        addBehaviour(new WakerBehaviour(myAgent, 10000) {
            public void onWake() {
```

```
        MaxCombine = NbrCombine(vectDate.size(), (MembreCoalition - 1));
                NombreCombine(vectDate.size(), (MembreCoalition - 1)
        ,0,L,L2,L3,L4,vect,vectDate,vectPrecis,vectTypes,vectDate.size());
                                        }
                                    } );
                            //}}}
                else {
                        if (MembreCoalition == (NbrAgent +1)){
                                if(Iam3 == true){
                                        Iam3 = false;
                MembreCoalition = 2;
                NbrRelance = NbrRelance + 1;
                System.out.println (" LE NOMBRE DE RELANCE VA PASSE A ------->"
        + NbrRelance);
                Capacite.setFin("nnok");
                }}}}}
else {
        block();}}}
private void NombreCombine(int n, int p, int k,
Vector<String> L,Vector<String> L2,Vector<Double>
L3,Vector<String> L4,Vector<String> vect, Vector<String> vectDate,
Vector<Double> vectPrecis,
Vector<String> vectType, int r) {
// TODO Auto-generated method stub
int i, j, j1;
Vector<String> t2 = new Vector<String>();
Vector<String> t3 = new Vector<String>();
Vector<Double> t4 = new Vector<Double>();
Vector<String> t5 = new Vector<String>();
Vector<String> Selction = new Vector<String>();
Vector<String> Selction2 = new Vector<String>();
Vector<Double> Selction3 = new Vector<Double>();
Vector<String> Selction4 = new Vector<String>();
if(r<p-k) return;
if(k==p) {
for(i=0;i<p;i++) {
        //System.out.print(L.elementAt(i)+" ");
        Selction.add(i, L.elementAt(i));
        Selction2.add(i,L2.elementAt(i));
        Selction3.add(i,L3.elementAt(i));
        Selction4.add(i,L4.elementAt(i));
}
Selction.add(getLocalName());
Selction2.add(getDate1());
Selction3.add(getPrecision());
Selction4.add(getType());
System.out.println(getLocalName() + " A former une coalition constitué de : " + Selction);
//CalculMesure(Selction,Selction2,Selction3,Selction4,p);
calculePrecision(Selction3, Selction2,Selction,Selction4);
return;
}
for(i=0;i<r;i++) {
L.add(k, vect.elementAt(i));
L2.add(k,vectDate.elementAt(i));
L3.add(k, vectPrecis.elementAt(i));
L4.add(k,vectType.elementAt(i));
for(j=i+1, j1=0;j<r;j++, j1++) {
        t2.add(j1, vect.elementAt(j));
```

```
        t3.add(j1, vectDate.elementAt(j));
        t4.add(j1, vectPrecis.elementAt(j));
        t5.add(j1, vectType.elementAt(j));}
NombreCombine(n, p, k+1, L,L2,L3,L4,t2,t3,t4,t5,j1);}}
private void calculePrecision(Vector<Double> selction3,
Vector<String> selction2, Vector<String> selction, Vector<String> selction4) {
// TODO Auto-generated method stub
compt = compt + 1;

for(int z=0; z <selction3.size(); z++){

P =  P + selction3.elementAt(z);
}
System.out.println(getLocalName() + " l coalition a " + selction + " " + P);
if(M >= P){
M = P;
System.out.println(getLocalName() + " M ET P " + M + " " + P);
NomPermute = (Vector<String>) selction.clone();
DatePermute = (Vector<String>) selction2.clone();
PrecisPermute = (Vector<Double>) selction3.clone();
TypePermute = (Vector<String>) selction4.clone();
//                        System.out.println(getLocalName() +" Changement de coalition "
+ NomPermute +" Avec P = " + M + " COMPTE ET MAXCOMBONE" + compt + " "+ MaxCombine);
//                        System.out.println("\n");
P = 0;
}
else {
if(M == 0){
        M = P;
        NomPermute = (Vector<String>) selction.clone();
        DatePermute = (Vector<String>) selction2.clone();
        PrecisPermute = (Vector<Double>) selction3.clone();
        TypePermute = (Vector<String>) selction4.clone();
        P = 0;
}
else {
        P = 0;
}}
if(compt == MaxCombine){
System.out.println(" ********* La coalition qui gagne est "
+ NomPermute +" avec N = " + M + " Type " + TypePermute +"******");
doWait(20000);
CalculMesure(NomPermute,DatePermute,PrecisPermute,TypePermute);}}
private int NbrCombine(int size, int i) {
// TODO Auto-generated method stub
int a = size;
int b = i;
int c = (size - i);
int resultat;
int fact = 1;
int fact2 = 1;
int fact3 = 1;
for(int k = a; k >= 1; k--){
fact = fact*k;
}
for(int k = b; k >= 1; k--){
fact2 = fact2*k;
}
for(int k = c; k >= 1; k--){
```

```
fact3 = fact3*k;
}
resultat = fact / (fact2 * fact3);
return resultat;
}
private void CalculMesure(Vector<String> selction, Vector<String> selction2,
Vector<Double> selction3, Vector<String> selction4) {
// TODO Auto-generated method stub
Vector<String> Decoupage = new Vector<String>();
Vector<String> Decoupage1 = new Vector<String>();
Vector<String> Comparaison = new Vector<String>();
compt = compt + 1;
for (int i = 0; i <selction2.size(); i++ ){
String Blanc [] = selction2.elementAt(i).split("\\s"); /** Split par rapport a blanc **/
String Slash [] = Blanc [0].split("/");/** Split par rapport au Slash **/
String Points [] = Blanc [1].split(":"); /** Split par rapport à 2 points **/
for(int j = 0; j<Slash.length; j++){
        Decoupage.add(Decoupage.size(), Slash[j]);
}
for(int k = 0; k<Points.length;k++){
        Decoupage.add(Decoupage.size(), Points[k]);
}
Comparaison.addElement(Decoupage.elementAt((7*i)+6));
}
System.out.println(" Decoupage " + Decoupage + " Comparaison " + Comparaison);

String Blanc1 [] = Effet_Reçu.split("\\s");// date reçu
String Slash1 [] = Blanc1 [0].split("/");/** Split par rapport au Slash **/
String Points1 [] = Blanc1 [1].split(":");
for(int j = 0; j<Slash1.length; j++){
Decoupage1.add(Decoupage1.size(), Slash1[j]);
}
for(int k = 0; k<Points1.length;k++){
Decoupage1.add(Decoupage1.size(), Points1[k]);
}
System.out.println(" Decoupage date effet " + Decoupage1);
for(int c = 0; c < Comparaison.size(); c++){

if(Math.abs(Integer.valueOf(Decoupage1.elementAt(6)).intValue() -
Integer.valueOf(Comparaison.elementAt(c)).intValue()) < 300){
        N = N + 1;
        //System.out.println(getLocalName() + " N " + N);

}
else {
if (Math.abs(Integer.valueOf(Decoupage1.elementAt(6)).intValue() -
Integer.valueOf(Comparaison.elementAt(c)).intValue()) > 300){
                N = N + 0;
                //System.out.println(getLocalName() + " N " + N);}}}
addBehaviour(new SendNeg());

}
private class ACK_coal extends OneShotBehaviour {
String destinataire ;
public ACK_coal(String localName) {
destinataire = localName;
}
@Override
public void action() {
```

```
ACLMessage ACK_coal = new ACLMessage(ACLMessage.CONFIRM);
ACK_coal.setPerformative(ACLMessage.CONFIRM);
ACK_coal.addReceiver(new  AID(destinataire , AID.ISLOCALNAME));
send(ACK_coal);
System.out.println("L'Agent Initiateur "+getLocalName() +
"A envoyé un ACK Coal à " + destinataire);
}}
private class SendNeg extends OneShotBehaviour {
@Override
public void action() {
// TODO Auto-generated method stub
ACLMessage neg = new ACLMessage(ACLMessage.PROPAGATE);

try {
        N = N + getIdentifiant();
        //System.out.println(" La coalition " + NomPermute
    +" A une Mesure N = ----------------------> " + N + " "+ getIdentifiant());
        neg.setContentObject(N);
        for (int i=0; i<Initiateurs.size(); i++){
                if(!(myAgent.getAID().getLocalName().equals(Initiateurs.elementAt(i))))
                neg.addReceiver(new  AID(Initiateurs.elementAt(i), AID.ISLOCALNAME));
        }
} catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
}
//doWait(1000);
send(neg);
System.out.println(" L'Agent " + getLocalName() + " A Envoyé le N de sa coalition " + N);
System.out.print("\n");
addBehaviour(new RecNeg());}}
private class RecNeg extends CyclicBehaviour {
@Override
public void action() {
// TODO Auto-generated method stub
ACLMessage msg = receive(MessageTemplate.MatchPerformative(ACLMessage.PROPAGATE));
if(msg != null){
        try {
        PrecisNeg.add(Integer.valueOf
    (msg.getContentObject().toString()).intValue());
    /** Pour la forme et surtt pour la taille **/

if ( N < Integer.valueOf(msg.getContentObject().toString()).intValue()){
        System.out.println(getLocalName() + " OUT AVEC N "
        + Integer.valueOf(msg.getContentObject().toString()).intValue());
        addBehaviour(new Block());
        gagne = 3;
        }
else {
        if ( N > Integer.valueOf(msg.getContentObject().toString()).intValue()){
                Nomsup.add(msg.getSender().getLocalName());
                Precissup.add(Integer.valueOf(msg.getContentObject().toString()).intValue());
                gagne = 1;
                if(PrecionBlokante < N){
                PrecionBlokante = N;
                        }
                }
                else {
                        if ( N == Integer.valueOf(msg.getContentObject().toString()).intValue()){
```

```
                        Nomeg.add(msg.getSender().getLocalName());
                        Preciseg.add(Integer.valueOf(msg.getContentObject().toString()).intValue());
                        gagne = 2;
                        PrecionBlokante = N;}}}
        //doWait(1000);
        if ((Initiateurs.size() == PrecisNeg.size())){

        System.out.println(getLocalName() + " " + Nomeg
    + " " + Preciseg + " " + Nomsup + " " +Precissup
    + " PRECIS BLOK" + PrecionBlokante);

        if(Initiateurs.size() == Nomsup.size()){
                if(getEtat() != null){
                System.out.println(getLocalName()+ " gagne ");
                JOptionPane jop1;
                jop1 = new JOptionPane();
                ImageIcon img = new ImageIcon("inf.png");
                jop1.showMessageDialog(null, "Agent "
        + getLocalName()+" A Gagné  "+ NomPermute, "Information",
        JOptionPane.INFORMATION_MESSAGE, img);
                ACLMessage GO = new ACLMessage(ACLMessage.CANCEL);
                GO.setContent("GO");
                for(int i = 0; i <(NomPermute.size()-1); i++){
                        GO.addReceiver(new  AID(NomPermute.elementAt(i), AID.ISLOCALNAME));
                }
                send(GO);
                addBehaviour(new RecMesure());
                }
                else {

                }}
        else {
                if ((Nomeg.size() > 0) && (gagne !=3) &&(PrecionBlokante <= P)){
                        System.out.println(getLocalName()+ " agal " + Nomeg);
                                doWait(10000);
                                addBehaviour(new SendARG());
                }
                } catch (NumberFormatException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        } catch (UnreadableException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }}
else {
        block();
}}}
private class SendARG extends OneShotBehaviour {
@Override
public void action() {
// TODO Auto-generated method stub
ACLMessage ARG = new ACLMessage(ACLMessage.INFORM_IF);
try {
        if(getEtat() == null){
        somme = 0;
        ARG.setContentObject(0);
        }
        else {
                somme = 1;
```

```
                ARG.setContentObject(1);
        }
        for(int i = 0; i <Nomeg.size(); i++){
                ARG.addReceiver(new  AID(Nomeg.elementAt(i), AID.ISLOCALNAME));
        }
} catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
}
send(ARG);
//doWait(10000);
addBehaviour(new RecARG());
}}
private class RecARG extends CyclicBehaviour {
@Override
public void action() {
// TODO Auto-generated method stub
ACLMessage ARG = receive(MessageTemplate.MatchPerformative(ACLMessage.INFORM_IF));
if(ARG != null){
        try {
                System.out.println("L'Agent " + getLocalName()
        + " A Eté Informé Par " + ARG.getSender().getLocalName()
        + " Qu'il Avait Avec Lui " + ARG.getContentObject() + " Mesures Récentes");
                NbrMesure.add(Integer.valueOf(ARG.getContentObject().toString()).intValue());

                if ( somme < Integer.valueOf(ARG.getContentObject().toString()).intValue()){
                        System.out.println(getLocalName() + " OUT "
                + Integer.valueOf(ARG.getContentObject().toString()).intValue());
                        addBehaviour(new Block());
                        gagne2 = 3;
                }
                else {
                if ( somme > Integer.valueOf(ARG.getContentObject().toString()).intValue()){
                        Nomsup2.add(ARG.getSender().getLocalName());
                        if(SommeBlockante < somme){
                        PrecionBlokante = somme;
                        }
                }
                else {
                        if  (somme == Integer.valueOf(ARG.getContentObject().toString()).intValue()){
                                egal2.add(ARG.getSender().getLocalName());
                                SommeBlockante = somme;}}}
                System.out.println(getLocalName() + " LE NEG ET NBRMESURE"
        + Nomeg.size()+ " " + NbrMesure.size() + " EGALE2" + egal2);
                if(Nomeg.size() == NbrMesure.size()){
        if(Nomeg.size() == Nomsup2.size()){
                JOptionPane jop1;
                jop1 = new JOptionPane();
                ImageIcon img = new ImageIcon("inf.png");
                jop1.showMessageDialog(null, " Agent "+ getLocalName()
        +" A Gagné  "+ NomPermute, "Information", JOptionPane.INFORMATION_MESSAGE, img);
                Capacite.setFin("ok");
                Date date1=new Date();
//              SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss:SSS");
//              System.out.println(" ----------------------> date: "+sdf.format(date1));
                ACLMessage GO = new ACLMessage(ACLMessage.CANCEL);
                GO.setContent("GO");
                for(int i = 0; i < NomPermute.size(); i++){
                        GO.addReceiver(new  AID(NomPermute.elementAt(i), AID.ISLOCALNAME));
```

```
                }
                send(GO);
                Capacite.setFin("ok");
                addBehaviour(new RecMesure());
        }
        else {
                if ((egal2.size() > 0) && (gagne2 !=3) &&(SommeBlockante <= somme)){
                System.out.println(getLocalName()+ " agal2 " + egal2);
                MembreCoalition = MembreCoalition + 1;
                Capacite.setFin("nnok");
                }
                catch (UnreadableException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }
}
else {
        block();}}}
private class Block extends CyclicBehaviour {

@Override
public void action() {
// TODO Auto-generated method stub
ACLMessage msg = receive(MessageTemplate.MatchPerformative(ACLMessage.CANCEL));
if (msg != null){

if(msg.getContent().equalsIgnoreCase("GO")){
        System.out.println(getLocalName()
    + " A Reçu Un Message de Déploiment de l'Initiateur " + msg.getSender().getLocalName());
        addBehaviour(new EnvoiMesure(msg.getSender().getLocalName()));
}
else {
        if(msg.getContent().equalsIgnoreCase("Relance")){
                System.out.println(getLocalName() + " A Reçu Un Message de Relance de l'Initiateur "
                + msg.getSender().getLocalName());
                MembreCoalition = MembreCoalition + 1;
                Capacite.setFin("nnok");
        }}
}else {
block();}}}
private class EnvoiMesure extends OneShotBehaviour {
Vector<Double> s = new Vector<Double>();
Boolean Iam;
String destinataire;
public EnvoiMesure(String localName) {
// TODO Auto-generated constructor stub
destinataire = localName;
}
@Override
public void action() {
// TODO Auto-generated method stub
ACLMessage Env = new ACLMessage(ACLMessage.ACCEPT_PROPOSAL);
if (getEtat() == null){
try {
Env.setContentObject(s);
Env.addReceiver(new  AID(destinataire, AID.ISLOCALNAME));
send(Env);
addBehaviour(new Block());
} catch (IOException e) {
```

```
// TODO Auto-generated catch block
e.printStackTrace();
}}
else {
if(getEtat() != null){
try {
        s.add(PositionX);
        s.add(PositionY);
        s.add(teta);
        s.add(new Double(1));
        Env.setContentObject(s);
        Env.addReceiver(new  AID(destinataire, AID.ISLOCALNAME));
        send(Env);
        addBehaviour(new Block());
} catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();}}}}}
private class RecMesure extends CyclicBehaviour {
Vector<Double> s = new Vector<Double>();
Vector<String> Trame2 = new Vector<String>();
Vector<Double> vectMesure = new Vector<Double>();
boolean Iam = true;
@Override
public void action() {
// TODO Auto-generated method stub
ACLMessage msg = receive(MessageTemplate.MatchPerformative(ACLMessage.ACCEPT_PROPOSAL));
Trame2 = (Vector<String>) Trame.clone();
if(msg != null){
        try {
        System.out.print(getLocalName() + " A reçu "+ msg.getContentObject());
        Q = Q + 1;
        s = (Vector<Double>) ((Vector<Double>) msg.getContentObject()).clone();
        if(!(s.isEmpty())){
                somme2 = somme2 + s.elementAt(s.size() - 1);
                //System.out.println("Q et Somme2" + Q + " " + somme2);
                }
                else {
                        somme2 = somme2 + 0;
                }
        vectMesure.add(teta);
        vectMesure.add(PositionX);
        vectMesure.add(PositionY);
        System.out.println(" Programme " + vectMesure);
} catch (UnreadableException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
        }
}
else {
        block();}}}
private class relance extends OneShotBehaviour {
@Override
public void action() {
// TODO Auto-generated method stub
ACLMessage Relance = new ACLMessage(ACLMessage.CANCEL);
Relance.setContent("Relance");
try {
        RechercheDF(getAID(),myAgent);
        for (int i=0; i<agents.length; i++){
```

```
                if(!(myAgent.getAID().equals(agents[i])))
                Relance.addReceiver(agents[i]);
        }
} catch (FIPAException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
}
send(Relance);
addBehaviour(new SendInitCoal());
}}
public void relance() {
// TODO Auto-generated method stub
this.vectDate.removeAllElements();
// ..... Ajouts des autres méthodes
System.out.println(getLocalName()+ "****************** RELANCE ******************";}}
```

Liste des tableaux

Table des figures

Bibliographie

[Abdallah2004] S. Abdallah and V. Lesser. Organization-based cooperative coalition formation. In *IEEE IAT*, 2004.

[Augusto2008] Juan Carlos Augusto. Ambient intelligence : Basic concepts and applications. In Boris Shishkov Joaquim Filipe and Markus Helfert, editors, *Communications in Computer and Information Science : Software and Data Technologies*, volume 10, Part 1, Berlin, Germany : Springer Berlin Heidelberg, 2008.

[Bahadori2006] S. Bahadori, A. Cesta, L. Iocchi, G.R. Leone, D. Nardi, F. Pecora, and R. Rasconi. Towards ambient intelligence for the domestic care of the elderly. *Ambient Intelligence A Novel Paradigm*, 2006.

[Banatre2007] Michel Banâtre, Ciaràn Bryce, Paul Couderc, and Frédéric Weiss. *Informatique diffuse, des concepts à la réalité*. Hermes Science Publication, Paris, lavoisier edition, 2007.

[Belabbas2007] Aissam Belabbas. *Méthodes de conception de systèmes contrôle/commande reconfigurables en présence d'aléas : application au handicap*. PhD thesis, Laboratoire d'Électronique des systèmes Temps Réel, Université de Bretagne-Sud, 2007.

[Bergamann2007] Ralph Bergamann. Ambient intelligence for decision making in fire service organizations. In *Lecture Notes in Computer Science : Ambient Intelligence*, volume 4794/2007, Berlin, Germany : Springer Berlin Heidelberg, 2007.

[Bermejo2004] Nieto Angeles Bermejo and Noelia Carretero de los Angeles. Inteligencia ambiental. Madrid : Centro de Difusion de Tecnologias CEDIT-FUNDETEL, 2004.

[Berners-Lee2001] Berners-Lee Tim, Hendler James, and Lassila Ora. The semantic web. *Scientific American*, 284(5) :34–43, 2001.

[Brickley2004] Brickley Dan and V. Guha Ramanathan. *Rdf vocabulary description Language 1.0. Rdf schema.* W3C, 2004.

[Brooks1986] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2(1) :14–23, april 1986.

[Brooks1991a] R. A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47(1-3) :139– 159, january 1991.

[Brooks1991b] R. A. Brooks. Intelligence without reason. In Reiter, editor, *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 569–595, Sydney, Australia, 1991. Morgan Kaufmann.

[Broxvall2007] M. Broxvall and W.Y. Kwon B.S. Seo. The peis kernel : A middleware for ubiquitous robotics. In *Workshop on Ubiquitous Robotic Space Design and Applications*, Octobre 2007.

[Castelfranchi1990] C. Castelfranchi. Social power. In Y. Demazeau and J.-P. Müller, editors, *Decentralized AI*, pages 49–62. Elsevier, 1990.

[Chaib-Draa1994] Levesque P. Chaib-Draa B. Hierarchical models and communication in multi-agent environments. In *Proceedings of the 6th European Workshop on Modelling Autonomous Agents and Multi-Agent Worlds*, pages 119–134, Odense, Danemark, 1994.

[Chen2004] Chen Harry, Perich Filip, Finin Tim, and Soupa Anupam, Josh. Standard ontology for ubiquitous and pervasive applications. In *In International Conference on Mobile and Ubiquitous Systems : Networking and Services*, pages 258–267, August 2004.

[Cook2003] J. Cook Diane, Youngblood Michael, O. Heierman Edwin, Gopalratnam Karthik, Rao Sira, Litvin Andrey, and Khawaja Farhan. Mavhome : An agent-based smart home. In *percom : First IEEE International Conference on Pervasive Computing and Communications*, page 521, 2003.

[Cote2006] C. Côté, Y. Brosseau, D. Létourneau, C. Raïevsky, and F. Michaud. Robotic software integration using marie. *The International Journal of Advanced Robotic Systems*, 3(1) :55–60, March 2006.

[Dignum1999] F. Dignum, B. Dunin-Keÿplicz, and R. Verbrugge. Dialogue in team formation : a formal approach. In J.-J. Meyer W. van der Hoek and C. Witteveen, editors, *ESSLLI99 Workshop on the Foundations and Applications of Collective Agent-Based Systems (CABS)*, 1999.

[Dignum2001] F. Dignum, B. Dunin-Keÿplicz, and R. Verbrugge. Agent theory for team formation by dialogue. *Lecture Notes in Computer Science*, pages 150–166, 2001.

[Drevelle2010] V. Drevelle and P. Bonnifait. Robust positioning using relaxed constraint-propagation. In *IROS 2010*, pages 4843–4848, Taipei : Taiwan, Province Of China, 2010.

[Drocourt2002] Cyril Drocourt. *Localisation et modélisation de l'environnement d'un robot mobile par coopération de deux capteurs omnidirectionnels*. PhD thesis, 2002.

[Enderle2001] S. Enderle, H. Utz, S. Sablatng, S. Simon, G. Kraetzschmar, and G. Palm. Miro : Middleware for autonomous mobile robots. *IFAC Conference on Telematics Applications in Automation and Robotics*, 2001.

[Fatiha2007a] Latfi Fatiha, Lefebvre Bernard, and Descheneaux Céline. Le rôle de l'ontologie de la tâche dans un habitat intelligent en télé-santé. 2007.

[Fatiha2007b] Latfi Fatiha, Lefebvre Bernard, and Descheneaux Céline. Habitat intelligent en télésanté : Ontologie de l'équipement. 2007.

[Fensel2001] Fensel Dieter, Harmelen Frank, van, Horrocks Ian, L. McGuiness Deborah, and F. Pattel-Schneider Peter. Oil ; an ontologie infrastructure for the semantic web. *IEEE Intelligent Systems*, 16(2) :38–45, 2001.

[Georgeff1987] Georgeff M. P. and Lansky A. L. Reactive, reasoning and planning. In *AAAI-87*, pages 677–682, Seattle, 1987.

[Gning2006] A. Gning. *Fusion multisensorielle ensembliste par propagation de contraintes sur les intervalles*. PhD thesis, 2006.

[Gray1978] J. Gray. Notes on database operating systems. *Operating Systems : an Advanced Course, Lecture Notes in Computer Science*, 60 :393Ű481, 1978.

[Greaves2000] M. Greaves, H. Holmback, and J. Bradshaw. What is a conversation policy ? In *Issues in Agent Communication*, page 118Ű131, London, UK, 2000. Springer-Verlag.

[Horrocks2002] Horrocks Ian. Daml+oil : A description logic for the semantic web. *IEEE Data Engeeniring Bulletin*, 25(1) :4–9, 2002.

[Huang2009] C. Huang, C. Trappey, A. Trappey, and C. Ku. The design of a jadebased autonomous workflow management system for collaborative soc design. *Expert System*, 36(2) :2659Ű2669, 2009.

[Jaulin1993] L. Jaulin and E. Walter. Set inversion via interval analysis for nonlinear bounded-error estimation. *Automatica*, 29(4) :1053Ű1064, 1993.

[Jaulin2000a] L. Jaulin. *HDR, Le calcul ensembliste par analyse par intervalles*. Université Paris-Sud Orsay, France, 2000.

[Jaulin2002] Jaulin Luc, Kieffer Michel, Walter Eric, and Meizel Dominique. Guaranteed robust nonlinear estimation with application to robot localization. *IEEE Trans. SMC, PartC Applications and Review*, 32(4), 2002.

[Jaulin2009] L. Jaulin. Robust set-membership state estimation ; application to underwater robotics. *Automatica*, 45(1) :202Ű206, 2009.

[Kieffer2000] M. Kieffer, L. Jaulin, E. Walter, and D. Meizel. Robust autonomous robot localization using interval analysis. *Reliable Computing*, 6(3) :337–362, 2000.

[Kivela2002] A. Kivela and Hyvonen. E. ontological theories for the semantic web. 2002.

[Kranz2006] M. Kranz, R. Rusu, A. Maldonado, M. Beetz, and A. Schmidth. A player/stage system for context-aware intelligent environments. *System Support for Ubiquitous Computing Workshop (UbiSys)*, Septembre 2006.

[Kraus1998] S. Kraus, K. Sycara, and A Evenchik. Reaching agreements through argumentation : a logical model and implementation. *Artificial Intelligence*, pages 1–69, 1998.

[Kraus2003] S. Kraus, O. Shehory, and G. Taase. Coalition formation with uncertain heterogeneous information. In *2nd AAMAS*, pages 1–8, 2003.

[Lankri2009] Said Lankri. *Services et Navigation pour Personnes Dépendantes en Environments Domotiques*. PhD thesis, Laboratoire en sciences et techniques de l'information, de la communication et de la connaissance (Lab-STICC) , Université de Bretagne-Sud, 2009.

[Lapique2006] Lapique Francis. Le langage d'ontologie web owl. *Flash informatique*, octobre 2006.

[Lassila2002] Lassila Ora and R. Swick Ralph. *Ressource description framework (rdf) model and syntax specification*. W3C, 2002 edition.

[Léveque1997] D. Meizel O. Lévêque, L. Jaullin and E. Walter. Vehicule localization from inaccurate telemetric dataă : a set of inversion approach. In *IFAC Symposium on robot Control SYROCO 97*, pages 179–186, Nantes, France, 1997.

[Mazouzi2002] H. Mazouzi, A. El Fallah-Seghrouchni, and S. Haddad. Open protocol design for complex interactions in multi-agent systems. In *AAMAS*, pages 517–526, Bologna,Italy, July 2002. ACM.

[Moore1979] R.E. Moore. Method and applications of internal analysis. In *SIAM*, Philadelphia, 1979.

[Moulin1996] Bernard Moulin and B. Chaib-draa. An overview of distributed artificial intelligence. Foundations of Distributed, 1996.

[Nader2008] Mohamed Nader, Al-Jaroodi Jameela, and Imad Jawhar. Middleware for robotics : A survey. In IEEE Intl, editor, *Robotics, Automation, and Mechatronics*, pages 736–742, September 2008.

[Nash1951] J. Nash. *Non-cooperative games. Annals of Mathematics, 286-295.*

[Ortiz2002] C. Ortiz and T. Rauenbusch. Dynamic negotiation. In *AAAI Workshop on Planning with and for Multiagent Systems*, 2002.

[Pareto1896] V. Pareto. *Cours d'économie politique*. Lausanne, 1896.

[Ranganathan2003] Ranganathan Arnand, E. McGrath Robert, H. Campbell Roy, and Mickunas M. Dennis. Use of ontologies in a pervasive computing environnement. *Knowledge Engineering Review*, 18(3) :209–220, 2003.

[Rao1991] Anand S. Rao and Michael P. Georgeff. Modeling rational agents within a bdi-architecture. In Richard Fikes James Allen and Eric Sandewall, editors, *Second International Conference on Principles of Knowledge Representation and Reasoning (KR-91)*, pages 473–484. Morgan Kaufmann, April 1991.

[Remahnino2006] P. Remahnino, H. Hagras, N. Monekoss, and S. Velastin. Ambient intelligence a gentle introduction. *Ambient Intelligence A Novel Paradigm*, 2006.

[Reynet2009] L. Jaulin O. Reynet and G. Chabert. Robust tdoa passive location using interval analysis and contractor programming. In *Radar*, Bordeaux, France, october 2009.

[Safiotti2008] Alessandro Saffiotti, Mathias Broxvall, Beom-Su Seo, and Young-Jo Cho. The peis-ecology project : a progress report. Technical report, Electronics and Telecommunications Research Institute, Gajeong-dong, Yuseong-gu, Daejeon, Korea, 2008.

[Sandholm1999] T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohmé. Coalition structure generation with worst case guarantees. *Artificial Intelligence*, Vol. 111(1-2) :209–238, 1999.

[Scully2004] T. Scully, M. Madden, and G. Lyons. Coalition calculation in a dynamic agent environment. In *the 21st ICML*, 2004.

[Sen2000] S. Sen and P. Dutta. Searching for the optimal coalition structure. In *the 3rd CEECMAS*, pages 286–292, 2000.

[Shapley1953] L. Shapley. A value for n-person game. In H. Kuhn and editors A. Tucker, editors, *Contribution to the Theory of Games*. Princeton University Press, 1953.

[Shehory1998] O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, Vol. 101(1-2) :165–200, 1998.

[Shehory1999] O. Shehory and S. Kraus. Feasible formation of coalitions among autonomous agents in non-superadditive environments. *Computational Intelligence*, Vol. 15(3) :218–251, 1999.

[Shoham1992] Shoham Yoav. Agent oriented programming. *Artificial Intelligence*, 60 :51–92, february 1992.

[Sims2003] M. Sims, C. Goldman, and V. Lesser. Selforganization through bottom-up coalition formation. In *the 2nd AAMAS*, 2003.

[Smith1980] R. Smith. The contract net protocol : high-level communication and control in a distributed problem solver. In *IEEE Transactions on computers*, pages 1104–1113, 1980.

[Smith2001] B Smith and C Welty. Fois introduction : Ontology-towards a new synthesis. In *International Conference on Formal Ontology in Information Systems*, pages 3 – 9, Ogunquit, Maine, USA, 2001. ACM.

[Soh2001] L.-K. Soh and C. Tsatsoulis. Reflective negotiating agents for real-time multisensor target tracking. In *IJCAI'01*, 2001.

[Soh2002] L.-K. Soh and C. Tsatsoulis. Allocation algorithms in dynamic negotiation-based coalition formation. In *AAMAS02 Workshop 7 "Teamwork and coalition formation"*, pages 16–23, 2002.

[Soh2003] L.-K. Soh and C. Tsatsoulis. Utility-based multiagent coalition formation with incomplete information and time constraints. In *IEEE SMC*, 2003.

[Songmin2007] Jia Songmin, Gakuhari Harunori, and Takase Kunikatsu. Network distributed monitoring system for home service robots based on rt middleware. *Advanced Computational Intelligence and Intelligent Informatics*, 11(7), 2007.

[Stanford2005] Stanford Université. *The Protégé Ontology Editor and Knowledge Acquisition System.* Université de Stanford, juin 2005.

[Tapus2010] Adriana Tapus, J. Matariťc Maja, and Scassellati Brian. The grand challenges in socially assistive robotics. *IEEE Robotics and Automation Magazine Special Issue On Grand Challenges in Robotics*, 2010.

[Tardieu2004] Samuel Tardieu. *Systèmes répartis.* ENST, 2004.

[Uschold1996] Mike Uschold and Michael Gruninger. Ontologies : principles, methods, and applications. Knowledge engineering review, 1996.

[Verrons2004] M.-H. Verrons. *GeNCA : Un modèle général de négociation de contrats entre agents.* PhD thesis, Université des Sciences et Technologies de Lille, Lille, Novembre 2004.

[Wang2004] Wang Xiao, Hang, Zhang Da, Qing, Gu Tao, and Pung Hung, Keng. Ontology based context modeling and reasoning using a owl. In *In PERCOMW '04, Second IEEE Annual Conference on Pervasive Computing and Communication Workshops*, page 18, Washington DC, USA, 2004. IEEE Computer Society.

[Weiser1993] Marc Weiser. *Hot topics : Ubiquitous computing.* IEEE Computer, October 1993.

[Wooldridge2009] Michael Wooldridge and Nicholas R. Jennings. Agent theories, architectures, and languages : a survey. In Wooldridge and Jennings Eds, editors, *Intelligent Agents*, pages 1–22, Berlin : Springer-Verlag, 2009.

[Zahia2009] Zahia Guessoum and Salima Hassas. *Systèmes multi-agents, Génie logiciel multi-agents.* octobre 2009.

Assistance robotisée à la personne en environnement coopérant

La rencontre de l'intelligence ambiante et de la robotique d'assistance a donné naissance à ce que nous avons appelé robotique ambiante d'assistance car destinée à assister une personne en perte d'autonomie. Elle s'appuie sur l'existence de réseaux d'objets communicants présents dans l'environnement de la personne pour décliner un ensemble de services et de téléservices destinés à faciliter la vie quotidienne de cette personne et de son entourage. Un, voir plusieurs robots peuvent être présents dans cet environnement.
Une communauté scientifique récente s'est construite autour de la robotique ubiquitaire. Tandis que les robots des générations précédentes ont été conçus pour réaliser des tâches spécifiques et construits en tant qu'unité indépendante, la nouvelle génération vise l'ubiquité. L'autonomie du robot est, dans ce cadre, obtenue par une interaction étroite entre le robot et l'environnement ambiant communicant. Jusqu'à ces dernières années le robot évoluait dans un environnement plutôt hostile qui ne lui facilitait pas la tâche. Dans le contexte de l'intelligence ambiante, les objets communicants de l'environnement peuvent jouer un rôle "facilitateur" en aidant le robot à se localiser, naviguer, rechercher un objet. Inversement, le robot peut être vu comme un objet communicant qui est mis à contribution par des services autres que l'assistance à la personne en perte d'autonomie.

Le premier objectif de cette thèse est de proposer une architecture informatique permettant la coopération entre le robot et les objets communicants présents dans l'environnement tels que les capteurs ou les actionneurs. Une des difficultés est que le robot évolue dans un environnement encombré, changeant de façon non prédictible au sens où, par exemple, les capteurs ambiants peuvent être disponibles/indisponibles, accessibles/inaccessibles. Il est donc nécessaire de disposer de mécanismes d'adaptation face à des situations diverses voir imprévues afin qu'au final, le robot réussisse sa mission, se déplacer d'un point A à un point B, si possible en tenant compte de contraintes telles que le degré d'urgence ou le niveau de précision. Nous avons aussi pris en compte une autre contrainte, d'ordre éthique, dimension très débattue dans les projets de robotique d'assistance à domicile, qui est le degré d'intrusion. L'intrusion d'une intelligence ambiante capable d'agir et de percevoir, en présence de la personne, occasionne une gêne pour celle-ci et son entourage. Dans le cadre de cette thèse, nous évaluerons notre architecture informatique de coopération entre le robot et le réseau de capteurs, ainsi que les mécanismes d'adaptation avec la tâche de localisation du robot. L'ambition est de fournir un résultat de localisation à tout prix, quitte à ce qu'il soit dégradé et ce, quelle que soit la situation rencontrée. Pour répondre à cette obligation de résultat, nous avons introduit la notion d'effet. Le système cherche à se rapprocher le plus près possible de l'effet demandé. C'est la solution acceptable, au sens de certains critères, qui est retenue. Ce n'est pas forcément la solution optimale.

Le deuxième objectif de la thèse est de proposer une méthode de localisation par coopération robot-environnement communicant. Notre ambition est de localiser avec certitude le robot dans une zone de l'habitat. Les dimensions de cette zone peuvent être très variables. Ces dimensions dépendent des besoins de la tâche, voire de l'enchaînement de tâches, que prévoit de réaliser le robot à cet instant, mais découlent aussi de facteurs comme la dispersion des capteurs dans l'environnement, la qualité des mesures disponibles. Nous partons du principe qu'une bonne connaissance de sa situation, la certitude d'être dans cette zone avec une précision connue, permettra au robot d'adopter la meilleure stratégie possible pour assurer sa mission qui est de retrouver la personne dans son habitat. Aussi, la position et l'orientation du robot doivent être données avec une précision garantie. C'est moins la précision que la certitude que le robot appartienne à cette zone qui importe. Bien que l'approche de localisation ait pour ambition d'utiliser les mesures provenant d'un réseau de capteurs, les plus variés possibles, situés dans l'environnement ou sur le robot, elle n'a pas vocation, du moins dans sa version de base, à répondre à tous les cas de figure. Nous allons préciser le périmètre de son utilisation.
L'un des intérêts de cette méthode est d'être apte à utiliser des mesures issues de capteurs du robot et de l'environnement. Les mesures sont des angles connus avec une certaine incertitude. La localisation est obtenue par multilatération.

La version de l'algorithme proposé dans cette thèse a pour vocation d'une part, de montrer l'intérêt de l'approche par intervalles, et d'autre part, de disposer d'une méthode de localisation qui sera utilisée par les agents capteurs du SMA décrit plus haut. Des améliorations peuvent être apportées sur plusieurs points. Tout d'abord, les mesures aberrantes qui violent de l'hypothèse d'erreur bornée ne sont pas traitées. Il existe des pistes dans la littérature pour y remédier. Par ailleurs, il a été considéré, pour simplifier la présentation de l'approche, que la localisation des capteurs et des marqueurs de l'environnement était connue. Mais il est facile d'introduire cette incertitude dans l'algorithme.

Mots-clés : Assistance ambiante, Robot, Objets communicants, Localisation, Architecture informatique, Coalitions, Effet souhaité, SMA, Adaptation, Degré d'urgence, Degré d'intrusion.

Robotic assistance to the personne in ambient intelligence

Abstract

The meeting of ambient intelligence and robotics support gave resulting in the so-called ambient robotic assistance, designed to assist a person with reduced autonomy. It relies on the existence of communicating objects networks in the environment of the person to decline a set of services and teleservices to facilitate the daily life of that person and his entourage. One or more robots may be present in this environment. A recent scientific community was built around the ubiquitous robotics. While previous generations of robots are designed to perform specific tasks and built as an independent unit, the new generation is ubiquitous. The autonomy of the robot is, in this context, obtained by a close interaction between the robot and communicating ambient environment. Until recent years the robot progress in a hostile environment that did not facilitate the task. In the context of ambient intelligence, communicating objects in the environment can play a "facilitator" in helping the robot to locate, to browse and to search object, and so on. Conversely, the robot can be seen as a communicating object that is put to use by other services than person with reduced autonomy assistance.

The first objective of this thesis is to propose computer architecture for the robot and communicating objects cooperation, in the environment such as sensors or actuators. One difficulty is that the robot is moving in a saturated environment, which changes so unpredictably in the sense that, for example, ambient sensors may be available or unavailable, accessible or inaccessible. It is therefore necessary to have mechanisms to adapt to different or unexpected situations to permit the robot succeeded in his mission, to move from point A to point B, where possible, taking account constraints such as degree of urgency or level of accuracy. We also considered another ethical constraint in the robotics projects home assistance, which is the degree of intrusion. The intrusion of ambient intelligence capable of acting and perceiving, in the presence of the person, causes discomfort for it and his entourage. As part of this thesis, we evaluate our computer architecture for the robot and communicating objects cooperation, as well as coping mechanisms with robot localization. The aim is to provide localization result, even if it is either degraded and this, regardless of the situation. To meet this result, we introduced the concept of effect. The system seeks to approximate as closely as possible the requested effect. This is the acceptable solution within the meaning of certain criteria, which is used. This is not necessarily the optimal solution. The contributions of this first part are as follows. We propose robot-environment cooperative architecture consists of a multi-agent system (MAS) based on a knowledge base and a gateway that interacts with the ambient environment including communicating objects which is the robot and with a user interface. Our contributions concern MAS structure and of the knowledge base based on ontology's formalism. To address the problem of adaptation to the context on the one hand, and respect the level of discomfort of the user (intrusiveness MAS) on the other hand, we propose a solution based on training coalitions of agents AA-CRE (ambient agents for robot-environment cooperation) according to the desired effect.

The second goal of this thesis is to propose robot-environment communicator tracking method. Our objective is to locate with certainty the robot in an area of habitat. The dimensions of this zone can vary widely. These dimensions depend on the needs of the task, even tasks sequence that the robot expects to achieve at this time, but also stem from factors such as the dispersal of sensors in the environment, quality of available measures. We assume that with a good knowledge of the situation, the certainty of being in this area with a known accuracy, will allow the robot to adopt the best strategy to ensure its mission which is to find the person in its habitat. Also, the position and orientation of the robot must be given with guaranteed accuracy. It is less accurate than the certainty that the robot belongs to the area that matters. Although the approach of localization is based on the ambition to use the measurements from a sensor network, the most varied possible in the environment or on the robot, it is not intended, at least in its basic answer to all situations. We will specify the scope of its use. One of the advantages of this method is to be able to use measurements from sensors of the robot and the environment. The measures are known angles with some uncertainty. The localization is obtained by multilateration. The contributions of this second part are as follows. We propose a localization method based on the principle of multiangulation. Solving the problem is obtained by set theoretic approach, specifically the interval analysis. As the size of the state vector is less than or equal to three, the calculations are relatively simple and quick (the simulation is performed in Matlab, a realistic assessment should be performed to estimate the execution time). The simple model chosen for the sensors makes it possible to integrate a wide variety of sensors, the crudest to the most complex, if it delivers a corner measurement. The version of the algorithm proposed in this thesis aims on the one hand, to show the interest of the intervals approach, and secondly, to have a localization method which will be used by sensor agents of SMA described above. Improvements can be made on several points. First, the outliers that violate the assumption of error bounded are not addressed. There are tracks in the literature about it. Moreover, it was considered, to simplify the presentation of the approach, that the localization of sensors and markers of the environment was known. But it is easy to introduce uncertainty in the algorithm.

Key-words : Ambient Intelligence, Robot, Localization, MAS, desired effect, multiangulation, cooperative architecture.