Supervision pour un robot interactif: action et interaction pour un robot autonome en environnement humain

Aurélie Clodic

Université Paul Sabatier – Toulouse III

École doctorale Systèmes Informatiques - Toulouse
UFR PCA

Supervision pour un robot interactif : Action et Interaction pour un robot autonome en environnement humain

THÈSE

présentée et soutenue publiquement le Mercredi 27 juin 2007

pour l'obtention du

Doctorat de l'Université de Toulouse
(spécialité informatique)

par

Aurélie Clodic

Sous la direction de Rachid Alami et Raja Chatila

Composition du jury

Président :	Régine André-Obrecht
Rapporteurs :	François Charpillet Peter Ford Dominey
Directeur de thèse :	Rachid Alami

Laboratoire d'Analyse et d'Architecture des Systèmes — UPR 8001 CNRS

Mis en page avec la classe thloria.

Résumé

Nos travaux portent sur la supervision d'un robot autonome en environnement humain et plus particulièrement sur la prise en compte de l'interaction homme-robot au niveau décisionnel. Dans ce cadre, l'homme et le robot constituent un système dans lequel ils partagent un espace commun et échangent des informations à travers différentes modalités. L'interaction peut intervenir soit sur une requête explicite de l'homme, soit parce que le robot l'a estimée utile et en a pris l'initiative. Dans les deux cas le robot doit agir afin de satisfaire un but en prenant en compte de manière explicite la présence et les préferences de son partenaire humain.

Pour cela, nous avons conçu et développé un système de supervision nommé Shary qui permet de gérer des tâches individuelles (où seul le robot est impliqué) et des tâches jointes (où le robot et un autre agent sont impliqués). Ce système se compose d'une mécanique d'exécution de tâches gérant d'une part la communication nécessaire au sein d'une tâche et d'autre part l'affinement de la tâche en sous-tâches. Pour cela chaque tâche est définie par un plan de communication et un plan de réalisation auxquels sont associés des moniteurs de suivi.

Nous avons développé ce système sur le robot Rackham dans le cadre d'une tâche de "Guide de musée". Nous avons également utilisé ce système dans le cadre d'une tâche du type : "apporter quelque chose à quelqu'un" sur le robot Jido pour montrer la généricité du système.

Mots-clés: interaction homme-robot, supervision de tâches robotiques

Abstract

Human-Robot collaborative task achievement requires specific task supervision and execution. In order to close the loop with their human partners robots must maintain an interaction stream in order to communicate their own intentions and beliefs and to monitor the activity of their human partner. In this work we introduce SHARY, a supervisor dedicated to collaborative task achievement in the human robot interaction context.

The system deals for one part with task refinement and on the other part with communication needed in the human-robot interaction context. To this end, each task is defined at a communication level and at an execution level.

This system has been developped on the robot Rackham for a tour-guide demonstration and has then be used on the robot Jido for a task of fetch and carry to demonstrate system genericity.

Keywords: human-robot interaction, supervision

Remerciements

Il est venu le temps d'enlever l'onglet version provisoire de cette thèse et qu'elle devienne un manuscrit auquel il faut mettre un point final. Cette thèse a été riche de projets et de rencontres, je souhaite remercier toutes les personnes qui m'ont permis de la réaliser.

Je tiens tout d'abord à remercier Raja Chatila, aujourd'hui directeur du laboratoire, sans qui rien de tout cela ne serait arrivé. Cela fait 5 ans je rentrais dans ton bureau Raja et après de savants calculs sur de petits papiers tu me disais : "ok, si tu veux tu viens en thèse chez nous". Je n'ai jamais compris pourquoi moi mais j'espère ne pas t'avoir déçu. Bien entendu merci à Rachid Alami d'avoir encadré mes travaux. Merci Rachid pour ton éternelle enthousiasme.

Je remercie les membres de mon Jury : Régine André-Obrecht, Peter Dominey et François Charpillet pour leur présence et leur intérêt pour mon travail.

Merci à tout le personnel du LAAS pour leur gentillesse et leur disponibilité. Merci également au personnel d'Avenance pour leurs sourires (et leurs délicates attentions lors de ces derniers mois).

Merci à tout mes collègues du groupe devenu pôle RIA du LAAS en espérant vous voir rester "unis dans la diversité". Merci à notre très chère Jackie pour son sourire et sa gentillesse. Merci à tous les thésards, stagiaires, permanents qui n'ont jamais démérités pour tenter de et réussir à faire marcher les robots (et merci aussi bien entendu à toute l'équipe robots-admin pour cela). Merci à la laasko-team@laas et à la Cité de l'espace et spécialement à Max et à Stéphan, les "gardiens" de Rackham. Merci à la cogniron@laas team (CHOCOLATINES forever ;-)) ainsi qu'aux sympathiques partenaires rencontrés lors du projet. Merci à Maxime pour tout le travail réalisé sur Shary.

Une dédicace spéciale à la Piltrafa Team et consorts : Thierry (La Cérémonie), Efraïn, Nacho (Le mexicain qui prend des photos allongé par terre), Léonardo, Luis, Claudia et Jib, Akin (What a Wonderful World), Jean-Michel (l'estomac sur pattes), Martial, Jérôme (petit bonhomme), Sylvain, Gaby, Felipe et Candy, Gustavo, Abed , Titine et Lolotte,... pour tout ces bons moments.

Gros Bisous aux colloc's Gaelle (Syntagma Vivra ! Vive Syntagma ! ! !) et Sara pour leur incroyable energie et pour les indispensables parties de rigolades pour décompresser.

D'autres personnes ont traversés ces quelques années à mes côtés. Une grande pensée pour Jérôme, dont le chemin s'est trop brutalement arrêté, ton appétit de vivre restera toujours pour moi un grand exemple, tu nous manques. Un coucou spécial pour Olivier, Anne-Laure et Axel, Benoit, Angélique et Jean-Yves. Un grand bonjour Vaudois à Sebastien, JSTLPT. Je n'oublie pas bien entendu "mes" Bretonnes et leurs compagnons avec un gros bisous (salé) spécial à Karine. Et puis bien entendu la bande des SI, Jean-Marc, Lilian, JC et Laeti ainsi qu'Olivier, Gee, Hugo et Laurent.

Une dédicace également à ma belle famille, merci de m'avoir accueilli si gentiment parmi vous.

Un grand merci à ma famille. A ma tribu H d'être toujours là. A mon grand frère Christophe et sa petite famille et mon "petit" frère Guillaume de savoir supporter leur petite soeur. A mes parents, pardon d'être aussi loin, mais je ne vous oublie pas et je vous aime.

Et enfin et surtout, un immense merci à Julien de me comprendre, de me soutenir et de m'aimer tout simplement et un gros bisous à Tom, le joli soleil de notre amour.

A mes parents.
A Jérôme, qui ne sera plus jamais là, qui sera toujours là.

Introduction Générale

Contexte

Notre travail a été effectué dans le cadre du projet européen Cogniron[1] : "The Cognitive Robot Companion". Les objectifs généraux du projet Cogniron sont d'étudier les capacités de perception, de représentation, de raisonnement et d'apprentissage nécessaires à un robot évoluant dans un environnement humain.

Dans ce contexte, nous nous sommes intéressés aux capacités d'interaction et de collaboration avec l'homme que le robot doit exhiber et mettre en oeuvre. Pour cela nous avonns développé une architecture de contrôle et un système de supervision et d'exécutions de tâches collaboratives prenant en compte non seulement la réalisationd de la tâche mais également la manière de la réaliser et la commmunication nécessaire à son déroulement.

Robots et scénarios

Pour concrétiser et contextualiser nos développements nous avons réalisé plusieurs démonstrations. Nous allons décrire ici brièvement les robots et les scénarios utilisés pour notre étude. Une description plus précise est disponible dans la partie Expérimentations page 127.

Rackham, le "Guide de musée"

Rackham est le robot présenté figure 1.

Pour étudier l'interaction homme-robot, nous devions trouver un environnement d'expérimentation, loin du laboratoire et de ses chercheurs qui savent exactement comment fonctionne le robot. Nous avons trouvé cet environnement au sein de l'exposition Mission BioSpace qui a eu lieu à la Cité de l'Espace à Toulouse

Sur une période de deux ans, effectuant des séjours de 2 semaines par trimestre, Rackham a guidé les gens au sein de l'exposition. Une mission basique de Rackham consiste à détecter une personne, se présenter : "Je suis Rackham et je peux vous guider au sein de l'exposition", lui proposer de l'amener à un endroit de l'exposition : "Cliquez sur la carte pour choisir une destination". Une fois la destination choisie par le visiteur, le robot calcule sa trajectoire, l'affiche et la réalise en s'assurant que la personne le suive et en réagissant de manière appropriée si çà n'est pas le cas. De plus des informations sont données à l'utilisateur au cours de la navigation à l'arrivée de certains événements : blocage du robot, besoin d'une relocalisation, etc.

[1] http ://www.cogniron.org

FIG. 1 – Rackham

Jido, le robot de service

Jido (cf figure 2) est un robot équipé d'un bras manipulateur. Il est utilisé pour une expérimentation du projet Cogniron dont le scénario consiste à détecter les éventuels besoin d'un homme assis dans la pièce où se trouve le robot. Le robot détectera par exemple que l'homme a besoin de boire. Il ira lui demander confirmation et si le besoin est avéré, le robot ramènera une bouteille à l'homme. Au sein de ce scénario, nous nous intéresserons plus précisément à la tâche consistant à donner un objet à l'homme.

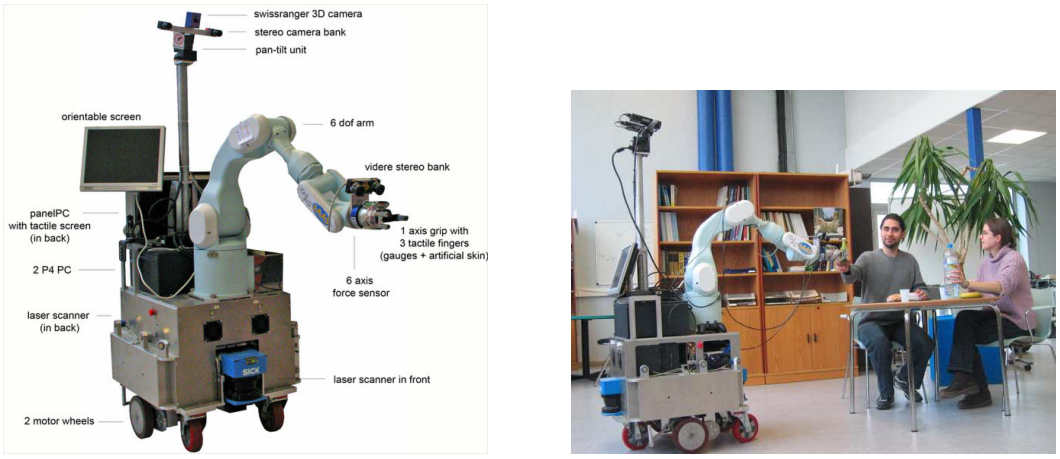


FIG. 2 – Jido

Plan du manuscrit

Ce manuscrit se découpe en trois parties.

Dans une première partie, nous présenterons l'état de l'art en nous focalisant d'une part sur les théories de la coopération et d'autre part sur des systèmes les implémentant. Nous insisterons notamment sur la notion de tâche coopérative et sur les capacités décisionnelles nécessaires au robot pour interagir avec l'homme de manière pertinente et pour produire un comportement compréhensible et acceptable par l'homme.

Dans dans une seconde partie, nous présenterons nos contributions en présentant l'architecture de contrôle pour un robot interactif que nous avons défini au sein de l'architecture LAAS. Cette architecture définit différents composants permettant au robot d'avoir une vue d'ensemble des buts qu'il doit accomplir, de modéliser les hommes avec qui il va interagir et les données qui lui sont nécessaires, ainsi que la prise en compte de plans d'actions et de communications qui serviront à une mécanique d'exécution et de supervision de tâches. Nous expliquerons ensuite plus précisément cette mécanique d'exécution : Shary ("Supervision for Human Adapted Robot I(Y)nteraction"), le système de supervision et d'exécution de tâches que nous avons développé. Ce système couvre l'établissement d'un but commun entre le robot et l'homme, l'affinement des tâches qui serviront à l'accomplir mais aussi l'établissement et le maintien de la communication entre l'homme et le robot pour réaliser ce but. Enfin, nous présenterons l'analyse que nous avons effectuée concernant la modélisation de la communication et notamment des croyances nécessaires à l'établissement d'une croyance partagée entre l'homme et le robot. Nous poursuivrons en étudiant le lien possible entre notre système et un système de dialogue.

Dans une dernière partie, nous présenterons les expérimentations. D'une part les démonstrations effectuées avec le robot Rackham à la Cité de l'Espace qui nous ont permis d'étudier le contexte de l'interaction homme-robot et ont été à la base de la définition de notre système. Ensuite, l'utilisation de notre système pour la réalisation d'une tâche consistant à donner un objet à une personne par le robot manipulateur Jido, montrant par là la généricité et les possibilités du système que nous avons défini.

Nous conclurons ensuite notre travail.

Table des matières

Partie I Etat de l'art

Chapitre 2
Les systèmes

Conclusion

Partie II Un cadre pour la réalisation de tâches collaboratives en robotique

Partie III Expérimentations

Introduction

Chapitre 6
Robots et Fonctionnalités

Chapitre 7
Rackham à la Cité de l'espace

Annexes

Annexe A
Définitions

Annexes

Annexe B
Interaction entre le dialogue et la décision pour la réalisation de tâches collaboratives

Première partie

Etat de l'art

Introduction

Dans [NF05], Nourbakhsh et Fong proposent un challenge intéressant pour la réalisation de l'interaction homme-robot, il s'agit de faire en sorte que l'homme et le robot se comprennent et soient prévisibles l'un pour l'autre : *"of primary interest is making the human and robot understandable and predictable to each other".*

Dans ce cadre, ils considèrent qu'il faut doter les robots de mécanismes de raisonnement et de représentations similaires à ceux utilisés par les hommes. Le "perspective taking" va être une de ces capacités, elle consiste à permettre au robot de prendre la perspective de l'homme.

Breazeal [Bre03] anticipait ces idées en définissant les robots "sociaux" (*social robots*) comme ceux pour lesquels les gens appliquent un modèle social dans le but d'interagir avec eux et de les comprendre. C'est à cette catégorie de robot que nous nous intéressons.

Ainsi, les robots autonomes perçoivent leurs environnement, prennent des décisions et exécutent des actions coordonnées pour accomplir leurs tâches. Ce que l'on souhaite ajouter à ces capacités ce sont les moyens de communiquer, de coopérer, d'apprendre des autres.

Notre présentation de l'état de l'art ne va pas porter sur des robots en particulier mais sur les théories de la coopération et sur des systèmes les implémentant.

Cette partie se compose de deux chapitres. Dans un premier chapitre nous allons étudier les théories de la coopération qui nous inspiré : principalement la théorie de l'intention jointe, Shared Plans et la théorie de l'action jointe. Dans le second chapitre nous étudierons des systèmes : le Teamwork, Collagen, HRI/OS et des travaux liant l'apprentissage et la collaboration. Nous conclurons sur les éléments qui ont retenu notre attention.

Chapitre 1

Les théories de la coopération

Dans ce chapitre, nous allons présenter les théories de la coopération qui nous ont inspiré : la théorie de l'intention jointe, SharedPlans et la théorie de l'action jointe. Nous finirons par des remarques plus générales effectuées à partir de ces théories.

1.1 La théorie de l'intention jointe

La théorie de l'intention jointe a été développée dans les années 1990 par Hector J. Levesque et Philip R. Cohen. Les articles ayant servi à notre étude sont les suivants : [LCN90], [CLS98], ainsi que [CL90], [CL91] dans une moindre mesure.

Dans un premier temps nous allons expliquer leurs définitions de l'engagement et de l'intention individuelle[2]. Ensuite nous expliquerons les notions d'engagement et d'intention jointe. Enfin nous introduirons la notion d'actes de communication et ses représentations possibles, puis nous conclurons.

1.1.1 Cas individuel

Engagement Individuel

L'engagement individuel (*individual commitment*) d'un agent x pour réaliser un but p sous la condition q se définit comme un but persistant (*persistent goal*) :

$$\begin{aligned}(PGOAL\ x\ p\ q) \triangleq\ & (BEL\ x\ \neg p)\ \wedge\ (GOAL\ x\ \Diamond p)\ \wedge \\ & (UNTIL\ [(BEL\ x\ p)\ \vee\ (BEL\ x\ \Box\neg p)\ \vee (BEL\ x\ \neg q)]\ (GOAL\ x\ \Diamond p))\end{aligned}$$

i.e. un agent x possède un but persistant $(PGOAL\ x\ p\ q)$ si x veut que p devienne vrai et il ne peut abandonner ce but à moins que p soit réalisé ou devienne impossible ou non-pertinent (i.e la condition q devient fausse).

La condition q représente une condition d'arrêt (ou de perte de pertinence) cela permet à l'agent de se dégager du but pour une condition (possiblement un ensemble de conditions). Cette condition peut par exemple encoder la ou les raisons qui ont fait que l'agent s'est engagé pour le

[2] N.B. :
- la plupart des notations de ce chapitre sont tirées de [KHC02] mais sont cohérentes avec les définitions de Cohen et Levesque,
- les définitions des notations mathématiques (e.g. $\Diamond$, $\Box$) peuvent être trouvé en annexe A.

but. Si cette ou ces conditions ne sont plus réunies, il est normal que l'agent abandonne le but (e.g. la condition q peut permettre de désactiver un sous-but si le but n'est plus pertinent).

De cette définition on peut retenir :
- une fois qu'il s'est engagé, l'agent ne peut laisser tomber son engagement que dans les conditions particulières qui ont été définies,
- les autres buts et engagements de l'agent doivent être cohérents avec ces conditions,
- si une première tentative de réalisation échoue, l'agent devra persévérer tant que l'une des conditions n'est pas satisfaite.

Intention individuelle

Une intention peut être définie comme un engagement d'agir dans un certain état d'esprit :

$$(INTEND\ x\ a\ q) \triangleq (PGOAL\ x\ (DONE\ x\ [UNTIL\ (DONE\ x\ a)\ (BEL\ x\ (DOING\ x\ a))]?; a)\ q)$$

Un agent à l'intention de faire une action sous certaines conditions si il a comme but persistant de faire cette action et qu'une fois qu'elle est réalisée, il a la croyance de l'avoir réalisée.

1.1.2 Engagement joint

Cohen et Levesque font remarquer qu'une action jointe ne peut se résumer à une collection d'actions individuelles, même si elles sont coordonnées et/ou réalisées de façon simultanée. Des agents peuvent agir de manière coordonnée sans pour autant agir ensemble et inversement des agents peuvent agir ensemble sans qu'ils aient à le faire de manière coordonnée. Ils font par exemple remarquer que dans le trafic automobile, les conducteurs agissent simultanément et de manière coordonnée en respectant le code de la route mais qu'on ne peut alors pas parler de travail "en équipe". Ce qui va distinguer une action jointe ou action faite en collaboration ou en commun d'une simple action coordonnée, c'est l'état mental commun des participants.

> "without a shared mental state a team does not exist"
> *"sans un état mental commun, une équipe n'existe pas"*

Définition des croyances

Pour définir cet état mental commun, il faut alors définir les notions de croyance mutuelle, pour cela nous adoptons les définitions et les notations de [KHC02] :

Unilateral Mutual Belief : croyance mutuelle unilatérale

$$(BMB\ x\ y\ p) \triangleq (BEL\ x\ p \wedge (BMB\ y\ x\ p))$$

Mutual Belief : croyance mutuelle

$$(MB\ x\ y\ p) \triangleq (BMB\ x\ y\ p) \wedge (BMB\ y\ x\ p)$$

Mutual Knowledge : connaissance mutuelle

$$(MK\ x\ y\ p) \triangleq p \wedge (MB\ x\ y\ p)$$

Mutual Goal : but mutuel

$$(MG\ x\ y\ p) \quad \triangleq \quad (MB\ x\ y\ (GOAL\ x \Diamond\ p)\ \wedge\ (GOAL\ y \Diamond\ p))$$

Les concepts de croyance mutuelle[3] (mutual belief) et de croyance mutuelle unilatérale (unilateral mutual belief) est représentée par une conjonction infinie de croyances à propos des croyances des autres agents (i.e. une conjonction des croyances à propos des autres agents à propos des croyances des autres agents (etc, à tous les niveaux)) concernant une même proposition.

Engagement en commun

L'un des principaux arguments de Cohen et Levesque est que dans le cas d'un engagement en commun, on se retrouve en présence des croyances de chacun des agents impliqués et que ces croyances peuvent diverger ce qui peut mettre l'équipe en défaut. Par exemple, un membre de l'équipe peut s'apercevoir que le but est impossible à atteindre et abandonner le but, alors l'équipe toute entière devrait abandonner le but mais l'équipe n'a pas forcément les connaissances pour le faire. Lorsqu'il acquiert une nouvelle information, un membre de l'équipe doit donc la communiquer aux autres.

C'est pourquoi ils définissent la notion de *weak achievement goal* :

$$\begin{aligned}(WAG\ x\ y\ p\ q) \quad \triangleq \quad & [\neg(BEL\ x\ p)\ \wedge\ (GOAL\ x \Diamond\ p)]\ \vee \\ & [(BEL\ x\ p)\ \wedge\ (GOAL\ x\ \diamond\ (MB\ x\ y\ p))]\ \vee \\ & [(BEL\ x \Box \neg\ p)\ \wedge\ (GOAL\ x \Diamond (MB\ x\ y \Diamond (MB\ x\ y \Box\neg p)))]\ \vee \\ & [(BEL\ x\ \neg q)\ \wedge\ (GOAL\ x \Diamond (MB\ x\ y\ \neg\ q))]\end{aligned}$$

Un agent a un WAG sous la condition q de réaliser p si : lorsqu'il considère le but comme inachevé, il tente de le réaliser et si lorsqu'il s'aperçoit que le but est réalisé, qu'il est irréalisable ou qu'il est non-pertinent il forme alors le but d'en informer les autres membres de l'équipe.

On va retrouver cette notion sous la forme d'un WMG *weak mutual goal* :

$$(WMG\ x\ y\ p\ q) \quad \triangleq \quad (MB\ x\ y\ (WAG\ x\ y\ p\ q)\ \wedge\ (WAG\ y\ x\ p\ q))$$

Et on va pouvoir définir le but commun persistant (ou *joint persistent goal*) de deux agents x et y sous la condition q d'accomplir p :

$$\begin{aligned}(JPG\ x\ y\ p\ q) \quad \triangleq \quad & (MB\ x\ y\ \neg p)\ \wedge\ (MG\ x\ y\ p)\ \wedge \\ & (UNTIL\ [(MB\ x\ y\ p)\ \vee\ (MB\ x\ y \Box\neg p)\ \vee\ (MB\ x\ y\ \neg q)]\ (WMG\ x\ y\ p\ q))\end{aligned}$$

Ici la notion de WMG introduite précédemment est essentielle, un MG ne suffirait pas car il ne garantirait pas que les agents s'informent mutuellement si ils venaient à penser que la tâche est terminée ou impossible ou non-pertinente.

Maintenant que l'on a défini la notion de but commun sous la forme d'un JPG, il reste à trouver un moyen de l'établir. Pour cela, le théorème suivant va nous aider :

[3] on peut aussi traduire "mutual" par :

réciproque le petit Larousse indique qu'en fait mutuel vient du latin *mutuus* qui signifie réciproque et donne la définition suivante : "qui s'échange entre deux ou plusieurs personnes, qui implique un comportement simultané et réciproque",

commun toujours d'après le petit Larousse, "qui est fait conjointement, à plusieurs" et pour conjointement "ensemble et en même temps qu'une autre chose ou qu'une autre personne"

Theorème 1 *La croyance mutuelle pour chacun des agents dans l'existence d'un PWAG envers les autres agents d'accomplir le but p est suffisante pour établir un engagement commun d'accomplir p sous réserve que (1) il y a une croyance mutuelle que p n'a pas encore été réalisée, et (2) les PWAGs sont interdépendants i.e., l'un des PWAG est relatif à l'autre. Formellement :*

$$\models (MB\ x\ y\ (PWAG\ x\ y\ p\ q)) \wedge (MB\ x\ y\ (PWAG\ y\ x\ p\ r \wedge q)) \wedge (MB\ x\ y\ \neg p) \supset (JPG\ x\ y\ r \wedge q),\ where\ r\ =\ (PWAG\ x\ y\ p\ q)$$

La démonstration peut être trouvée dans [KHC02].

Ainsi l'on définit qu'il est suffisant d'avoir la croyance réciproque dans le PWAG de l'autre pour établir qu'un but commun existe. Pour cela on définit la notion de PWAG :

$$\begin{aligned}(PWAG\ x\ y\ p\ q) \triangleq\ & [\neg(BEL\ x\ p)\ \wedge\ (PGOAL\ x\ p\ q]\ \vee \\ & [(BEL\ x\ p) \wedge (PGOAL\ x\ (MB\ x\ y\ p)\ q)]\ \vee \\ & [(BEL\ x\ \Box\neg p)\ \wedge\ (PGOAL\ x\ (MB\ x\ y\ \Box\neg p)\ q)]\ \vee \\ & [(BEL\ x\ \neg q) \wedge (PGOAL\ x\ (MB\ x\ y\ \neg q))])\end{aligned}$$

Un persistent weak achievement goal ou PWAG représente l'engagement d'un agent envers un autre agent.

1.1.3 Etablissement des croyances et communication

Etablissement d'une croyance mutuelle

Maintenant que l'on sait que la croyance mutuelle dans l'existence d'un PWAG suffit à établir l'existence d'un JPG, la question se pose de savoir comment établir cette croyance mutuelle. Cette phase est communément appelée phase d'établissement des engagements (*establish commitment phase*).

Une des conséquences du théorème 1 est qu'il est dès lors possible d'établir un engagement en commun entre deux agents en utilisant des actions de communication qui vont nous permettre d'établir la croyance commune in each other PWAG.

Les agents sont considérés de bonne volonté c'est à dire que si on leur pose une question, ils sont sensés y répondre.

On définit une notion de sincérité (*sincerity*) pour les agents : un agent x est sincère vis à vis d'un autre agent y concernant une proposition p, si : si x veut que y croit p, il veut en fait que y prenne connaissance de p. Cela implique qu'un agent sincère n'est censé communiquer que des informations qu'il considère comme vraie. De plus, ils considèrent qu'un agent peut croire quelque chose par défaut si il n'a aucune preuve du contraire.

De plus, la communication est considérée comme parfaite, i.e. il n'y a pas de différence entre la réalisation d'un acte de communication et son succès, ce qui revient, si l'on considère a, un acte de communication à :

$$(DONE\ a) \triangleq (MB\ x\ y\ (DONE\ a))$$

Attempt

Un acte de communication peut se définir comme une tentative (an attempt) pour le locuteur de transmettre son état mental.

Définition 1 *Attempt*

$$\begin{aligned}(ATTEMPT\ x\ e\ p\ q\ t) \quad \triangleq\ & t?;\ [(BEL\ x\ \neg p)\ \wedge \\ & (GOAL\ x\ (HAPPENS\ e; \Diamond p?))\ \wedge \\ & (INTEND\ x\ t?; e; q?\ (GOAL\ x\ (HAPPENS\ e; \Diamond p?)))]?; e\end{aligned}$$

L'expression d'une tentative à l'instant t d'accomplir p sous la condition q dans laquelle l'agent x est acteur de l'événement e est la suivante : tout d'abord l'agent x pense que p est actuellement faux, ensuite x a comme but que p devienne vrai après la réalisation de e et finalement il produit l'intention que q puisse devenir vrai après la réalisation de e tant qu'il a le but énoncé précédemment. Cette définition indique que p est un but qui peut ou non être accompli à l'issue de la tentative alors que q représente le moindre effort. Si la tentative n'amène pas la réalisation de p, l'agent peut retenter ou essayer une autre stratégie ou même laisser tomber le but. Cependant si la tentative ne conduit pas à q, l'agent est engagé à reessayer tant que q soit achevé, devenue impossible ou non-pertinente.

Exemple : si je veux vous demander d'ouvrir une porte. Le but de ma requête est que vous ouvriez la porte, l'intention de ma requête est que vous sachiez que je veux que vous ouvriez la porte. Ma requête est réussie si vous comprenez que je veux que vous ouvriez la porte, ma requête est satisfaite si vous ouvrez la porte en réponse à ma requête. Le mieux que je puisse faire est de vous faire connaître mon intention, mais c'est ensuite à vous d'ouvrir ou non la porte. Si j'ai des raisons de penser que vous n'avez pas compris mon intention, je dois alors répéter ma requête, i.e. je dois refaire une tentative de vous faire connaître mon intention. Par conséquent, une tentative se définit par un but et une intention.

Request

Cette définition permet ensuite de définir une requête de la manière suivante :

Définition 2 *Request*

$$(REQUEST\ x\ y\ e\ a\ q\ t) \quad \triangleq \quad (ATTEMPT\ x\ e\ \phi\ \psi\ t)$$

$$\begin{aligned}where\ \phi \quad &= \quad (DONE\ y\ a)\ \wedge\ [PWAG\ y\ x\ (DONE\ y\ a)\ (PWAG\ x\ y\ (DONE\ y\ a)\ q) \wedge\ q] \\ and\ \psi \quad &= \quad (BMB\ y\ x\ (BEFORE\ e\ [GOAL\ x\ (AFTER\ e\ (BEL\ y\ [PWAG\ x\ y\ \phi\ q]))]))\end{aligned}$$

Le but de la requête est que y réalise finalement l'action a et qu'il forme également un PWAG vis-à-vis de x de réaliser a. Le PWAG de x est relatif à un but de plus haut niveau q. Le $PWAG$ de y n'est pas seulement vis à vis du $PWAG$ de x que y réalise a mais également sur ce but q. L'intention de la requête est que y vienne à penser qu'il y a une croyance mutuelle entre lui et x qu'avant de réaliser la requête x avait pour but qu'après avoir effectué sa requête que y aurait la croyance que x a un PWAG vis-à-vis de y d'accomplir le but ϕ de la requête.

JPG establishment

Cohen et Levesque proposent un théorème pour l'établissement d'un JPG :

Theorème 2 *Si une requête de x vers y est suivie par une acceptation de y vers x relativement à a, alors un engagement en commun (JPG) entre x et y que y va réaliser a a été établi (relatif au $PWAG$ original de x).*

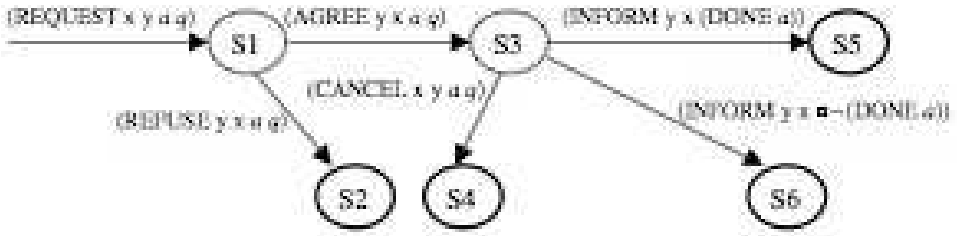


Figure 8.6: A Request Conversation Protocol as a Finite State Machine

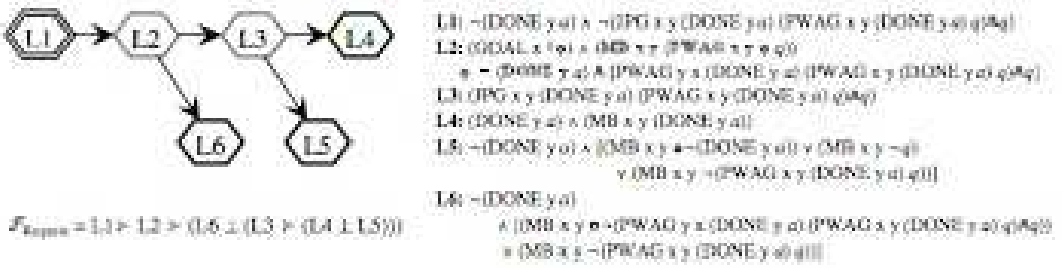


Figure 8.7: Protocol Family For Request Protocol

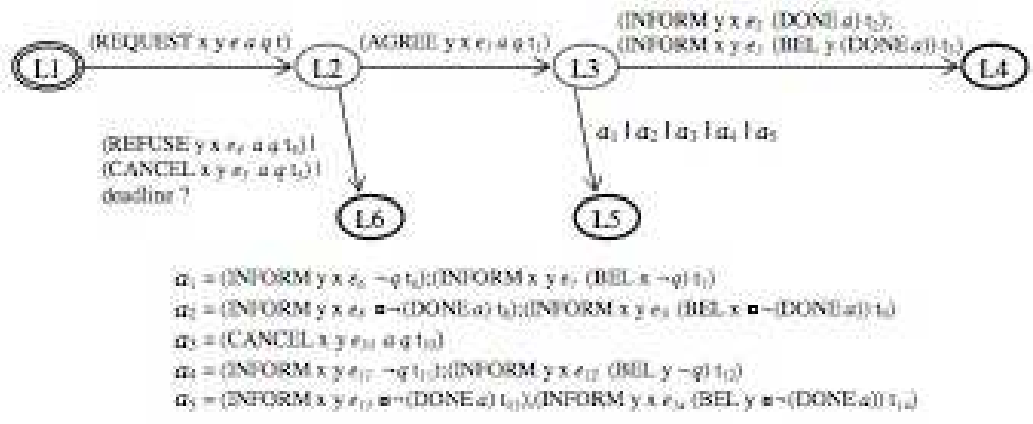


Figure 8.8: Realizing a Request Conversation Protocol From a Request Protocol Family

FIG. 1 – Travaux de Kumar sur les protocoles de communication : Le premier schéma représente un protocole de communication représenté sous la forme d'une machine à état fini, dans ce cas, ce sont les transitions et non les états qui sont importants. Le second schéma représente le même protocole représenté sous la forme d'un ensemble partiellement ordonné de "landmark" qui va représenter une famille de protocole où là ce sont les états et leurs valeurs qui sont importants. Le troisième schéma représente la réalisation du protocole défini précédemment.

Ce que Kumar traduit sous la forme de protocoles de communication tel que montré sur les schémas de la figure 1. Les protocoles de communication sont généralement modélisés sous la forme de machine à état fini tel que le montre le premier schéma de la figure 1, dans ce cas, ce sont les transitions et non les états qui sont importants.

Kumar argumente qu'un protocole de communication peut être exprimé à un niveau abstrait par un ensemble partiellement ordonné de "landmark" ou chaque landmark serait caractérisé par une proposition (qui serait vraie lorsque l'on se trouve dans ce landmark), i.e. ce sont alors les états et leurs valeurs qui deviennent importants tel que modélisé sur le second schéma de la figure 1 . Ces "landmarks" partiellement ordonnés représenteraient une famille de protocoles. Les actes de communications deviennent alors les outils pour concrétiser ces protocoles.

Cette représentation sous forme de landmarks spécifie les points de passages du premier au dernier landmark. Il est possible de spécifier qu'un landmark soit optionnel (qui peut être omis lors de la réalisation du protocole) alors que d'autres doivent être obligatoirement présents. Suivre un chemin signifie alors réaliser les actions associées à un landmark pour permettre d'atteindre le prochain landmark. Rien n'est spécifié sur la complexité des actions.

Ainsi le troisième schéma de la figure 1 représente la réalisation concrète du protocole défini sur le second schéma.

1.1.4 Analyse

La théorie de l'intention jointe a proposé qu'une action jointe était plus qu'une collection d'actions individuelles et qu'il était nécessaire de prendre en compte un "état mental commun" pour former une équipe. Par conséquent, elle définit des buts persistants et la nécessité d'établir des engagements qu'il faut maintenir tant que tous les intervenants de la tâche ne sont pas avertis de sa terminaison ou de son échec.

La théorie de l'intention jointe pose de manière claire les croyances, engagements et obligations nécessaires à la réalisation d'une action jointe. Cependant la définition de croyance mutuelle qui sous-tend ces notions n'est pas simple à modéliser lorsque comme nous, on souhaitera se placer du côté d'un seul agent : i.e. le robot. En effet se pose le problème de la modélisation des croyances de l'homme. De plus les notions de sincérité et de communication parfaite doivent être prises avec précaution pour un robot interagissant avec des hommes.

1.2 Les plans partagés *SharedPlans*

Les travaux de Grosz et de Sidner [GS86], puis de Grosz et Kraus [GK99] qui ont servi à notre étude prennent leur origine dans l'étude du dialogue.

1.2.1 Structure du discours

Les termes d'orateur (*speaker*) et d'auditeur (*hearer*) ne différenciant pas assez à leurs goûts le rôle tenu par les participants d'un discours, on fera la distinction entre l'initiateur de la conversation (*initiating conversational participant* (ICP)) et les autres participants à la conversation (*other conversational participant* (OCP)) pour permettre de faire la différence entre l'instigateur du discours et les autres participants.

Dans leur modélisation, une conversation peut être structurée en trois éléments distincts mais étroitement liés :

- la structure linguistique,
- la structure intentionnelle,

– l'état attentionnel.

La structure linguistique

Tout comme les mots sont les constituants d'une phrase, les éléments du discours (*utterances*) sont naturellement agrégés en segments de discours. Il y a une interaction entre la structure d'une conversation et les éléments la constituant : les expressions utilisées peuvent donner des indications sur la structure du discours et inversement, sa structure contraint l'interprétation des expressions (et a une influence sur quand quelqu'un parle et comment son interlocuteur interprète ce qu'il dit).

La structure intentionnelle

Toute conversation a pour origine une intention, ils distinguent :
– le but du discours (*Discourse Purpose* ou DP) : l'intention qui sous-tend l'engagement dans la conversation, la raison pour laquelle cette conversation a lieu, la raison pour laquelle on va dire çà plutôt qu'autre chose,
– de la contribution d'un segment de discours au discours lui-même (*Discourse Segment Purpose* ou DSP) c'est à dire comment ce segment contribue à l'ensemble.

Ce qui va distinguer un DSP de toute autre intention c'est qu'il est destiné à être rendu public (alors que d'autres intentions sont destinées à rester privées). La réussite d'un discours passe d'ailleurs par la reconnaissance par l'interlocuteur du DP ou du DSP, de manière générale, lorsque l'on dit quelque chose à quelqu'un on s'attend ou du moins l'on souhaite être compris.

Ils identifient deux relations jouant un rôle dans la structuration du discours : la relation de prédominance ou relation de dominance hiérarchique (*dominance hierarchy*) et la relation de pré-satisfaction (*satisfaction precedence*).

Une action qui satisfait une intention DSP1 peut faire partie d'une autre intention DSP2. Dans ce cas, on dit que DSP1 contribue à DSP2 et on dira que DSP2 prédomine DSP1 (DSP2 DOM DSP1). Cette relation est une relation de dominance hiérarchique (*dominance hierarchy*).

En ce qui concerne la relation de pré-satisfaction (*satisfaction precedence*), elle intervient lorsque l'ordre dans lequel les actions sont réalisées est important. On dit alors que DSP1 pré-satisfait DSP2 (ou DSP1 SP DSP2) quand DSP1 doit être réalisé avant DSP2.

Ce qu'il est important de noter c'est que la structure intentionelle n'est ni isomorphe ni identique à un plan. Elle n'est pas identique car un plan regroupe plus qu'une collection d'intentions. Elle n'est pas isomorphe car la structure intentionelle n'a pas la même sous-structure qu'un plan.

Etat attentionel

L'état attentionnel est une abstraction de ce qui focalise l'attention des participants tout au long du discours. Le processus de focalisation (*focusing process*) associe un *focus space* à chaque segment de la conversation. Cet élément contient les éléments qui sont saillants soit parce qu'ils sont mentionnés de manière explicite dans le discours soit parce qu'ils le deviennent au cours du processus de compréhension ou de production du segment de discours.

L'état attentionnel va contenir les informations nécessaires à la compréhension du discours.

La "focusing structure" contient les informations contextuelles nécessaires au traitement des échanges à chaque moment de la conversation. Elle va mettre en exergue les éléments, propriétés, relations qui sont les plus saillantes à un moment donné et elle contient également des liens vers les parties concernées des structures linguistiques et intentionelles.

1.2.2 Définitions

Action

Une action est une entité abstraite et complexe à qui sont associées des propriétés tel qu'un type, un agent, une date et toute autre chose nécessaire à sa réalisation.

A chaque action est associé un ensemble de recette (*recipe*) pour faire l'action : recette(α) indique l'ensemble des recettes pour l'action α. A chaque agent est associé une bibliothèque de recettes qui peut être mise à jour au fur et à mesure de la réussite ou non d'une action.

Une recette est la spécification des actions (de groupe ou non), nécessaires à la réalisation de l'action sous les contraintes définies par la recette. Une recette peut inclure différents niveaux d'abstraction et les paramètres d'une action peuvent ne pas être complètement spécifiés, i.e. une recette peut avoir des variables non-instanciées (même si il peut exister des contraintes sur ces variables), dans ce cas on ne pourra avoir qu'un plan partiel. Une recette, pour une action complexe, peut être représentée par un arbre montrant la décomposition de l'action complexe en sous-actions.

Une distinction est faite entre les actions basiques (*basic level action*) qui sont exécutables à la demande si la situation est appropriée, et les actions complexes (*complex action*) qui nécessite la définition d'une recette.

On distingue également les actions réalisées par un seul agent : actions individuelles (*single-agent action*) et les actions réalisées par plusieurs agents : actions multi-agents (*multi-agent action*).

Dans ce formalisme, toutes les actions basiques sont des actions individuelles.

C_{alpha} représente le contexte dans lequel une action α va se dérouler, on peut distinguer deux parties dans ce contexte :

1. C_α va comprendre les contraintes sur la réalisation de α, e.g. La plan de Kate pour réaliser l'entrée va avoir comme contrainte d'être terminée au début du repas ou de ne pas utiliser telle ou telle casserole.
2. C_α va aussi inclure une représentation du contexte intentionnel dans lequel va se produire l'action, e.g. si α est réalisée dans le cadre d'une action de plus haut-niveau, ce contexte particulier (puisqu'il pourra différer de la même action réalisée "librement") sera représenté ici.

Opérateurs définis

Les opérateurs **Bel** et **MB** sont définis comme précédemment (cf annexe A). On trouve également :

- $Exec(G, \alpha, T_\alpha, \theta)$: représente le fait qu'un agent G est capable de réaliser l'action α au temps T_α sous les contraintes θ. Cet opérateur ne s'applique qu'aux actions basiques.
- $Commit(G, \alpha, T_\alpha, T_i, C_\alpha)$: représente l'engagement de l'agent G au temps T_i pour réaliser l'action α au temps T_α. Cet opérateur ne s'applique qu'aux actions basiques.
- $Do(G, \alpha, T_\alpha, \theta)$: l'agent G réalise l'action α durant T_α sous les contraintes θ. G peut représenter un agent seul ou un groupe d'agents.

Intentions

Voici les différents types d'intentions définies :

- Intention To (Int.To), Potential Intention To (Pot.Int.To) : représentent l'intention (potentielle) d'un agent pour réaliser une action,

– Intention That (Int.That), Potential Intention That (Pot.Int.That) : représentent l'intention (potentielle) d'un agent qu'une proposition soit maintenue.

Les notions d'intentions potentielles représentent l'état mental d'un agent lorsqu'il est en train de considérer la possibilité d'avoir une intention mais n'a pas encore pris de décision notamment sur l'interaction pouvant survenir entre cette intention et les autres. En gros, cela représente les intentions qu'un agent voudrait adopter mais pour lesquelles il ne s'est pas encore engagé.

Un agent peut avoir une IntentionTo seulement pour une action dont il est l'agent et d'ailleurs il ne peut adopter cette intention seulement si il croit pouvoir réaliser l'action. Grosz adopte la position forte qu'un agent ne peut s'engager que sur des actions qu'il croit pouvoir réussir. IntentionThat quand à lui va représenter la responsabilité d'un agent vis à vis des actions des autres agents.

Capacités

Pour représenter les connaissances que les agents ont de leur capacités et des capacités des autres agents, ils définissent : CBA : can bring about et $CBAG$: can bring about group. Plus précisément : CBA représente la capacité pour un agent G de réaliser l'action α en utilisant la recette R_α au temps T_α sous les contraintes θ. L'agent peut soit réaliser lui même les sous-actions de R_α, soit engager quelqu'un pour les réaliser. Il est à noter que dans la définition de CBA, seule la croyance dans l'existence d'une recette est demandée, pas l'identification d'une recette particulière.

Pour représenter le fait qu'un agent peut déléguer une tâche à un ou plusieurs agents, ils définissent : CC : can contract, CCG : group of agent can contract. CC spécifie les conditions selon lesquelles un agent peut sous-traiter une action β à un autre agent.

Le dernier prédicat défini est GTD : get to do, pour les actions α d'un agent individuel, GTD établit le fait qu'en réalisant une action, l'agent peut amener un autre agent à effectuer une autre action.

Planification

Il va être question de deux types de planification, tout d'abord trouver la recette pour réaliser une action (Select_Rec et Select_Rec_GR) et élaborer un plan (Elaborate_Individual, Elaborate_Group) :

– Select_Rec et Select_Rec_GR font références aux actes de planification effectués par les agents de manière individuelle ou collective pour trouver un moyen de réaliser ue action.
– Elaborate_Individual et Elaborate_Group font références aux actes de planification pour étendre un plan partiel en un plan complet.

Pour donner une idée de la différence, vous trouvez la recette de votre plat préféré et ensuite vous établissez un plan pour la réaliser. Dans le cas de décision de groupe, il sera nécessaire de définir des procédures de médiation entre les agents.

Différents types de plans

Différents types de plans sont distingués :

– les plans individuels qui sont constitués par des agents individuels,
– les plans partagés (*SharedPlans*), qui sont construits par des groupes d'agents collaborant.

Quand des agents ont un SharedPlan pour faire une action de groupe, ils ont certaines croyances individuelles et mutuelles à propos de l'action elle-même et des sous-actions qu'il

sera nécessaire de réaliser. Chaque agent va avoir des intentions individuelles et des plans pour réaliser les sous-actions. Les agents peuvent aussi avoir des intentions individuelles concernant l'éxécution de ses actions individuelles et de ses actions de groupe. Ils font une distinction entre les plans complets (plans pour lequel l'agent ou les agents ont determiné complètement la manière de faire l'action) et les plans partiels.

Pour avoir un plan collaboratif pour une action, un groupe d'agents doit avoir :

1. une croyance mutuelle concernant un plan (même partiel) : les agents doivent avoir des procédures pour décider quelle recette doit être utilisée (possiblement en combinant leur savoir sur les recettes),
2. – a) des intentions individuelles que l'action soit réalisée : les agents doivent s'engager non seulement sur leurs propres actions mais également sur les actions qui doivent être réalisées par le groupe
   – b) des intentions individuelles que les collaborateurs réussissent les sous-tâches dans lesquelles ils sont engagés,
3. des plans individuels ou collaboratifs pour les sous-actions : les plans pour les activités de groupe vont être composés de plans pour les agents individuels pour les sous-actions et aussi de plans pour les sous-groupes d'agents devant réaliser des sous-actions. Les groupes doivent avoir les moyens de décider qui de l'agent ou d'un sous-groupe doit réaliser l'action.

Dans tous les cas, il est nécessaire de prendre en compte une donnée importante qui est la nature dynamique des plans. Les plans se construisent, se développent dans le temps. Les agents commencent avec un plan partiel qu'ils étendent jusqu'à ce qu'il devienne complet. Les croyances des agents peuvent être fausses ou l'état du monde peut évoluer pendant la planification ou la réalisation de l'action, tout plan partiel doit pouvoir être révisé, modifié.

1.2.3 Plans individuels

Complet

Un plan individuel complet (*Full Individual Plan*) ou FIP spécifie les conditions dans lesquels un individu G a un plan P, au temps T_p pour faire l'action α suivant la recette R_α dans le contexte C_α, comme indiqué figure 2. Dans cette définition, il est dit qu'un agent peut faire une action, soit en la réalisant lui-même i.e. en y prenant part lui-même, soit en la sous-traitant. Il est à noter que la sous-action β_i sera réalisée dans le contexte défini par l'action α et hérite donc des contraintes C_α.

FIP(P, G, α, T_p, T_α, R_α, C_α)

1. L'agent G connait une recette pour réaliser α ; i.e., il "connaît" les sous-actions à réaliser (β_i) et les contraintes qui y sont associées (ρ_i) :
   $R_\alpha = \{\beta_i, \rho_i\} \wedge Bel(G, R_\alpha \in Recipes(\alpha), T_p)$
   Pour chaque β_i se produira soit (2) soit (3) :
2. **Core case** : Agent G va réaliser la sous-action β_i lui même.
3. **Contracting-out case** : G va demander à un autre agent G_c de réaliser la sous-action β_i.

FIG. 2 – définition d'un FIP ou plan individuel complet

Revenons sur la notion de sous-traitance (*contract-out*), même si on ne développera pas cette notion ici, on notera quand même que cette notion suppose que G dispose d'une action γ

qui lui permettra de faire en sorte que G_c s'engage à faire β_i (et donc G doit s'engager à réaliser γ pour que cela arrive). De plus, G devra s'engager sur les capacités de G_c à réaliser β_i (par l'intermédiaire d'un opérateur CBA) et aussi et surtout ne pas avoir d'intentions qui viendrait en conflit avec la réalisation de β_i par G_c (par l'utilisation d'un Intention That).

Partiel

Un plan peut être partiel pour quatre raisons :

1. l'agent n'a qu'une recette partielle pour réaliser l'action α,
2. l'agent n'a qu'une recette partielle pour réaliser l'une des sous-actions (β_i) de α,
3. il existe une ou des sous-actions pour lesquelles l'agent n'a encore que des intentions potentielles,
4. le plan concernant le moyen de sous-traiter une action peut également être partiel.

A partir de ces données, un plan individuel partiel (*partial individual plan*) ou PIP se définit comme indiqué figure 3.

PIP(P, G, α, T_p, T_α, C_α)

1. l'Agent G pense qu'il est possible de réaliser α ; sa recette pour réaliser α est incomplète, i.e., G a seulement une idée de la manière dont il faut réaliser α.
   Pour chaque sous-action de cette recette partielle, il peut se produire l'un des cas suivants :
2. **Core case** : l'agent G va réaliser la sous-action par lui même, même si il n'a qu'un plan partiel pour le faire.
3. **Contracting-out case** : G va demander à un autre agent G_c de réaliser la sous-action mais ne va pas avoir un plan complet pour l'action qu'il va sous-traiter.
4. **Unreconciled case** : G ne s'est pas encore décidé à réaliser la sous-action, il n'a qu'une intention potentielle de le faire.

FIG. 3 – définition d'un PIP ou plan individuel partiel

Il est clair que dans tout ces cas, l'agent doit se doter des moyens de trouver les informations manquantes (par exemple avoir un moyen de trouver une recette complète lorsqu'elle ne l'est pas) mais l'intéressant c'est qu'il peut répondre à des sollicitations avant même d'avoir toutes les informations (une requête peut par exemple n'être que partiellement instanciée).

De cette analyse et de cette formalisation, il ressort que :

1. il faut déterminer la connaissance minimale requise concernant la manière d'effectuer une tâche (pour exclure les cas où l'agent ne pourra rien faire en raison d'une connaissance insuffisante),
2. il faut spécifier quelles connaissances et quels engagements l'agent peut prendre avant qu'il ait toutes les informations pour réaliser l'action (dans quels conditions est-il possible de définir des Int.To sur des plans partiels),
3. définir les manières de déterminer la recette et le processus de réconciliation d'intentions pour étendre un plan partiel à un plan complet et spécifier dans quels conditions on considère qu'une action est terminée,
4. spécifier ce que les agents doivent établir concernant leur capacité à accomplir une action complexe en ayant des informations incomplètes sur la manière de réaliser l'action

(pour permettre aux agents d'élaborer un plan avant même d'avoir complètement établi les capacités).

Dans ce dernier cadre, ils étendent les définitions concernant "bring about" :

- BCBA : "believe can bring about", littéralement "croyance que l'on peut le faire" représente le niveau requis pour un agent pour avoir un plan complet individuel (FIP) pour pouvoir réaliser une action,
- WBCBA : "weakly believe can bring about", littéralement "croyance faible qu'on peut le faire" représente le niveau de croyance qu'un agent G doit avoir en ses capacités pour selectionner la recette adéquate et exécuter ou sous-traiter chacune des actions constituants la recette.

En effet, lorsqu'un agent a un plan partiel pour α, ses croyances concernant ses capacités sont limitées, parce que sa connaissance de la recette peut être incomplète. Les agents peuvent n'avoir qu'une croyance concernant le moyen de trouver une recette pour réaliser α. Avant de connaître les actions de la recette, l'agent ne peut pas se prononcer sur sa capacité à les réaliser.

La notion de partialité découle en partie de la structure dynamique des plans (*dynamic structure of plans*), en effet les plans se développent au fur et à mesure. Les agents peuvent commencer avec des plans partiels et les étendre jusqu'à avoir des plans complets. Les croyances des agents peuvent être mises en défaut, l'état du monde peut changer pendant la planification ou lors de la réalisation de l'action, les plans doivent être révisés. Si un agent découvre que son plan n'est plus correct il doit le réviser, même si cela implique que le plan passe d'une version complète à une version partielle.

1.2.4 Plans partagés

Les plans partagés (*shared plans*) sont ceux contruits par des groupes d'agents collaborant. A partir des notions définies précédemment on peut lister les propriétés clés du système permettant la définition de ces plans :

- utilisation des intentions individuelles pour établir l'engagement des collaborateurs dans le cadre d'une activité jointe ;
- établissement d'un engagement de l'agent vis à vis des capacités des collaborateurs à faire avancer l'activité jointe,
- définition un comportement d'aide (*helpful behavior*) dans le contexte d'une activité collaborative ;
- définition la notion de sous-traitance et distinction de la sous-traitance et de la collaboration ;
- le besoin de communiquer des agents n'est pas stipulé, il suit l'engagement général du groupe dans l'activité,
- l'imbrication des plans est assuré, cela est aussi dérivé d'autres contraintes plus générales.

L'opérateur intention-that joue un rôle important dans plusieurs de ces propriétés. Tout d'abord il assure que les agents ne vont pas adopter des intentions qui rentreraient en conflit avec le plan du groupe et il est à l'origine du comportement d'aide (*helpful behavior*) i.e. un agent peut par exemple faire une action supplémentaire si cela permet à un autre agent de gagner du temps.

Ainsi un agent ne va avoir des intentions-to que pour les actions dont il est personnellement l'agent alors que intentions-that va représenter sa responsabilité vis à vis des actions des autres agents.

Comme Cohen le définissait déjà, avoir des intentions individuelles et la croyance mutuelle de ces intentions n'est pas suffisant à représenter l'état mental des participants pour des actions

collaboratives. L'introduction dans SharedPlan de Intention-That va permettre de modéliser la nécessaire collaboration pour ce type d'action jointe.

Le modèle SharedPlans ne stipule qu'un minimum de contraintes sur ce que l'agent doit savoir à propos des recettes des actions réalisées par les autres agents. Ainsi, il est possible qu'un agent ne reconnaisse pas un conflit potentiel entre ses intentions-to et ses intentions-that concernant le succès de ses collaborateurs. Un exemple peut être un conflit de ressource, si vous ne savez pas qu'un autre agent a besoin de la ressource que vous prévoyiez d'utiliser, vous ne pouvez éviter le conflit. Les possibilités de détection et les mécanismes permettant de résoudre ce type de conflits ne sont pas étudiés dans cette théorie.

Une des principales différences entre SharedPlans et un plan individuel est que la connaissance concernant la manière d'agir, l'engagement dans l'action sont partagés. Même si le plan est complet, il n'y a pas forcément un individu qui connaît le plan d'action (*recipe tree*) complet ; aucun agent individuel n'a besoin d'être capable de réaliser toutes les actions et les intentions sont répartis entre les membres du groupe. Le groupe à un SharedPlan, mais aucun membre individuel n'a à lui seul un SharedPlan.

Complet

A partir de là un plan partagé complet (*full shared plan*) ou FSP se définit comme indiqué figure 1.2.4.0.

FSP(P, GR, α, T_p, T_α, C_α) :

0 Le groupe GR a la croyance mutuelle que tous les membres du groupe se sont engagés à la réalisation de $\alpha : MB(GR, (\forall\, G_j \in GR) Int.Th(G_j, Do(GR, \alpha, T_\alpha, constr(C_\alpha)), T_p, T_\alpha, C_\alpha), T_p$

1 Le groupe GR a la croyance mutuelle des actions (β_i) qu'il est nécessaire de réaliser et des contraintes (ρ_j) s'y referrant :
$R_{alpha} = \beta_i, \rho_i \wedge MB(GR, R_\alpha \in Recipes(\alpha), T_p)$ Pour chaque β_i peut se produire (2) ou (3) :

2 **Core case** :

- 2a **La sous-action β_i est une action impliquant un seul agent** : Un des membres du groupe va réaliser la sous-action.
- 2b **La sous-action β_i est une action de groupe** : Un sous-groupe va réaliser l'action.

3 **Contracting case** :

- 3a **La sous-action β_i est une action impliquant un seul agent** : Le groupe va déléguer l'action à un autre agent G_c.
- 3b **La sous-action β_i est une action de groupe** : Le groupe va demander à un autre groupe d'agents de réaliser l'action.

Partiel

Un plan partagé partiel (*partial shared plan*) ou PSP se définit comme indiqué figure 1.2.4.0.

PSP(P, GR, α, T_p, T_α, C_α) :

0 Le groupe GR a la croyance mutuelle que tous les membres du groupe se sont engagés à la réalisation de $\alpha : MB(GR, (\forall\, G_j \in GR) Int.Th(G_j, Do(GR, \alpha, T_\alpha, constr(C_\alpha)), T_p, T_\alpha, C_\alpha), T_p$

1 Le groupe GR a la croyance mutuelle dans une recette pour réaliser α, mais cette recette est partielle ; i.e. ils ont une idée de comment faire mais ils n'ont pas identifiées toutes les sous-actions nécessaires à la réalisation.
Pour chaque sous-action β_i de la recette partielle, l'un des cas suivants va se produire :

2 **Core case** :

2a **Single-agent subaction** : Un membrer du group G_k va réaliser l'action, en ayant seulement un plan partiel pour le faire.

2b **Multi-agent subaction** : Un sous-groupe GR_k a un plan partagé (SP) pour réaliser la sous-action, mais ce plan est partiel.

3 **Contracting case** :

3a **Single-agent subaction** : Le groupe a décidé de déléguer la sous-action à un agent G_c extérieur au groupe. Il n'existe qu'un plan partiel (PIP ou PSP) de réalisation de l'action.

3b **Multi-agent subaction** : Le groupe a décidé de déléguer la sous-action à un sous-groupe GR_c, il n'existe qu'un plan partiel (PIPI ou PSP) pour l'action.

4 **Unreconciled case** : Le groupe n'a pas encore décidé qui allait réaliser cette sous-action.

1.2.5 Apport de cette formalisation

Leurs conclusions est que si l'on cherche à développer une formalisation prenant en compte des actions complexes et le développement incrémental de plans partiels dans le cadre d'activités de groupe on est amené à étudier 3 dimensions : l'engagement dans les actions, la connaissance sur comment réaliser les actions et les capacités nécessaires pour réaliser les actions.

Les contraintes que l'on a imposé :

1. La définition d'intention-to requiert qu'un agent ne doit pas avoir des intentions qui sont en conflit, mais ait également une idée de comment réaliser l'action ou tout du moins, une idée de comment trouver comment réaliser l'action. Cette contrainte est essentielle si l'on considère que les agents ne peuvent compter que sur eux-mêmes pour réaliser les actions qui leurs sont assignées,
2. Pour qu'un groupe ait un plan, la formalisation requiert au minimum que les membres du groupe se soient mis d'accord sur les procédures de décision, notamment les procédures concernant la selection des recettes et identifiant quels agents vont réaliser les sous-actions.
3. La formalisation requiert que les groupes aient des procédures pour établir la croyance mutuelle et arriver à des consensus ce qui est un composant essentiel pour l'élaboration d'un plan. Les agents doivent communiquer les informations nécessaires aux autres agents pour qu'ils sachent ce qu'ils sont capables et ce qu'ils ont l'intention de faire.
4. Les définitions des prédicats CBA et CC requiert que les agents calculent le contexte dans lequel les actions sont planifiées.
5. La formalisation spécifie que les agents ne peuvent avoir des intentions-to et des intentions-that qui seraient en conflit.

Ils citent Bratman qui établit trois critères pour qu'une activité multi-agent devienne une "activité coopérative partagée" (*"shared cooperative activity"*) :

- (a) capacité de réaction mutuelle (*mutual responsiveness*) ;
- (b) engagement dans l'activité jointe (*commitment to the joint activity*)

– (c) engagement à un support mutuel (*commitment to mutual support*

Concernant la capacité de réaction mutuelle, Bratman distingue entre la capacité de réaction mutuelle concernant les intentions et celle concernant l'action. SharedPlans traite la capacité de réaction mutuelle concernant les intentions par l'intermédiaire des intentions-that. SharedPlans traite également un aspect de la capacité de réaction mutuelle concernant l'action en réquérant que les plans soient modifiés en cas de problème d'exécution, ce qui apporte une base pour interlacer la planification et l'action. Cependant l'aspect de capacité de réaction mutuelle concernant le monitoring des actions des autres agents n'est pas pris en compte par SharedPlans.

1.2.6 Analyse

SharedPlans est une théorie héritée de l'étude du dialogue. Cette théorie définit des notions très pertinentes, en particulier les notions de sous-traitance et de partialité du plan. En effet, un plan peut se construire au long du déroulement de la tâche, de plus il peut être nécessaire de sous-traiter une partie de la tâche à réaliser. Tout ceci induit des contraintes exprimées notamment sous la forme d'intention-to (les contraintes de l'agent sur ses propres actions) et d'intention-that (les contraintes/engagements de l'agent concernant les actions des autres agents). De plus, est exprimé la notion de représentation hiérarchique de tâches (avec les notions de prédominance et de précédence).

1.3 Joint Action Theory

Nous avons porté notre étude sur le travail de Clark sur le langage et plus particulièrement sur la théorie de l'action jointe ([BC03], [CK04], [CB91], [Cla96]) .

Nous nous sommes intéressés au langage car comme le dit Clark ([Cla96], page 29), deux personnes ou plus ne peuvent réaliser une action jointe sans communiquer et cela nécessite l'utilisation du langage au sens large. D'ailleurs pour lui, la notion d'activité jointe est indissociable de la notion de langage et on ne peut comprendre l'une sans l'autre.

Pour commencer nous allons expliquer les différents types d'actions que définit Clark, puis nous expliquerons la notion de socle commun (*common ground*) et nous étudierons son contenu, sa représentation, sa mise à jour pour enfin terminer par les coordinations nécessaires.

1.3.1 Différents types d'actions

Deux individus, par exemple, Ann et Ben, exécutent une action jointe lorsqu'ils jouent un duo flûte-piano. Les actions individuelles sont accomplies par des individus et les actions jointes par des ensembles d'individus. Il est clair qu'il n'y a pas d'agent nommé : AnnEtBen qui décide "Ah, maintenant je vais jouer ce duo" et qui l'exécute. Les ensembles d'individus n'ont pas d'intentions de réaliser des choses. Seuls les individus en ont. Cependant un ensemble de gens peut jouer un duo ou faire d'autres choses qu'une personne seule ne peut réaliser. Le paradoxe est le suivant : un ensemble de personnes peut réaliser des actions qu'il ne peut avoir l'intention de faire.

En ce qui concerne les actions individuelles, on distingue les actions autonomes des actions participatives. Quand Ann joue seule dans son living-room, elle n'a pas à coordonner ses actions avec quelqu'un d'autre, elle réalise une action autonome. Par contre, lorsqu'elle joue de la flûte pour un duo, elle réalise une action dans le cadre de sa participation au duo avec Ben, c'est une action participative. Les actions participatives sont des actions individuelles réalisées

au sein d'une action jointe et on peut considérer qu'une action jointe est constituée d'actions participatives.

De la même manière qu'une action autonome peut se subdiviser en sous-actions et peut être représentée sous la forme d'une hiérarchie de projets et de sous-projets, une action jointe peut trouver le même découpage. Dans [BC03] est définie une notion de projet joint (*joint project*) qui prend la forme d'une hiérarchie d'actions. De cette hiérarchie, ils font émerger deux types de transitions. Les transitions verticales permettent la navigation entre les différents niveaux de la hiérarchie. Elles permettent aux participants, d'entrer ou de sortir d'une tâche jointe. On peut les comparer au push et pop dans un programme. Les transitions horizontales représentent un changement à l'intérieur d'un niveau de la hiérarchie, au fur et à mesure que la tâche évolue. Ils font cependant remarquer que pour une action jointe, par rapport aux actions autonomes, il y a une différence notable dans le sens où cette action doit être planifiée et menée à bien par ses participants de manière jointe. Ce qui fait qu'une action est une action jointe, c'est la coordination d'actions individuelles par deux personnes ou plus ([Cla96], page 59). Cette coordination se fait sur le contenu, sur ce que les participants veulent faire, sur la manière dont ils veulent le faire, et sur les systèmes qui vont être mis en oeuvre.

1.3.2 Le socle commun et son évolution

Dans le cadre d'une action en commun, les participants vont partager (ou non) un ensemble de données, de présuppositions, de conventions etc.

Cet ensemble va constituer ce que Clark nomme socle commun de connaissance (*common ground*), ce socle commun peut se diviser en trois parties :

- le socle commun initial,
- l'état courant de l'activité jointe,
- les événements rendus publics jusqu'à présent.

Le socle commun initial est constitué des présuppositions des participants à l'action. Les auteurs citent Stalnaker qui dit que dans la cadre d'une conversation : "cela fait partie du concept de présupposition que le locuteur pense que l'audience à laquelle il s'adresse présuppose tout ce qu'il présuppose lui-même". De même lorsqu'ils s'engagent dans un jeu, les participants présupposent que chacun en connaît les règles. Il est à noter que cela peut se révéler faux, l'important étant d'avoir les moyens pour s'en apercevoir pour résoudre d'éventuelles contradictions.

Dans le cadre d'une action jointe, Clark [Cla96] parle d'attente mutuelle (*mutual expectation*). Une attente mutuelle est une croyance mutuelle ou une supposition concernant les participants à une action. Par exemple, si j'ai un rendez-vous avec vous à 8h dans le hall du labo. Ma croyance dans ma participation à cette action jointe, va dépendre de ma croyance en votre venue à ce rendez-vous et vice-versa. Si je n'ai pas la croyance que vous allez venir, je ne viendrais pas non plus. Ce type d'attente mutuelle va permettre la réalisation d'actions jointes.

Une partie non-négligeable de l'état courant de l'activité est supporté par une représentation externe de l'état courant. Ces représentations externes peuvent être caractérisées par un lieu et ce qui l'entoure (ex : un terrain de foot, une salle de classe, une cuisine), des marqueurs qui sont des éléments de l'activité jointe (ex : un ticket de caisse est un marqueur d'une transaction commerciale), ces marqueurs pouvant être interprétés les uns vis à vis des autres (ex : les pièces sur un échiquier),

Le point capital de cette notion de représentation externe est que c'est une information qui est perçue de manière simultanée par les partenaires et c'est une information qui est disponible et va être accessible à tous les participants à un même moment, nous ne parlons là encore que de perception car la compréhension va dépendre des présuppositions existantes sur cette

représentation externe

Bien entendu, cette représentation externe et l'ensemble des présuppositions existantes ne sont pas des données statiques, elles vont évoluer au cours de la réalisation de la tâche, ces éléments sont en perpétuel évolution et doivent donc être perçus et analysés comme tel.

Se pose alors un problème de coordination sur la lecture de ces informations, ainsi les auteurs citent Shelling qui dit qu'il est nécessaire de coordonner les prédictions, de lire le même message dans une même situation, d'identifier un même déroulement des événements basés sur leurs attentes. Ils doivent "mutuellement reconnaître" un même signal pour coordonner leurs attentes.

On parle d'événement commun (*joint event*) ([Cla96], page 41) pour les événements qui vont avoir une influence ou représenter ce que je suis en train de faire, mais qui représentent également une avancée reconnue pour notre activité jointe. Quand un caissier a enregistré les produits, une cloche sonne lorsqu'il obtient le total. Lorsque moi et le caissier entendons cette cloche, nous savons tous les deux que ce total est disponible, donc le caissier peut supposer que je comprends ce qu'il me dit lorsqu'il me lance "12,77". Ainsi, l'état courant d'une transaction est incrémenté non seulement par ce que je fais, mais par tous les événements que nous reconnaissons faire partie de notre activité jointe.

Cela implique également que l'on ne va pas analyser les données d'un point de vue personnel, mais d'un point de vue "joint". Ainsi on ne considère pas une solution qui satisfait un agent mais une solution qui va satisfaire l'ensemble des participants à l'action. Le principe de saillance jointe (*joint salience*) ([Cla96], page 67) indique que la solution idéale pour coordonner deux agents ou plus est la solution la plus saillante, évidente, ou flagrante au vu de leur connaissance commune.

Une manière de régler ce problème de coordination est d'utiliser une convention. La notion de convention citée et modifié par Clark est donné par Lewis pour qui une convention a 5 propriétés :

1. un comportement régulier
2. partiellement arbitraire
3. qui constitue un socle commun au sein d'une communauté
4. est un moyen de coordination
5. pour un certain problème de coordination

Les personnes qui participent à une activité jointe ne sont pas seulement des participants. Ils ont des rôles au sein de cette activité - activity roles. Dans un quartet, l'un des musiciens est premier violon, un autre second violon, un troisième est alto et le dernier est violoncelle. Cette notion de rôle aide à formuler ce qu'ils font et ce qu'on attend qu'ils fassent. Lors d'un concert, les autres personnes constituent le public, un rôle qui définit d'autres activités. Ce ne sont pas seulement des participants, ce sont des participants dans un certain rôle. Ce que les gens vont faire au sein d'une activité est défini par leur rôle. Bien entendu ce rôle, peut changer d'une activé ou sous-activité à une autre, ou même émerger au moment où la nature de l'activité jointe émerge.

1.3.3 Se coordonner

Les actions jointes nécessitent que les participants coordonnent leurs actions individuelles. Dans chaque action jointe, les participants ont des problèmes de coordination : quels actions participatives attendent-ils de l'autre ? Pour résoudre ce problème, ils ont besoin de moyen de coordination - quelque chose qui va dire quelles actions sont attendues. Maintenant selon le principe de salience conjointe, le moyen de coordination idéal pour ces problèmes est la solution qui est la plus saillante, prominente, connaissant leur common ground.

L'utilisation du language nécessite une coordination continue.

Dans tous les cas, il faut noter qu'il n'est pas nécessaire que cette coordination soit équilibrée. La coordination peut être plus ou moins équilibré. Dans certaines actions jointes, les participants vont prendre des actions similaires sans leader spécifique. Se serrer le main, jouer un duo,... peuvent être initié par une personne mais être ensuite équilibré. D'autres actions jointes sont plus déséquilibrées, à certain moments ou même à tous, elles peuvent être conduites ou dirigées par un des participants, les autres suivants.

1.3.4 *Grounding*

Tout d'abord revenons sur la notion de *grounding*. Le meilleur moyen de l'expliquer à mon sens est d'utiliser la définition donnée en conclusion de l'article [CB91] : "Grounding is essential to communication. Once we have formulated a message, we must do more than just send it off. We need to assure ourselves that it has been understood as we intended it to be. Otherwise, we have little assurance that the discourse we are taking part in will proceed in an orderly way. For whatever we say, our goal is to reach the grounding criterion : that we and our addressees mutually believe that they have understood what we meant well enough for current purposes. This is the process we have called grounding."

Pour commencer à étudier le *grounding*, on peut déjà analyser le déroulement d'un échange. et pour cela ils distinguent deux phases (qui peuvent se rapprocher de ce qu'ils appellent "adjacency pair"). On peut dégager deux phases (que l'on peut rapprocher de la notion d'"adjacecy pair") dans une conversation :

- la phase de présentation : "A presents utterance u for B to consider. He does so on the assumption that, if B gives evidence e or stronger, he can believe that she understands what he means by u"
- la phase d'acceptation/ de consentement : "B accepts utterance u by giving evidence e that she believes she understands what A means by u. She does so on the assumption that, once A registers that evidence, he will also believe that she understands."

Ensuite les contributions qui peuvent se faire au niveau de la deuxième phase peuvent être divisée en 4 niveaux, qui vont montrer l'intérêt ou non de l'interlocuteur, ils vont représenter les niveaux de consentement :

- niveau 1 : "to attend to a vocalization" : est ce que la personne m'entend ?
- niveau 2 : "to identify the words, phrases, and sentence" : est ce que la personne m'écoute ?
- niveau 3 : "to understand what the words mean" : est ce que la personne me comprend ?
- niveau 4 : "to consider answering" : est ce que la personne est prête à me répondre ?

Ce que Clark définit c'est qu'il faut obtenir le *grounding* à chacun des niveaux, et surtout que cela ne veut pas dire la même chose selon le niveau obtenu bien entendu, mais cependant que le processus est incrémental, si vous avez obtenu le niveau 3 c'est que le niveau 1 et 2 sont ok par exemple.

Maintenant, comment s'assurer du *grounding*. On pourrait se contenter de la *negative evidence*, c'est à dire que tant que vous n'avez pas de preuve que le grounding n'est pas réalisé, vous supposez que tout va bien i.e. que vous avez été parfaitement compris. Cette forme d'évidence est rarement suffisante et pour cette raison, peu de gens s'en contentent. C'est là qu'intervient la notion de *positive evidence*, quelle preuve je peux avoir que la personne m'a bien suivi ? Trois formes sont communément admises pour cette *positive evidence* :

- *acknowledgement* (accusé de reception) : cà peut être par exemple tout ces petits mots tels que mm, uh uh, c'est sur... qui montre que vous suivez la conversation et que vous avez compris, ce sont des *back-channel responses continuers*,

- *relevant next turn* : dans le cadre d'une conversation, si la personne répond à la question que vous lui avez posé, vous pouvez supposer qu'elle a au moins suivi et peut être compris ce que vous venez de dire. Dans ce cadre, ils définissent la notion d' *adjacency pairs* qui ressemblent aux deux phases énoncé précédemment,
- *continued attention* : tout simplement le fait que vous soyez attentif à une conversation peut montrer que vous la suivez.

Le *grounding* peut se faire par l'intermédiaire de : la voix, du visage, de l'espace de travail, du corps, de l'espace partagé (*shared scene*). Le fait qu'on ne fasse pas tout par l'intermédiaire du dialogue participe de ce qu'ils appellent le *least collaborative effort*, c'est à dire que les gens privilégient le média ou l'association de médias qui vont leur demander le moins d'effort à eux même et à la personne à qui ils s'addressent, i.e. une efficacité maximum pour un minimum d'effort tout en restant précis sur le message que l'on souhaite émettre.

De plus, si le locuteur ne peut pas surveiller son interlocuteur alors il y aura d'avantages d'erreurs commises. Dans [CB91], les auteurs défendaient que le *grounding* était plus facilement atteint lorsque les gens pouvaient surveiller/observer ce que les autres disaient mais aussi leur visage, leurs gestes et leur espace de travail. Cependant dans l'étude de [CK04], ils vont plus loin et indiquent notamment que de pouvoir surveiller l'espace de travail (*workspace*) de l'autre est crucial (la tâche qu'ils étudiaient était la réalisation d'un tangram), plus en tous les cas que de voir le visage des autres participants.

Ensuite le *grounding* va changer : selon le but et/ou l'objet concerné, selon le média utilisé.

Concernant le premier point, ils donnent deux exemples, le premier concerne les possibilités de *grounding* lorsque l'on souhaite se mettre d'accord sur l'objet dont on parle, ainsi on pourra utiliser une description alternative pour montrer qu'on a compris ou montrer l'objet d'un geste... Le second concerne un contenu écrit, par exemple un numéro de téléphone écrit sur une feuille, un locuteur peut le dire tout haut et son interlocuteur va le repéter pour montrer qu'il a compris ; si l'information est dense, le locuteur va donner l'information petit à petit et s'assurer que l'information aura été comprise au fur et à mesure ; il est aussi possible d'épeler l'information... On s'aperçoit qu'en fait ce ne sont que des cas particuliers de ce qu'on a énoncé précédemment.

Concernant le second point : le média utilisé. Il est évident qu'un média offre des possibilités mais impose également des contraintes sur son utilisation, plus précisémment ils listent 8 caractéristiques : copresence, visibilité, auditibilité, cotemporalité, simultanéité, sequentialité, vérifiabilité (*reviewabilité*), revisabilité.

Ainsi une conversation en face à face va imposer la copresence, la visibilité, l'auditibilité, la cotemporalité, la simultanéité, la séquentialité. Alors qu'un échange de lettre va seulement imposer une reviewabilité et une révisabilité.

Lorsqu'il manque une de ces caractéristiques à un média, on doit envisager un changement, seulement ce changement peut avoir un coût supporté par le locuteur et/ou son interlocuteur.

- les coûts supportés par le locuteur : coûts de formulation, de production
- les coûts supportés par l'interlocuteur : coûts de réception, de compréhension
- les coûts supportés par les deux : coûts de démarrage (de mise en route), de délai, d'asynchronie, de changement de locuteur, d'affichage, d'erreurs, de réparation.

Ces coûts ne sont pas indépendants les uns des autres.

Enfin, ils indiquent qu'il y a un lien très fort entre le média qu'on utilise et le propos. Savoir quel média doit être préféré va dépendre de la forme que prend le "grounding" dans ce média et comment le média véhicule le propos. Ainsi on ne donne pas les mêmes informations par mail et de visu.

La question de la mise à jour du *grounding* est abordée plus particulièrement dans [CK04]. En effet, on ne peut considérer qu'une fois obtenu il va se maintenir tout au long de la conversation.

La mise à jour du *grounding* ne peut être considéré comme un processus qui va être réalisé au début ou à la fin d'une conversation. C'est un processus qui doit être considéré comme continu. Ils indiquent que c'est un processus bilatéral et que dans ce cadre on considère que les gens ne sont pas seulement capable de monitorer ce que leurs interlocuteurs font mais on attend d'eux qu'ils le fassent, d'ailleurs que ce soit du côté du locuteur ou de l'interlocuteur. Ensuite que c'est un processus opportuniste, le dialogue est un processus incrémental qui va se nourrir de l'apport de chacun et par exemple des opportunités apportés par son partenaire ou des opportunités qu'on lui donne.

1.3.5 Analyse

Clark ne définit pas clairement une théorie équationnelle, comme on a pu le trouver dans SharedPlans et la théorie de l'intention jointe, mais exprime un ensemble de définitions qui vont servir à qualifier, présenter, représenter, permettre de conduire une activité jointe. La notion de socle commun de connaissance peut être rapproché de la notion croyance mutuelle rencontrée précédemment (cf section 1.1.2.0 page 26). Le point clé de ce travail se situe au niveau de la définition du "grounding" et de sa prise en compte de la notion de compréhension. Ce travail ne s'est pas intéressé seulement au début et à la fin d'un engagement dans une tâche mais considère le "grounding" comme un processus au long cours de la collaboration. Clark s'est également intéressé au maintien et à la mise à jour du socle commun et offre une intéressante étude sur différents médias et les possibilités qu'ils offrent.

1.4 Basic compact

Dans cette dernière section, nous ne présentons pas à proprement parler une théorie mais plutôt un ensemble de réflexions proposées principalement par Klein et Bradshaw.

Klein et Bradshaw [KFBW04] identifient deux critères pour une activité jointe : les parties doivent avoir l'intention de travailler ensemble, et leur travail doit être interdépendant.

Ils proposent qu'une activité jointe requière une convention de base (*basic compact*), un contrat qui constitue le niveau d'engagement que doit avoir chaque participants pour soutenir la coordination. Cette convention de base représente l'accord (généralement tacite) pour participer à l'activité jointe et mettre en oeuvre la coordination, e.g. les coureurs d'un relais entre dans cette convention de base car ils font partie de l'équipe.

1.4.1 The fundamental common ground breakdown

Quoi qu'on fasse, les ruptures de *common ground* sont inévitables : rien ne pourra tout prévenir. Les principales causes de ces ruptures peuvent être :

- les membres de l'équipe peuvent ne pas avoir l'habitude de travailler ensemble,
- ils peuvent avoir des sources de données différentes,
- il peut y avoir un problème de compréhension de ce qu'a demandé le leader,
- il peut y avoir des problèmes de pertes de communications sans qu'ils sachent comment "réparer" cette situation,
- ils peuvent échouer à recevoir un message de confirmation et peuvent s'embrouiller à propos de qui sait quoi.

Ils nomment cette rupture *fundamental common ground breakdown*, dont ils donnent plusieurs formulations.

1er cas

L'un des participants fait une hypothèse fausse concernant les croyances de l'autre. Exemple : une personne A suppose que B a l'information X, alors que la personne B n'en sait rien. Un nouvel événement arrive, A interprète les croyances et les actions de B d'une manière divergente à la logique des actions et des croyances actuelles de B. La divergence augmente de plus en plus jusqu'à ce qu'il advienne une "surprise de coordination".

2eme cas

Parfois les partenaires vont s'avertir qu'ils abandonnent l'activité jointe. Dans la plupart des cas, l'accord est obtenu de manière tacite (les partenaires utilisant des moyens subtils pour signaler qu'ils ne participent plus à l'activité jointe). Cependant, si ce signal échoue, nous avons encore une forme de rupture du *common ground*.

1.4.2 Conditions pour une coordination efficace dans le cadre d'une activité jointe

Ils listent ce qu'ils nomment les conditions pour une coordination efficace dans le cadre d'une activité jointe :

Interprédictabilité (*interpredictability*) La coordination va dépendre de notre capacité à prédire les actions des autres de manière précise. Chacun des participants a aussi alors la responsabilité de rendre ses actions lisibles pour permettre la collaboration.

Common Ground Ils reprennent ici la notion développée par Clark (cf section 1.3.2 page 41) qui fait référence à la connaissance mutuelle pertinente et aux croyances et suppositions réciproques qui vont soutenir les actions jointes et leur interdépendance au sein des activités jointes. Le *Common Ground* peut être divisé en trois catégories :

Initial common ground inclut toute la connaissance a priori des participants concernant l'activité jointe. Il comprend non seulement les connaissances qu'ils partagent sur le monde mais également les conventions qui sont associées à la réalisation de leur tâche jointe.

Evénements rendus publics jusqu'à présent inclut la connaissance des événements survenus jusqu'à présent.

Etat courant de l'activité regroupe les signaux permettant de prédire les actions ultérieures et ainsi de prévoir la forme correcte de coordination.

Dirigeabilité qui refère aux capacités de délibération pour modifier les actions de ses partenaires en fonction des changements de contexte, de priorités, ...

Ensuite, en analysant plusieurs types de coordination d'équipe, ils ont listé les types les plus importants de connaissances, croyances, hypothèses :

- les rôles et fonctions de chacun des participants ;
- les routines que chaque équipe est capable d'exécuter ;
- les savoir-faire et compétences de chacun des participants ;
- les buts de chacun des participants, et leurs engagaments ;
- l'état de chacun des participants (position, posture,...)

Ils dégagent alors plusieurs activités que les équipes vont réaliser pour maintenir le *common ground* :

1. **structurer** leur préparation pour établir les paramètres initiaux et pour établir les routines qui seront utilisées à l'exécution.
2. **soutenir** le *common ground* en insérant si besoin des clarifications et des rappels, soit pour être sûr de quelque chose, soit pour valider les hypothèses des autres participants,
3. **informer** les autres, mettre à jour leurs connaissances à propos des éléments se produisant hors de leur point de vue,
4. **monitorer** les autres membres de l'équipe pour juger si le *common ground* peut être compromis ou si il est rompu,
5. **détecter les anomalies** signalant une potentielle perte de *common ground*,
6. **réparer** les pertes de *common ground*.

1.4.3 10 Challenges

A la suite de leur travaux, les auteurs ont publié ce qu'ils ont nommé les 10 défis pour automatiser un équipier dans le cadre d'une activité jointe homme-robot (*Ten challenges for Making Automation a "Team Player" in Joint Human-Agent Activity*) :

1. pour être membre d'une équipe, un agent doit satisfaire la convention basique de s'engager dans une activité à faire en commun,
2. les agents doivent être capables de modéliser les autres participants de manière adéquate,
3. les agents doivent être mutuellement prévisibles,
4. les agents doivent pouvoir être dirigés,
5. les agents doivent être capables d'exhiber, rendre visibles leurs intentions à leurs partenaires,
6. les agents doivent être capable d'interpréter les signaux envoyés par leurs partenaires,
7. les agents doivent avoir les moyens de négocier sur leur but,
8. lest algorithmes de planification et de gestion de l'autonomie doivent supporter la collaboration,
9. les agents participent à la gestion de l'attention des autres agents (possiblement en les prévenant quand un événement important se produit),
10. les agents doivent pouvoir contrôler les coûts de leurs activités coordonnées,

1.4.4 Analyse

Nous avons présenté ce travail car ces 10 challenges nous semblent un résumé intéressant des qualités que doit avoir un agent collaboratif. Cependant, si ces challenges peuvent aider à la définition d'agents logiciels et robotiques, il n'est pas évident de placer l'agent "humain" ou plutôt l'homme tout court au milieu de l'équipe.

Chapitre 2

Les systèmes

Dans ce chapitre, nous allons présenter différents systèmes élaborés sur la base de théorie précédente. Il s'agit de *Teamwork* de Tambe, Collagen de Rich et Sidner, HRI/OS de Fong et nous finirons sur les travaux de Breazeal.

2.1 Teamwork

2.1.1 Définition

teamwork "cooperative effort by the members of a team to achieve a common goal." - American Heritage Dictionary

Le teamwork que l'on pourrait traduire par travail d'équipe a découlé de la théorie de l'intention jointe présenté précédemment, mais pas seulement.

Ainsi dans le premier système implémenté appelé STEAM [Tam97], les auteurs indiquaient que 4 conditions devraient être ajouté aux principes généraux de la théorie de l'intention jointe :

- La première condition concerne la cohérence dans le travail d'équipe. Les membres de l'équipe doivent avoir une solution commune pour accomplir le but. Sans cette contrainte, exprimée par Jennings (1995), les membres de l'équipe peuvent poursuivre des solutions qui s'annulent les unes les autres n'amenant aucun progrès dans l'accomplissement du but commun.
- Ensuite, ils relèvent aussi que Grosz et Kraus [GK96] posent la question du compromis concernant la quantité d'information qui doit être maintenue concernant l'état de l'équipe et de ses équipiers. Dans SharedPlans, ils considèrent juste qu'il est nécessaire que les membres de l'équipe connaissent le moyen pour permettre aux membres de l'équipe d'exécuter les actions. Dans STEAM, les acteurs d'une tâche doivent s'assurer que les acteurs des éventuels sous-tâches exécutent ou non les actions qui leurs sont dévolues, cela pour permettre de garder une information sur l'activité actuelle de l'équipe. Ce tracking ne nécessite pas pour l'agent d'avoir une reconnaissance du plan qui est poursuivi, seulement il doit savoir si l'agent chargé de réaliser le sous-but a toujours l'intention de le réaliser.
- Troisièmement, encore en analogie avec les SharedPlans et leur notion de *"unreconciled case"*. A chaque fois que l'intention jointe d'une équipe est déclarée *"unachievable"*, STEAM va former une autre intention jointe consistant à replanifier. Cette replanification va en plus avoir l'avantage de permettre l'analyse des causes du précédent échec.
- La dernière remarque concerne la communication. Là où Joint Intention se contente d'un mutual belief sur la terminaison d'une tâche, STEAM considère que cette information bien

que nécessaire n'est pas suffisante. Il définit des informations dépendantes des actions qu'il est aussi nécessaire de communiquer, par exemple à la fin d'un mouvement le robot ne communiquera pas seulement sur l'accomplissement de sa tâche mais aussi sur sa position actuelle par exemple.

Nous n'expliquerons pas ici en détail le fonctionnement de STEAM, mais plutôt celui de Machinetta qui est son successeur, il est à noter que dans [SPS+04] les auteurs indiquent qu'ils trouvaient 3 limitations à ce premier système. Ils considéraient que si dans le cas d'équipe de petite taille il pouvait être raisonnable d'allouer les rôles un à un avant le début de la tâche, cela ne l'était pas pour des équipes de grandes tailles. De plus, dans un environnement très dynamique, allocation et réallocation de rôle devaient pouvoir avoir lieu de manière transparente, i.e. pas un algorithme spécifique pour l'allocation et un autre pour la réallocation mais un et un seul qui gérerait l'ensemble, avec également le but que cette réallocation ne survienne pas qu'en cas d'échec mais sur l'arrivée de nouvelles opportunités. Enfin, avec la création d'équipes de plus en plus hétérogènes incluant notamment des hommes, le système doit permettre la prise en compte de l'expertise de l'homme non seulement durant la tâche mais également dans le cadre de la coopération entre les agents. Ainsi, ils ont développé des algorithmes pour prendre en compte l'homme et ses capacités non seulement pour les tâches du domaine (*in domain-level task*) mais également au niveau de la coordination et cela même si la prise en compte de la contribution de l'homme est en désaccord avec l'algorithme courament utilisé par le proxy (exemple : prise de décision arbitraire de l'homme pour arrêter une tâche).

Toutes ces remarques ont conduit à la définition d'un système multi-agent basé sur une notion de proxy[4] appelé Machinetta.

Chaque agent possède un proxy. Un proxy est un composant logiciel qui facilite les actions et les communications nécessaires entre les robots, les agents et les personnes (RAPs[5]) pour réaliser conjointement un plan d'équipe. Pour cela, il va contenir la connaissance concernant le travail d'équipe et va gérer la coordination entre les agents.

Les proxies vont exécuter des TOP *Team Oriented Programs*. En écrivant ces TOP , les programmeurs n'ont pas besoin de connaître les détails de bas niveau . A la place, ils décrivent les activités de l'équipe sous forme de primitives de haut-niveau tel que des rôles (*roles*) et des *team plan operators*. Ce niveau d'abstraction permet au programmeur de spécifier rapidement les activités d'une équipe.

Un proxy est constitué de 5 composants dont les liens sont représentés figure 4 :

Communication gestion de la communication avec les autres proxies, il prend en compte les différents moyens de communication (forcément spécifique de chaque RAP)

Coordination gestion du raisonnement concernant les plans d'équipe et la communication

State mémoire de travail du proxy

Adjustable Autonomy gestion du raisonnement pour savoir si l'agent doit agir de manière autonome ou passer le contrôle au RAP

RAP Interface gestion de la communication avec le membre de l'équipe auquel est associé le proxy. L'interface RAP connaît le type de RAP auquel il est connecté et les moyens de communiquer avec lui. Par exemple, l'interface pour une personne sera une interface graphique alors que pour des agents purement logiciels de simple sockets de communication peuvent suffire.

[4]Nous ne traduirons pas le terme *proxy*, pour information proxy se traduit en français par mandataire, fondé de pouvoir (ainsi *to vote by proxy* signifie voter par procuration)

[5]RAP : Robot Agent Personne

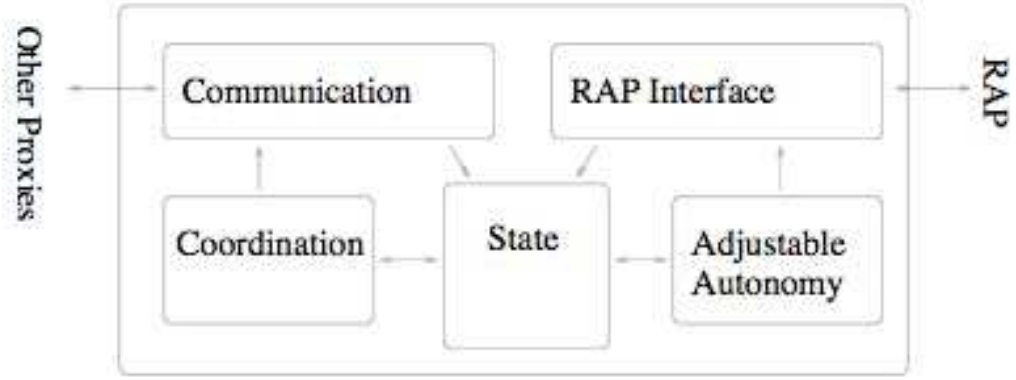


FIG. 4 – ...

Dans un premier temps, nous allons expliquer plus en détail le rôle de chacun de ses composants. Ensuite, nous examinerons le déroulement du processus.

2.1.2 Les composants du proxy

Adjustable Autonomy

Ce composant examine les circonstances pour lesquelles le proxy doit agir de manière autonome ou à l'opposé doit attendre une entrée de son RAP. Les autres composants et les autres proxies ne vont avoir connaissance que de la décision prise (ils ne sauront même pas si cette décision aura été prise de manière autonome ou suite à une demande faite à l'équipier concerné). Le composant adjustable autonomy gère le RAP comme une entité abstraite qui a des capacités particulières. Ce composant va décider si une interaction doit avoir lieu avec le RAP ou pas (mais c'est l'interface du RAP qui va gérer cette interaction si elle doit avoir lieu).

RAP interface

L'interface du RAP est la seule partie du proxy qui doit être conçue de manière spécifique en fonction du membre de l'équipe qu'il représente. Par exemple, l'interface pour une personne sera une interface graphique alors que pour des agents purement logiciels de simple sockets de communication peuvent suffire.

L'interface RAP connaît le type de RAP auquel il est connecté et les moyens de communiquer avec lui.

Communication

Le composant de communication doit prendre en compte les moyens de communications (forcément spécifique de chaque RAP)

2.1.3 Déroulement

Notion de Team Oriented Plans

Une équipe de proxies met en oeuvre des TOPs (pour *Team Oriented Plans* ou plan d'équipe). Un TOP est la description des activités nécessaires à l'accomplissement des buts de l'équipe. Un

TOP contient trois parties : les préconditions, le déroulement et les postconditions du plan.

Le processus d'exécution du proxy est conduit par les messages (*message driven*). Lorsqu'un message arrive du composant RAP ou d'un autre proxy, une nouvelle croyance (*belief*) est ajoutée à l'état du proxy. Les croyances de l'état constituent la connaissance du proxy concernant l'état de l'équipe et de son environnement. **State**, l'**État** constitue un tableau noir sur lequel les différents composants vont écrire des informations tandis que d'autres vont réagir aux informations écrites. Chaque changement d'état va déclencher deux algorithmes : *Coordination* et *Adjustable Autonomy*. Chacun de ces algorithmes pouvant déclencher un changement d'état qui réenclenchera lui aussi ces mêmes algorithmes.

L'algorithme de coordination

L'algorihtme de coordination implémente la théorie de l'intention jointe.

Quand les croyances du proxy correspondent aux préconditions de démarrage d'un plan (i.e. lorsque *startTeamPlan* est vrai), un nouveau plan est instancié. Les préconditions peuvent coder des contraintes temporelles (tel plan doit avoir été achevé avant de démarrer tel autre) ou de hiérarchie, tel but est sous but d'un autre ainsi il faut que le but soit actif pour que le sous-but le soit aussi entre autres contraintes.

Les fonctions *establishJointCommitment* et *endJointCommitment* établissent et terminent les engagements en communicant avec les autres proxies quand des nouvelles croyances déclenchent le démarrage ou la fin d'un plan d'équipe. Dans cet algorithme, la fonction *communicate ?* retourne vrai si des informations sur les capacités ou le progrès dans le plan doivent être communiquées aux autres.

Lorsqu'un plan est déclenché, les proxies vont exécuter la procédure d'*establishJointCommitment* spécifié par leur procédure de coordination. Il est possible d'implémenter des politiques plus ou moins fines prenant en compte par exemple une communication parfaite ou alors calculant les bénéfices d'une communication sur l'établissement de la tâche ou de son absence.

Le TOP inclut les conditions de terminaisons du plan : achieved, irrelevant, unachievable. Ces spécifications requierent une connaissance partagée de ces données, Machinetta se base sur ces conditions de terminaison pour générer automatiquement les communications nécessaires à la terminaison du plan en commun.

Algorithm 1 : Coordination
COORDINATION(B_{in})
(1) **foreach** $b \in B_{in}$
(2) **if** b is *CapabilityInformation* or *Role Progress*
(3) **if** *communicate ?(b)*
(4) *sendToOthers(b)*
(5) **else if** *startTeamPlan* α *?(b)*
(6) *establishJointCommitment*(α)
(7) ALLOCATERole*role*(α)
(8) **else if** *endTeamPlan* α *?(b)*
(9) *endJointCommitment*(α)
(10) **return** B_{out}

L'algorithme d'Autonomie Ajustable

Ensuite, l'algorithme d'autonomie ajustable va gérer les interactions entre le proxy et le RAP. La fonction *shouldRAPbeAsked ?* est au coeur de l'algorithme de raisonnement sur l'autonomie et décide si une décision peut être prise par le proxy ou si le RAP doit être consulté. Ainsi lorsqu'un nouveau rôle est proposé au RAP, le proxy décide en premier lieu d'agir de manière autonome ou non.

Algorithm 2 : Adjustable Autonomy
ADJUSTABLEAUTONOMY(B_{in})
(1) **foreach** $b \in B_{in}$
(2) **if** b is *role offer*
(3) **if** RAP is capable of role
(4) **if** *shouldRAPbeAsked ?*
(5) Ask RAP
(6) **else if** accept autonomously ?
(7) $B_{out} \leftarrow$ *role accepted*
(8) **else**
(9) $B_{out} \leftarrow$ *role rejected*
(10) **else if** b is a new role
(11) send role to RAP
(12) **return** B_{out}

2.1.4 Analyse

Ce système a été déployé et a fonctionné dans plusieurs environnements et notamment au sein d'un système nommé Electric-Elves dont ils font une auto-critique intéressante dans [TBP+06].

L'algorithme d'autonomie ajustable choisit si il doit y avoir un transfert ou non de contrôle vers le RAP, seulement si le RAP ne prend pas le contrôle qu'on lui donne alors il y a un blocage du système. Cela est dû au fait que leur système est construit selon un principe considérant la communication comme parfaite (i.e. si une information est envoyée elle est forcément reçue et comprise par son destinataire), ce qui peut être vrai (et encore) pour des agents logiciels mais qui demande des ajustements avec des agents humains.

L'insertion de l'incertitude dans leurs algorithmes leur a aussi posé des problèmes, en effet, cela peut conduire à des situations où le proxy ne réagit pas toujours de la même manière ce qui est très déconcertant pour les utilisateurs.

Cependant ce travail a montré une utilisation intéressante de la théorie de l'intention jointe.

2.2 Collagen

L'un des systèmes développés sur la base de SharedPlans est Collagen. Collagen a principalement été développé et utilisé dans le cadre IHM [RS97, RS98, RSL01], même si tout récemment il y a eu des développements en robotique [SLK+05]. La question de départ était de construire un agent collaboratif ouvrant des perspectives différentes d'une simple interface interactive.

2.2.1 Collaboration

Ils définissent la collaboration comme un processus dans lequel deux participants ou plus coordonnent leurs actions pour accomplir des buts qu'ils ont en commun. Pour eux, la collaboration

entre deux personnes qui sont co-présentes implique :

- qu'ils communiquent,
- une interaction avec l'objet sur lequel porte l'interaction (i.e. l'interface),
- et l'observation de ce que l'autre exécute avec l'objet de l'interaction. .

Tout au long de la collaboration, les participants vont être amenés à partager certaines croyances notamment concernant leurs intentions d'exécuter des actions, l'avancement de leur tâches en commun, à propos de l'état de l'interface, etc.

Ceci est à replacer dans leur contexte d'interaction homme-machine où en fait leur objectif est de placer un agent entre l'homme et l'interface (l'homme et l'agent pouvant indifféremment contrôler l'interface). L'agent doit être capâble de :

- communiquer avec un homme,
- manipuler l'interface,
- observer les manipulations de l'homme sur l'interface.

Ce qu'amène la collaboration d'après eux c'est la possibilité donnée à l'utilisateur de ne plus s'occuper des détails. Il donne à l'agent une tâche ou exprime ce qu'il veut voir réaliser et l'agent s'occupe de faire en sorte que cela se réalise. L'utilisateur n'a plus à se soucier des détails.

De ce point de vue, une interface interactive dispose de trop peu d'informations, alors que l'agent collaboratif dispose d'informations concernant ce que fait l'utilisateur, il peut en déduire ce que l'utilisateur souhaite faire et donc être pro-actif en proposant les solutions adéquates. Notamment, ils ont développé un processus appelé : "intention plan recognition" qui permet à l'agent d'interpréter les faits et gestes de l'homme pour reconnaître ce qu'il souhaite faire.

2.2.2 Etat du discours

Collagen reprend le formalisme élaboré par Grosz et Sidner [GS86] (cf section 1.2.1 page 31) dans lequel le discours collaboratif peut être structuré en 3 parties :

- la structure intentionnelle qui prend la forme d'un plan partagé partiel (*partial Shared-Plans*),
- la structure linguistique qui groupe de manière hiérarchique les actions en segments,
- la structure attentionnelle, qui se présente sous la forme d'une *focus stack of segments*.

Ces trois parties vont être représentées dans Collagen sous la forme d'un Etat du Discours (*Discourse State*) tel que représenté figure 5.

La structure intentionnelle

La représentation arborescente du plan représente la plan partagé partiel (*partial SharedPlan*) existant entre l'utilisateur et l'agent. Pour réaliser leur collaboration, l'utilisateur et l'agent doivent mutuellement croire que :

- qu'ils ont un but en commun,
- qu'ils sont d'accord sur une séquence d'actions (une recette (*recipe*)) pour accomplir leur but commun,
- qu'ils sont capables de réaliser les actions qu'on leur a assigné,
- qu'ils vont réaliser ces actions,
- et enfin qu'ils sont engagés sur le succès global de la collaboration (et non seulement sur la réalisation de leur propre partie).

Cela définit le plan partagé entre l'utilisateur et l'agent.

Ce qu'il faut remarquer c'est que :

– en raison de la connaissance partielle des agents entre eux et de l'environnement qu'ils partagent, la collaboration ne débute généralement pas avec toutes ces conditions définies, c'est pourquoi l'on définit un plan partagé partiel. La communication peut alors servir à décider quelle recette utilisée, qui doit faire quoi, etc.
– les plans partagés sont récursifs, ainsi, la première étape d'une recette peut être elle aussi un but pour lequel les agents devront collaborer,
– la planification (se mettre d'accord sur les croyances et les intentions nécessaires à la collaboration) et l'exécution sont généralement intercalées pour chaque participant et entre les participants.

La structure linguistique

Le concept de segments du discours (*discourse segments*) provient de la théorie du discours. L'analyse d'un grand nombre de discours d'interactions a conduit à un accord sur le fait que le discours à une structure hiérarchique. Les éléments de la hiérarchie sont appelés segments. Un segment est une séquence d'actes de communication contigus qui portent sur le même propos. Par exemple, une question et sa réponse constituent un segment du discours dont le but est d'obtenir une connaissance partagée sur un fait.

Dans Collagen, la structure linguistique est représentée sous la forme d'un historique d'interactions segmenté (*segmented interaction history*) comme sur la partie droite de la figure 5. C'est une représentation compacte, sous forme de texte, du passé, du présent et des futurs états du discours. Les segments inachevés qui sont actuellement dans la pile de focalisation sont annotés au présent, alors que ceux qui sont terminés sont au passé. Les lignes en italiques à la fin de chaque segment, incluant le mot clé *expecting*, indique les étapes du plan qui n'ont pas encore été exécutées.

La structure attentionnelle

La structure attentionnelle est représentée par une pile de focalisation (*focus stack*). Le but se trouvant sur le haut de la pile étant le propos actuel du discours.

Le changement de direction de l'attention (*the shifting focus of attention*) dans un discours est représenté au sein de la pile (*focus stack*) de segments du discours. Au cours d'un discours collaboratif, de nouveaux segments et sous segments sont créés, mis sur la pile de focus, complété et ensuite dépilé au fur et à mesure que le SharedPlans se déroule au cours de la conversation. Parfois les participants peuvent s'interrompre, abandonner leur plan courant même si il n'est pas terminé ou retourner à de précédents segments.

2.2.3 Interprétation du discours

Collagen met à jour l'état du discours à l'arrivée de chaque nouvel acte de communication ou de la détection d'une nouvelle action en utilisant l'algorithme d'interprétation du discours de Lochbaum [Loc98] (avec des extensions pour prendre en compte la reconnaissance de plan et les déplacements d'attention non prévus (Lesh et al. 2001)).

L'interprétation du discours va regarder comment la communication reçue ou l'action observée peut prendre part à l'actuel propos du discours (i.e. celui qui se trouve en haut de la pile de focalisation). 5 cas sont pris en considération, ainsi l'acte courant peut :
– réaliser, accomplir le propos,
– être une des étapes de la recette pour accomplir le propos courant (ce qui peut induire une recherche dans une bibliothèque de recette),

– permettre d'identifier la recette qui doit être utilisée pour accomplir le but,
– permettre d'identifier qui doit accomplir le but ou l'étape de la recette choisie ;
– permettre d'identifier un paramètre non-spécifié du propos courant ou une étape dans la recette courante.

Si l'un de ces cas est identifié, l'acte courant est ajouté au segment courant (et le plan partagé partiel est mis à jour). De plus, si l'acte a permis de terminer le segment actuel du discours, le segment est dépilé.

Si aucun de ces cas n'est identifié, l'interprétation du discours conclut que l'action est le début d'une interruption, i.e., c'est un segment qui ne contribue pas au segment parent. Un nouveau segment est alors empilé sur la pile de focalisation (*on the focus stack*) avec comme premier élément l'acte courant. Le but de ce segment peut ou non être connu, cela dépend du contenu de l'acte qui en est à l'initiative.

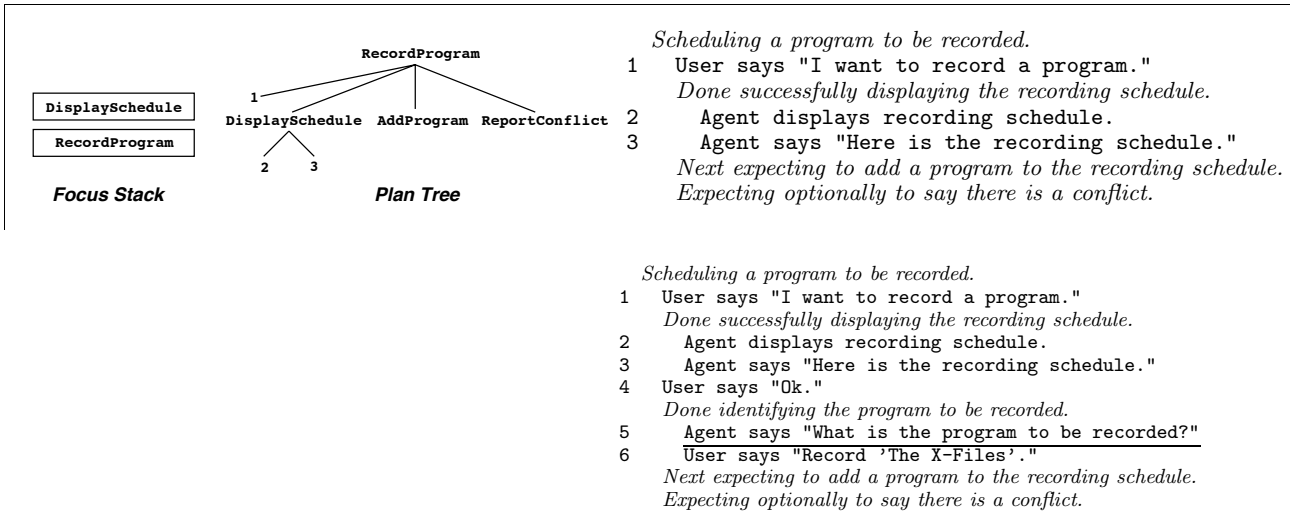


FIG. 5 – Etat du discours dans Collagen

Une recette (*recipe*) est une méthode de décomposition de buts (qui fait partie du modèle de la tâche). Le language de définition de recette de Collagen dont on peut voir un exemple sur

```
public recipe RecordRecipe achieves RecordProgram {
   step DisplaySchedule display;
   step AddProgram add;
   optional step ReportConflict report;
   constraints {
      display precedes add;
      add precedes report;
      add.program == achieves.program;
      report.program == achieves.program;
      report.conflict == add.conflict;
   }
}
```

FIG. 6 – Exemple de définition d'une recette dans Collagen

la figure 6, permet de prendre en compte des plans partiellement ordonnés, les paramètres, les contraintes, les pré- et post- conditions et des décompositions alternatives. Une recette est représentée concrètement comme une séquence d'actions partiellement ordonnée avec des contraintes entre les actions. La librairie de recettes contient les recettes indexées par leurs objectifs. Il peut y avoir plus d'une recette pour un objectif.

Leur implémentation de l'algorithme d'interprétation du discours nécessite que les déclarations soient représentées sous la forme du language artificiel du discours de Sidner [Sid94] puis l'utilisation des techniques standard de du language naturel pour calculer cette représentation à partir des entrées (*spoken input*) de l'utilisateur. Sidner définit un ensemble de constructeurs pour les actes basiques de communications comme : proposer, se retirer, accepter/refuser une proposition. L'implémentation courante comprend deux de ces actes : `PFA` (propose for accept) et `AP` (accept proposal).

$$\texttt{PFA}(t, participant_1, belief, participant_2)$$

`PFA` : à l'instant t, $participant_1$ croit $belief$, il communique sa croyance au $participant_2$, et a pour objectif (*intends*) que le $participant_2$ le croit également. Si $participant_2$ répond par `AP`, e.g., "Ok", alors la croyance $belief$ est partagée (*mutually believed*).

Le language définit par Sidner à ce niveau est très général, $belief$ pouvant être n'importe quoi. Pour communiquer à propos d'activité collaborative, ils introduisent deux opérateurs *application independent* pour former des croyances à propos des actions : `SHOULD`(act) et `RECIPE`$(act, recipe)$.

Le reste du language est spécifique de l'application.

2.2.4 Génération du discours

L'algorithme de génération du discours est pour une grande partie l'inverse de l'interprétation. En fonction du focus stack et du SharedPlan associé, il produit un agenda des actions (possiblement partiellement spécifié) qui pourront contribuer (selon les 5 cas définis précédemment) au segment courant du discours. Par exemple, si le but du segment courant est de programmer conjointement un voyage, l'agenda va inclure une action pour que l'agent demande à l'utilisateur de choisir une route. Dans Collagen, l'agenda contient des actions de communication et de manipulation réalisables par l'agent ou l'utilisateur permettant de faire progresser le processus de résolution du problème.

La figure 5 illustre comment l'état du discours de Collagen est utilisé pour générer et interpréter le discours.

2.2.5 Architecture du système

En entrée, Collagen dispose de toutes les phrases dites par l'homme et l'agent et de toutes les actions effectuées sur l'interface.

La meilleure façon de comprendre un cycle d'exécution de l'architecture est d'analyser la figure 7. Les communications ou les observations (provenant de l'agent ou de l'utilisateur) arrivent au module d'interprétation au milieu du diagramme. Le module d'interprétation met à jour l'état du discours tel que décrit section 2.2.3. Ensuite un nouvel agenda des communications et des actions attendues est généré par le module de génération du discours. L'agent peut alors choisir une entrée de cet agenda pour l'exécuter. Remarque, si il arrive qu'il manque une information à l'agent, Collagen génère une requête à l'utilisateur concernant l'information manquante.

Un sous-ensemble de l'agenda est présenté à l'utilisateur en ouvrant une fenêtre de communication, ce sous-ensemble présentera les actions de l'agenda que l'utilisateur peut effectuer.

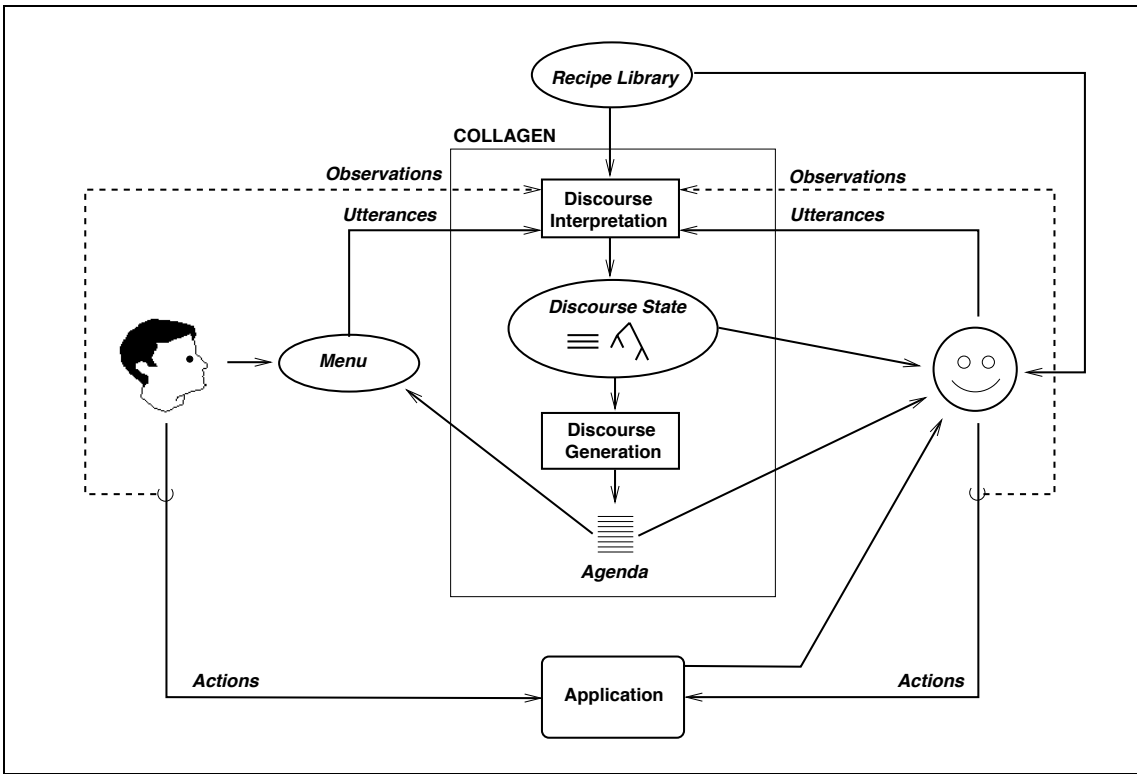


FIG. 7 – Architecture de Collagen

Ainsi, l'utilisateur ne peut pas effectuer des communications ou des actions de façon arbitraire mais seulement celles qui sont attendues par l'algorithme d'interprétation du discours. Le menu utilisateur de Collagen est généré automatiquement en fonction du discours. Si l'utilisateur choisit une des entrées du menu, cela devient une entrée du module d'interprétation du discours, bouclant un cycle d'exécution

Quelques remarques intéressantes concernant leur retour d'expérience,

- il est nécessaire de bien délimiter/faire comprendre à l'utilisateur où l'on en est dans la conversation : par exemple lorsqu'on passe à autre chose, l'agent dit : "OK, What's next ?", ce qui montre à l'utilisateur qu'il peut reprendre l'initiative.
- il est intéressant de permettre à l'utilisateur de ne pas se soucier des détails, de lui permettre de se reposer sur l'agent : si une information complémentaire est nécessaire, l'agent lui la demandera.

A partir de ces éléments, ils donnent des éléments de design :

- donner des pistes à l'homme pour lui indiquer les prochaines actions possibles
- pour savoir comment présenter une action à l'homme, 2 concepts :
  - garder la trace de ce que l'utilisateur sait déjà faire,
  - expliquer les fonctionalités à l'utilisateur lorsqu'elles sont nouvelles.

2.2.6 Engagement

Les derniers développements parlent d'engagement (*engagement*). L'engagement est alors défini comme une processus par lequel deux participants ou plus établissent, maintiennent et terminent . L'engagement est supporté par l'utilisation de la conversation , par la capacité de

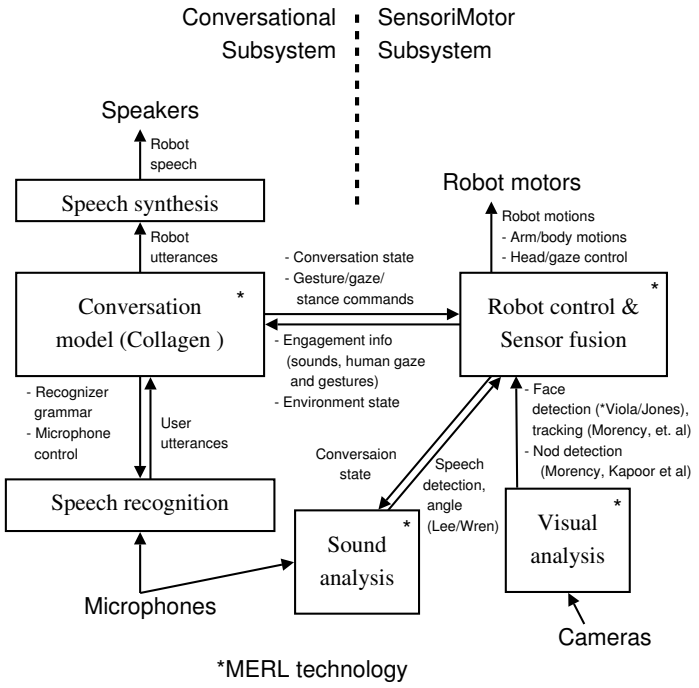


FIG. 8 – Collagen

collaborer dans une tâche , et par les gestes qui supporte la communication.

Ils considèrent que pour créer un robot montrant de l'engagement dans le cadre d'une interaction conversationnel, on doit donner un certain nombre de capacités au robot. Ils listent alors un ensemble de points clés (*key communicative capabilities*) incluant :

- des comportements d'engagement (*engagement behaviors*) : initier, maintenir ou se désengager d'une interaction,
- des capacités de gestion de conversation (*conversation management*) : turn taking, interprétation des intentions de son interlocuteur, établissement de relations entre les intentions et les buts des participants et relié les prises de paroles à l'état attentionnel de la conversation,
- des comportements collaboratifs (*collaboration behaviors*) : choisir quoi dire ou faire dans la conversation, entretenir les but collaboratifs partagés de l'homme et du robot, interpréter les contributions de l'homme (en terme de conversation et en terme de collaboration).

A partir des éléments du dialogue, Collagen extrait les intentions de l'homme et du robot et calcule les possible prochaines prises de paroles du robot.

A côté de ça, Collagen utilise un ensemble de règles dérivées des points clés définis précédemment pour déterminer les gestes que le robot doit faire et pour évaluer l'engagement de l'homme à partir des informations recueillies par le robot.

Ce système, et les deux sous-systèmes qu'il induit est décrit figure 8.

Du fait de ce découplage, ils éprouvent des problèmes dans la synchronisation de la parole et des gestes du robot (some robotic gestures must be synchronized with spoken language e.g. the beak movement must be timed to start and end of speech synthesis).

2.2.7 Analyse

Collagen est basé sur la théorie des SharedPlans. Il offre une étude intéressante de la prise en compte des besoins et des attentes de l'homme à partir de connaissances sur l'homme donnée au système. Ainsi mieux qu'une interface normale, le système peut prévoir les attentes et les actions futures à envisagées ce qui permet à l'homme d'être guidé.

Cependant, les premiers développements effectués sur un robot montrent la difficulté de combiner, d'entrelacer les actions et la communication au sein de ce système.

2.3 Collaboration, Dialogue, and HRI

Nous nous intéressons maintenant aux travaux de Fong sur le dialogue [FTB01].

2.3.1 Collaboration

Ils commencent par définir la collaboration comme un système où un homme et un robot travaillent en tant que partenaires, collaborant pour réaliser des tâches et des buts communs. Au lieu d'avoir un système maître-esclave, l'homme et le robot engagent un dialogue pour échanger des informations, poser des questions et résoudre les conflits. Une des possibilités que doit donner le système c'est que le robot peut décider ou non de suivre l'avis de l'homme, de même si l'homme n'est pas disponible pour répondre à la requête du robot cela n'empêche pas le système de fonctionner.

Pour cela ils se définissent des points clés :

self awareness : le robot doit être capable de détecter ses limitations et celles de l'homme, de déterminer si il doit demander de l'aide à un moment donnée et aussi savoir reconnaître quand un problème a été résolu,

self reliance : étant donné que le robot ne peut pas toujours compter sur l'homme pour résoudre ses problèmes, il doit avoir les capacités d'assurer sa propre sécurité,

dialogue : l'homme et le robot doivent être capable de communiquer de manière effective, chaque participant doit être capable de faire passer des informations, de poser des questions et de juger de la qualité des réponses reçues,

adaptive : le robot doit être capable de s'adapter à différents opérateurs et doit pouvoir s'adapter si besoin.

Ils donnent ensuite leur définition de l'interaction homme-robot comme étant l'étude des hommes, des robots et de la manière dont ils s'influencent l'un l'autre (*the study of the humans, robots, and the ways they influence each other*).

2.3.2 System Design

Ils ont implémenté leur architecture de contrôle comme un ensemble distribué de modules connecté via une architecture basé message (*message-based architecture*).

Modèle d'utilisateurs

Le modèle de l'utilisateur va contenir un ensemble d'attributs qui décrit un utilisateur ou un groupe d'utilisateurs. Parce que le dialogue est très dépendant de la tâche, les attributs associés au modèle de l'utilisateur varieront selon les systèmes et des modèles d'utilisateurs avec les caractéristiques suivantes :

– attributs (response accuracy, expertise, query interval) : ces attributs représentent les informations qu'il est nécessaire de connaître sur l'utilisateur,
– definition (novice, scientist, expert) : 3 stéréotypes d'utilisateur sont définis

Query Manager

Lorsque le robot a une question à poser à l'homme, il envoie un message au *QueryManager*. Un message est défini par les attributs de l'utilisateur, les attributs de la requête (type, niveau de priorité, délai d'expiration) , et des données associés à la requête (images, texte, etc.). Le système actuel supporte deux types de demandes : oui/non (*y/n*), l'utilisateur doit répondre oui ou non, et valeur (*value*), l'utilisateur doit donner une valeur décimale.

Le *QueryManager* stocke les messages arrivant dans une mémoire tampon, qui les stocke par ordre de priorité et selon leur date d'expiration. Quand cette mémoire contient des messages non-expirés, le *QueryManager* informe l'homme (par l'intermédiaire d'une interface graphique) qu'il y a des requêtes en attente. Ensuite quand l'homme indique qu'il est disponible pour répondre à une question, le *QueryManager* choisit un message dans la mémoire en filtrant selon les attributs de l'homme.

Les questions urgentes sont postées en priorité. Les questions qui ont expirées sont abandonnées. Une fois qu'une question est posée, le *QueryManager* attend pendant un intervalle de temps jusqu'à expiration du message.

Dialogue

Leur dialogue ne nécessite pas l'utilisation du language naturel, mais juste d'un language pertinent concernant la tâche et qui permet de faire passer les informations nécessaires, ce language et la grammaire associée seront limités à des questions concernant la mobilité du robot (e.g. navigation).

Ils utilisent une trentaine de messages permettant la téléopération du robot (translation, goto un point particulier). Le robot peut aussi envoyer des questions à l'homme par exemple en demandant si il est capable de franchir l'obstacle qui est devant lui ou pas. La réponse est caractérisée par son type (y/n ou valeur), par l'importance de la réponse, et par l'expertise nécessaire à la personne qui répondra à la question.

2.3.3 Discussion

Bénefice

La collaboration permet au robot et à l'homme d'utiliser leur compétence de manière complémentaire.

De plus, le robot connaissant la personne à qui il s'adresse choisi de manière dynamique de lui poser une question ou pas (en fonction de la capacité de la personne à bien répondre à la question par exemple).

Limitations

Tout d'abord, trouver les bonnes valeurs pour les paramètres du dialogue pour chacune des demandes associées aux tâches est difficiles.

Ensuite, ils indiquent qu'un des effets de bord du dialogue est que cela affecte la perception que les gens ont du robot conduisant souvent à une personification du robot et par là même à la construction d'un modèle cognitifs pour le robot qui est inadapté ou incorrect.

2.4 The Peer-to-Peer Human-Robot Interaction Project

L'objectif du projet *Peer-to-Peer Human-Robot Interaction* (P2P-HRI) [NF05, FNK+05] est de développer un ensemble de techniques pour permettre aux hommes et aux robots de collaborer comme des partenaires et cela à travers un ensemble de configurations variées : côte à côte, line-of-site remote, and far remote (over the horizon or even interplanetary distance).

Trois composants sont développés au sein de cette approche :

1. un nouveau cadre d'interaction baptisé *Human-Robot Interaction Operating System* (HRI/OS) basé sur un modèle collaboratif, il doit permettre aux hommes et aux robots d'avoir des dialogues orientés tâches et résolution de problèmes,
2. utilisation de *computational cognitive architectures* pour modéliser le comportement de l'homme et faire en sorte que le robot soit plus compréhensible pour permettre une meilleure interaction avec l'homme, dans ce cadre nous n'étudierons que le système de raisonnement sur l'espace qu'ils y ont développé,
3. développement d'un ensemble de métriques pour la mesure de performance du système, sur lesquels nous ne reviendrons pas.

Leur approche est de développer un modèle d'interaction dans lequel les hommes et les robots communiquent d'égal à égal (*as peers*). Ils ont développé un système de dialogue qui permet aux robots de poser des questions à l'homme quand cela est nécessaire et à des moments appropriés, pour permettre aux robots d'obtenir une assistance de l'homme.

Un point clé de leur approche est de permettre aux hommes et aux robots de se coordonner par l'intermédiaire du dialogue. Cela permet de maintenir la connaissance du contexte et de la situation au sein de l'équipe.

Les deux points clés :
- permettre aux hommes et aux robots de communiquer et de coordonner leurs action et
- fournir le support à l'interaction pour que les hommes et les robots puissent répondre et s'aider mutuellement de manière rapide.

L'idée est de permettre au robot de réaliser les tâches par eux mêmes mais de leur donner la possibilité de poser des questions et par là même d'utiliser l'expertise de l'homme quand cela est nécessaire. Un autre challenge est de permettre aux robots de comprendre des commandes "orientées tâches" comme le font les hommes (e.g. "move *that* panel to *my* left").

2.4.1 Human-Robot Interaction Operating System

Approche

HRIOS [FKHB06, FSS+06] a été conçu pour permettre la réaliser de tâches opérationnelles (*operational tasks*) i.e. des tâches concrètes, bien définies, de portée limitée (*narrow in scope*) et qui relève de l'interaction homme-robot (*amenable to joint human-robot performance*).

Dans le cadre d'exploration spatiale dans lequels ils se placent, les tâches opérationnelles incluent : la construction de hangar ou d'abris, l'assemblage de tuyaux, diverses tâches d'inspection, etc.

Design

Une architecture basée agent HRI/OS est un système multi-agents, les agents pouvant être incarnés (les hommes et les robots) ou des agents logiciels comme on le voit sur la figure 9. Les agents représentant les hommes et les robots agissent dans le monde "réel" et décrivent

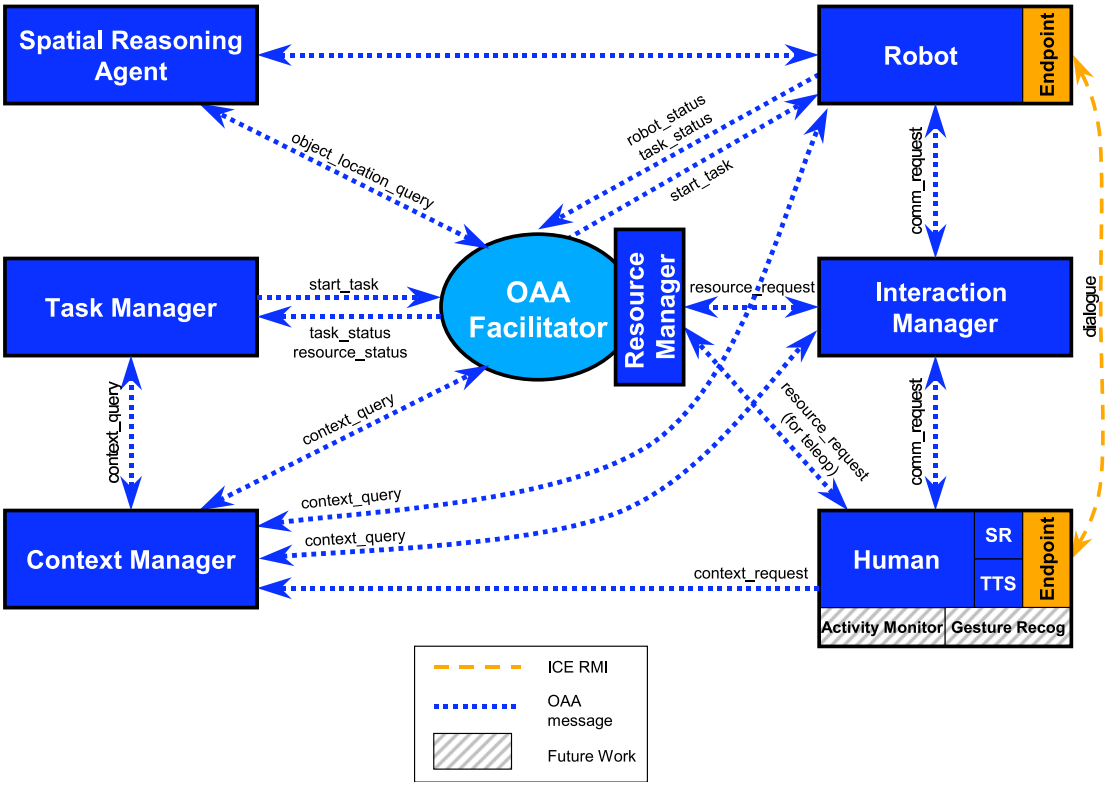


FIG. 9 – HRIOS

leurs capacités à un niveau grossier alors que les agents logiciels n'ont aucune influence sur l'environnement et ne fournissent qu'un ensemble limité de services.

Task Manager Le *task manager* (TM) coordonne et gère l'exécution des tâches opérationnelles en décomposant le but du système en tâches de haut-niveau qui sont ensuite assignées aux hommes ou aux robots pour execution. Une seule tâche est assignée à un agent à un même moment.

Le TM n'a pas a se soucier des détails de réalisation de la tâche, ce sont les agents qui gèrent l'exécution des tâches qui leurs sont assignées.

Dans leur système ce n'est pas l'agent qui fait la demande à un agent spécifique qu'il aura choisi comme étant le plus compétent pour cette tâche, mais la requête est envoyée à un système centralisé qui LUI choisit à qui est envoyé la requête en fonction des contraintes et des connaissances de chacun.

Task Representation

La représentation des tâches est faites en TDL ([SA98]) comme on peut le voir figure 10. Cette représentation permet de représenter des contraintes inter-tâches.

Task Assignement

Pour attribuer une tâche, le *Task Manager* contacte le *Resource Manager* pour trouver un agent capable de réaliser la tâche. Si le *Resource Manager* attribue la tâche à un agent, le *Task Manager* peut alors débuter le monitoring du statut de la tâche. Si le *Resource Manager* ne trouve pas d'agent disponible, il en informe le *Task Manager*, dans ce cas, le *Task Manager* va attendre qu'un agent soit disponible et réitérera sa requête.

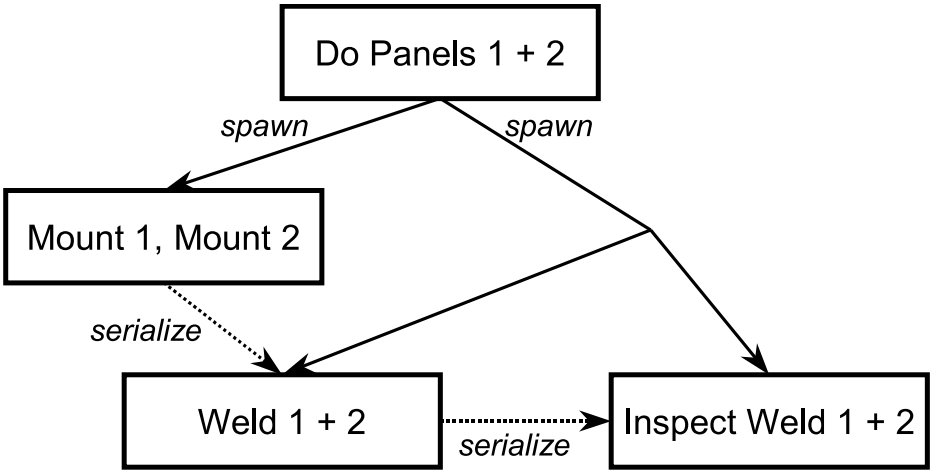


FIG. 10 – HRI-TDL

Management Le *Task Manager* est conçu pour faire de la reprise d'erreurs. Si une tâche échoue, le *Task Manager* va en créer une autre instance pour essayer une nouvelle fois de la réaliser. Il est capable de réagir à des feedbacks prédéfinis des tâches via task monitoring.

Resource Manager Le *Resource Manager* est chargé d'attribuer les tâches aux agents. Cette attribution se fait par le biais d'une rechercher multi-critères, ainsi il ne vérifie pas seulement la capacité de l'agent à réaliser l'action mais également un ensemble de facteurs tels que disponibilité, la position actuelle, la charge de travail, les performances de l'agent, etc.

Interaction Management L'*Interaction Manager* coordonne l'interaction (*dialogue-based interaction*) entre les agents. L'*Interaction Manager* donne à chaque agent la possibilité de communiquer avec les autres agents : pour demander de l'aide, pour en fournir, etc. Pour le moment, HRIOS inclut des interfaces graphiques et vocales mais il est prévu d'intégrer d'autres modalités.

Un agent peut envoyer une requête à l'*Interaction Manager*. L'*Interaction Manager* va alors demander au *Resource Manager* une liste d'agent capable de répondre à la requête et ensuite contacte le premier. Quand un agent répond à une requête, il est ensuite de la responsabilité des deux parties de gérer le reste de la communication intervenant entre les deux agents. Il n'est fait appel à l'*Interaction Manager* seulement si l'agent ayant émis la requête lui notifie que la communication a échoué. Dans ce cas, l'*Interaction Manager* tentera de connecter l'agent à un nouvel agent pour répondre à sa requête.

Robot Agent C'est l'interface entre un robot et HRIOS. Le *Robot Agent* gère les requêtes qu'ils reçoient des autres agents, l'exécution des tâches et le dialogue avec les autres agents.

Les tâches de haut-niveau sont allouées au robot par le *Task Manager*. Quand une tâche lui est attribuée, le *Robot Agent* décompose la tâche de haut-niveau en primitives qui pourront être exécutés. Lorsque le robot début l'exécution de ces primitives, il ne peut être interrompu ou s'engager dans un dialogue qu'à des moments précis définis comme des *break point*. Un *break point* est un moment de l'exécution où :

1. la tâche peut être reprise sans nécessité de préservation de l'état ou du contexte,
2. la tâche ne peut pas ne pas reprendre à cause d'une erreur ou d'un échec,
3. le robot ne fait rien.

Le *Robot Agent* dispose de méthodes pour poser des questions aux hommes et recevoir les réponses. Quand le robot a une question à poser, il envoie un message à l'*Interaction Manager*. Un message est définit par :

- les attributs de la demande (*query attributes*) : niveau de priorité, date d'expiration, etc.
- le type de la demande (*query type*) : oui/non, question à choix multiples, etc.
- les spécificités de la demander (*message specific data*) : image, texte, etc.

Les fonctions du *Robot Agent* sont les suivantes :

- au démarrage : le *Robot Agent* liste ses capacités au *Ressource Manager*
- en cours d'exécution : *Robot Agent* broadcast périodiquement des messages (état du robot, progrès de la tâche, etc) aux autres agents,
- *Robot Agent* gère le changement de mode entre "system operation" (Task Manager driven) et "user mode" (homme téléopère directement le robot)

Human proxy Les agents représentant les hommes, de la même manière que ceux des robots publient la liste de leurs capacités, leurs domaines d'expertise (définissant ce qu'il est possible de leur demander via le dialogue) et fournit provide health monitoring feedback that can be used by other agents to track the overall progress of the task.

Les *Human proxy agents* utilise des interfaces utilisateur pour communiquer avec les usagers qu'ils représentent, et utilise l'*Interaction Manager* pour la gestion du dialogue avec les autres agents.

HRIOS Execution

Dans les exemples présentés ci-dessous et figure 11 et 12, les hommes peuvent travailler au côté ou de manière distante avec deux robots : le rover K-10 et le Robonaut-B La tâche consiste à souder des panneaux pour renforcer une structure.

Robot supports human EVA-1 utilise HRIOS pour demander qu'une lumière soit pointée sur le coin sur lequel EVA-1 doit effectuer sa soudure (cf figure 11). La requête de l'homme passe initialement à l'*Interaction Manager* et au *Ressource Manager*. Le *Ressource Manager* décide que le robot K-10 est le plus adapté pour réaliser la requête. Quand K-10 arrive à un breakpoint dans sa tâche actuelle, il s'associe à EVA-1. Tant que K-10 et EVA-1 sont associés, K-10 répond aux requêtes de EVA-1. Quand cela est terminé, K-10 reprend l'exécution de sa tâche précédente.

Ce qu'ils affirment être un des points clés de leur approche, c'est que l'équipe EVA-1 - K-10 est formé dynamiquement et reste formé uniquement le temps nécessaire.

Human supports robots L'une des caractéristiques de HRIOS est qu'il permet à l'homme d'apporter son aide au robot. Par exemple, le robot veut que la soudure qu'il a faite soit vérifiée (cf figure 12). Pour cela, K-10 demande "is this seam okay" accompagné d'une image de la soudure. Cela est envoyé à l'*Interaction Manager* et au *Robot Manager*. Ce dernier décide quel homme est le plus à même de répondre à la requête. Après avoir vu les données envoyées par le robot, l'homme décide qu'il a besoin d'information complémentaire. Il demande de prendre le contrôle de K10. K10 suspend alors sa tâche lorsqu'il arrive à un breakpoint, l'homme téléopère le robot (par exemple en bougeant la caméra de K10 pour obtenir d'autres points de vue de

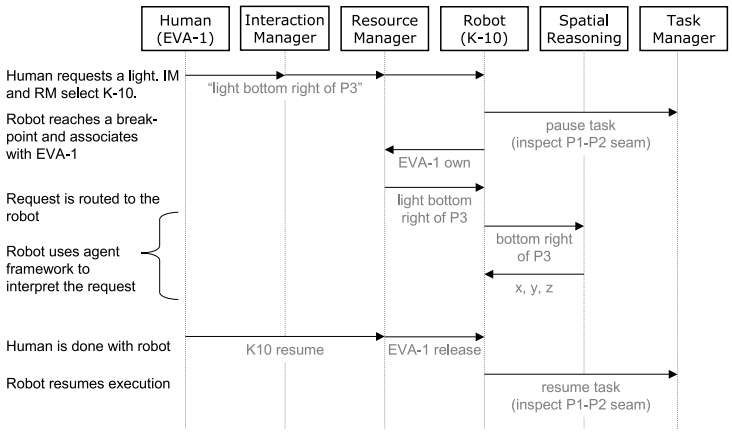


FIG. 11 – HRIOS

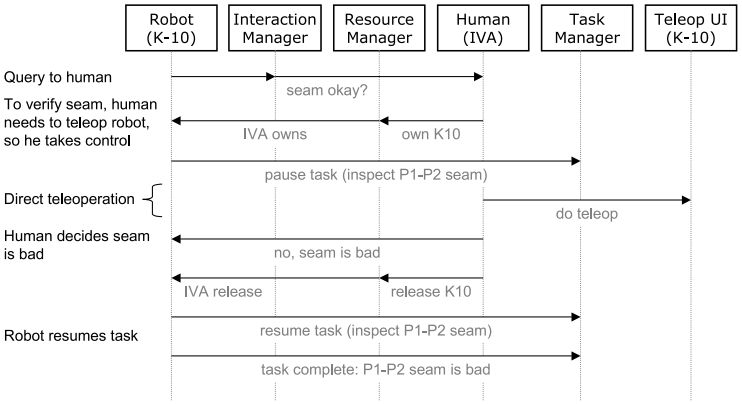


FIG. 12 – HRIOS

la soudure). L'homme décide que la soudure est de mauvaise qualité, il répond à la question. L'association entre l'homme et le robot est alors terminé et le robot reprend sa tâche.

On note ici la possibilité donnée à l'homme de prendre le contrôle du robot.

2.4.2 Spatial reasoning agent

Lorsque l'homme et le robot doivent partager un espace en commun, le robot doit être capable de comprendre comment l'homme perçoit l'espace et les positions des objets qui l'entourent.

Cela a à voir avec le "perspective taking" [NF05, TCB+05], i.e. la capacité pour un agent de prendre la perspective d'un autre agent.

Pour cela, ils ont développé "*spatial reasoning agent*" qui doit permettre de résoudre les ambiguités lors d'un dialogue homme-robot. Lorsqu'un dialogue homme-robot fait intervenir un language ayant trait à la gestion de l'espace (*spatial language*), HRIOS transmet la requête au SRA sous la forme : (speaker, addressee, type of command, reference objects, and frame of reference), e.g. (astronaut, Robonaut, move, Box1, left, ego). Le SRA transforme alors le dialogue dans l'espace en une référence géométrique en utilisant le modèle cognitif. Pour cela, comme indiqué figure 13, ils effectuent une simulation "mentale" de chacun des éléments. Ensuite un repère de référence est choisi entre :

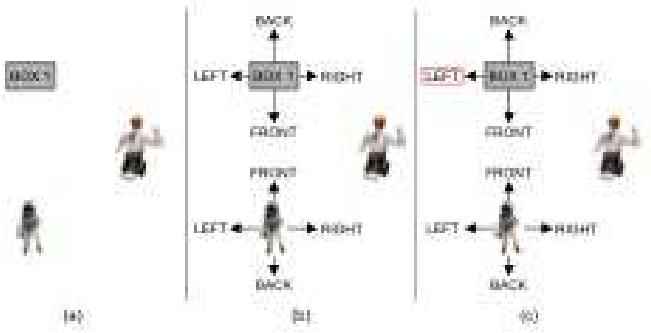


Figure 5: Disambiguation process: (a) Configuration of mental simulation in Stage, (b) Assignment of frames of reference relevant objects, (c) Resolution of ambiguous location.

FIG. 13 – HRIOS-SRA

– ego : repère egocentrique qui peut être appliqué aux agents et aux objets,
– referent : le repère de référence est utilisé quand :
  1. l'objet et l'agent auquel il est fait référence ne dispose pas de son propre repère,
  2. un repère qui est le miroir du repère de référence d'un autre agent doit être utilisé,
– exo : le repère exocentrique représente le repère monde.

Enfin, la requête est exécuté en fonction des paramètres et du modèle du monde ainsi construit.

2.4.3 Analyse

HRI/OS présente un système ambitieux, essayant de prendre en compte la globalité du problème. Il définit pour cela des agents robots et des agents humains (représenté sous la forme de proxy comme on l'avait déjà retrouvé dans teamwork) qui sont pilotés par un Task Manager (gérant l'exécution des tâches), un Resource Manager (gérant l'attribution des tâches) et un Interaction Manager (gérant la communication entre les agents).

Cependant leurs besoins ont aboutit au fait que toutes les demandes doivent passer par un composant central avant d'être assigné à un agent particulier, même si ensuite la collaboration semble se faire de manière plus directe. Ainsi une tâche n'est pas géré au niveau d'un agent mais par une entité central qui aura connaissance des différents éléments du système.

2.5 Apprentissage et Collaboration

Nous allons maintenant étudier les travaux de Breazeal [BHL04, BBG+04] qui utilisent la théorie de l'intention jointe pour intégrer l'apprentissage et la collaboration et permettre au robot d'apprendre des structures de tâches. Une fois apprises, le robot doit être capable de reproduire les tâches et de collaborer avec l'homme en utilisant des actes de communication. Le but de l'étude est de créer des robots qui pourraient coopérer avec les humains à la manière de partenaires avec lesquels nous pourrions avoir des activités et non pas des robots semi-autonomes qui seraient pilotés par des hommes.

L'approche modélise l'apprentissage comme un processus collaboratif qui se déroule au sein d'une interaction entre un professeur et un élève.

De même que les robots ont besoin de savoir-faire sociaux pour collaborer avec les gens à la manière d'un coéquipier, ils doivent comprendre nos intentions, croyances, désirs et buts pour leurs permettre d'accomplir la bonne action au bon moment. Pour qu'une équipe homme-robot réussisse, les équipiers doivent communiquer leurs propres intentions et buts pour établir et maintenir l'ensemble des croyances partagées et coordonner leurs actions pour exécuter leurs plans partagés. Cela définit la collaboration (*collaboration*) plutôt que l'interaction.

Dans le cadre de la réalisation d'un tel robot coopératif, plusieurs défis sont posés :

Knowing what matters Savoir reconnaître les signaux pertinents, reconnaître ce qui est important au milieu de l'ensemble des données perçues, est un processus fondamental qui peut être guidé de manière externe (par l'environnement) ou de manière interne (par les buts du robot, etc) ou socialement (par l'homme). De même, le robot doit rendre son état visible à son partenaire, i.e. lui permettre de savoir ce que le robot attend et ce qu'il va faire.

Knowing what action to try Une fois que le robot a reconnu les signaux pertinents, comment déterminer les actions à réaliser ? Un processus de collaboration peut permettre à l'homme de guider le robot à travers son processus de sélection d'actions dans le cadre d'un apprentissage.

Knowing how to recognize success and correct failures Une fois que le robot a réalisé l'action, il s'agit de savoir si elle est réussie ou non, et en cas d'insuccès de savoir déterminer ce qui n'a pas fonctionné. Pour faciliter cette évaluation, il est possible que l'homme donne un certains nombre de feed-back au robot tout au long du déroulement de la tâche. Cela peut se réaliser par l'intermédiaire de synchronisation (de communication) permettant de maintenir une connaissance partagée au cours du déroulement de la tâche. D'un côté l'homme doit être capable de confirmer que le robot a correctement compris et de l'autre le robot doit réaliser la bonne action pour arriver à l'issue finale de chacune des actions, tâches, sous-tâches.

Knowing how to explore L'apprentissage et la réalisation de la tâche elle-même ne sont pas vus ici comme des activités séparées. Le processus d'apprentisage est toujours présent. En ce sens la collaboration peut être vue comme un processus permettant d'affiner la performance et la compréhension de la tâche. L'idée est que l'homme entre très tôt dans la boucle d'apprentissage de la tâche par le robot pour inférer rapidement les préconditions et les résultats espérés pour chacune des étapes d'autant que chacune des étapes sont reliées au sein d'une structure globale de tâche. Cela doit permettre de corriger les erreurs au plus vite.

2.5.1 Théories utilisées

L'étude est basée sur la théorie de l'intention jointe et la théorie du "*situated learning*".

En ce qui concernera la théorie de l'intention jointe, on peut répéter qu'une collaboration efficace et robuste nécessite une communication continue (*an open channel of communication*). De plus, un comportement coopératif est un processus au long-cours pour lequel il est nécessaire de maintenir des croyances mutuelles, de partager des connaissances pertinentes, de coordonner des actions, de montrer son engagement à réaliser sa part et d'aider les autre à effectuer la leur et ainsi compléter la tâche jointe.

FIG. 14 – Le robot Leonardo

En ce qui concerne la théorie du "*situated learning*", dans ce mode d'apprentissage, l'éducateur doit maintenir un modèle de ce que l'élève a déjà accompli, de ses croyances actuelles concernant la tâche. L'éducateur doit savoir ce qui est clair ou non pour envoyer le bon feedback et modifier la structure de la tâche si nécessaire. Ainsi par l'intermédiaire d'une interaction réciproque l'élève et son professeur coopèrent pour (i) aider l'instructeur à garder un bon modèle mental de l'élève, et (ii) aider l'élève en le guidant pour qu'il construise les bons modèles de la tâche et des représentations associées.

2.5.2 Plate-Forme et Modélisation

Le robot Leonardo (figure 14) sur lequel est basé l'étude est équipé :
- d'un système de reconnaissance de la parole lui permettant de reconnaitre des actions (grasping, pressing, looking, etc.), des entités (buttons, people, etc.), des caractéristiques (color, button-ON, button-OFF, shape, size, etc.), ainsi que des gestes.
- d'un système de vision lui permettant de voir les gens devant lui, les objets et de reconnaître des gestes de pointage (*pointing gestures*) dans le repère du robot.

A partir des ces données à été construit un système d'attention jointe. Ce système observe le focus attentionel du robot et de l'homme et utilise ces données pour suivre l'évolution des croyances mutuelles et ce qui est l'objet de la focalisation à un moment donné. Le robot ne modélise pas seulement son propre état attentionel mais également celui de l'homme.

Ce système attentionel modélise un niveau de pertinence, d'intérêt (*level of saliency*) pour chacun des objets et des événements perçus dans la scène. Trois facteurs contribuent à cette valeur :
- propriété perceptuelle (*perceptual properties*) : proximité du robot, couleur, mouvement, etc.
- état interne du robot (*internal state*) : ce que le robot est en train de faire,
- référence socialement dirigée (*socially directed reference*) : pointer vers, regarder vers, parler de quelque chose

Pour chacun des composants présents dans l'espace 3D du robot, un niveau d'intérêt global est calculé qui sera la somme pondérée de chacun de ces facteurs. De la même façon, le robot modélise ce que l'homme regarde, ce dont il parle, ce qu'il montre. Les objets ainsi désignés sont taggés avec une valeur de pertinence qui indique que ces objets ont été au centre l'attention de l'homme à un moment donné. Cela permet de trouver les points de convergence entre ce que le robot et ce que l'homme ont perçu, ces points de convergence vont modéliser le "*common ground*".

Ensuite des politiques de conversation ont été développés pour accompagner la mise à jour et le maintien de ces connaissances partagées. Des fonctions clées du discours ont été définies. Elles recouvrent l'utilisation de marqueurs du discours, des procédures d'élaborations (quand l'instructeur pense que l'élève a mal compris), de clarifications (quand l'élève a mal compris), de confirmations (pour s'assurer qu'une étape a bien été franchie). Ces fonctions peuvent être réalisées par l'intermédiaire de la parole mais également par l'intermédiaire de gestes, d'expressions faciales... A partir de là, a été élaboré un système de "*turn-taking*" auquel a été ajouté des signaux paralinguistiques (*paralinguistic social cues*) comme regarder quelqu'un quand on veut lui parler par exemple et aussi des *"back channel signals"* pour permettre à la personne qui écoute de faire comprendre au locuteur qu'il a été compris et si la communication est toujours en cours ou non.

2.5.3 Apprentissage collaboratif de tâches

Les tâches sont représentées sous la forme d'une structure hiérarchique d'actions et de sous-tâches et d'une représentation du but global de la tâche. Un *task manager module* maintient une collection de modèles de tâches connues et des noms qui leurs sont associées. A partir de cet ensemble de tâches, on peut demander au robot des tâches du type : "Léo réalise la tâche x". Si le robot ne reconnaît pas la tâche qu'on lui demande ou si il ne sait pas comment la réaliser il semble perdu et soulève les épaules l'air de dire : "je ne sais pas".

Dans le cas d'une tâche inconnue, l'homme peut proposer au robot de lui apprendre la tâche.

Une nouvelle tâche se construit à partir de l'ensemble des actions et tâches déjà connues. Pendant l'apprentissage le robot enregistre l'action qu'on lui a demandé de faire et encode les buts qui sous-tendent ces actions. Pour coder l'état du but d'une action ou d'une tâche qu'il aura réalisé, le robot compare l'état du monde avant et après l'exécution. Cela induit la production d'une hiérarchie avec un but défini pour chacune des parties individuelles de la tâche ainsi qu'un but pour la tâche globale.

Ainsi le but d'une tâche est défini par un ensemble de croyances sur ce qui doit être vrai ou faux dans le monde pour que cette tâche soit réalisée.

2.5.4 Réaliser la tâche apprise en collaboration avec l'homme

Un mécanisme de "*turn-taking*" a été implémenté dans lequel le collaborateur humain et le robot travaillent comme des partenaires pour réaliser un but. Cela est réalisé par une évaluation continue d'une part de l'état de la tâche, d'autre part de l'état du monde avant la réalisation de chacune des actions. Un robot réalisant une tel tâche doit être doté de mécanismes pour analyser le monde, évaluer l'état des buts et combiner ces informations pour choisir les actions adéquats.

Cependant, le système doit également assigner de manière dynamique les tâches entre les membres de l'équipe. Ainsi, avant d'essayer de réaliser une action, le robot décide qui doit la réaliser. Typiquement pour le moment, le robot détermine si il peut réaliser l'action ou non. Si il ne peut pas la réaliser, il demande de l'aide.

Le robot peut également suivre des actions simultanées, i.e. l'homme réalise une action alors que le robot effectue une autre action. Dans ce cas, la contribution de l'homme est prise en compte et il y a une réévalution de l'état de la tâche. Selon sa décision, il devra en informer l'homme pour maintenir la croyance mutuelle sur l'état de la tâche globale.

Un certains nombre de possibilités sont laissées au robot pour communiquer son état au cours la collaboration, par exemple en montrant qui doit réaliser une action ou si il pense qu'une action est terminée.

2.5.5 Analyse

Dans ce système, les auteurs considèrent la collaboration plutôt que l'interaction Le robot communique de manière continue son état à son partenaire humain pour maintenir une croyance mutuelle concernant la tâche ce qui rend la tâche d'apprentissage plus efficace. Pour cela, ils ont développé un système basique de "*turn-taking*" qui contient la gestion de signaux paralinguistiques et des "*back-channel signals*" permettant la prise en compte de la compréhension.

L'idée est celle d'un partenariat où l'homme et le robot maintiennent et travaillent ensemble sur des buts communs. Pour cela le robot doit non-seulement suivre ce que la personne (son équipier) est en train de réaliser mais également avoir une idée de ses intentions et de ses buts. Dans ce cadre, leur idée est que le robot ne modélise pas seulement son propre état mais également celui de l'homme.

Leur représentation de la tâche permet au robot de raisonner sur la tâche à plusieurs niveaux, permettant de partager facilement le plan avec le partenaire et de réaliser des ajustements lors de changements de l'état du monde. Etant donné que l'homme et le robot agissent au sein d'un même environnement, leur système prend en compte les changements intervenus à la suite d'actions de l'homme et met à jour le plan en conséquence. Il est aussi capable de prendre en compte des actions simultanées de l'homme et du robot.

Comme nous le verrons par la suite, ce système développe les idées les plus proches de notre travail.

2.6 Autres systèmes

Dans [HB07], Hoffman et Breazeal font la remarque intéressante que pour collaborer, les agents doivent non-seulement prendre en compte les événements passés et l'état actuel des perceptions du robot, mais également les attentes et les réponses escomptés du collaborateur avec qui s'effectue la tâche. Ainsi ils ont développé un système où le robot anticipe les actions des autres agents pour montrer une meilleure réactivité. Dans le cadre d'une tâche de construction, le robot peut alors prévoir de passer un outil à l'homme si il sait qu'il doit l'utiliser.

Feil-Seifer et Mataric [FSM05] introduisent une modélisation de la théorie de l'intention jointe pour une tâche consistant à définir si un homme souhaitait ou non collaborer avec le robot. Le robot devait alors choisir si il devait proposer ou non une interaction à l'homme.

Des recherches se développent également sur l'utilisation de la collaboration pour l'apprentissage de tâches. Comme nous l'avons vu, Breazeal [BHL04, BBG$^+$04] utilise la théorie de l'intention jointe pour intégrer l'apprentissage et la collaboration et permettre au robot d'apprendre des structures de tâches. Une fois apprises le robot doit être capable de reproduire les tâches et de collaborer avec l'homme en utilisant des actes de communication. Dominey [Dom07b, Dom07a] étudie l'imitation dans le contexte d'une tâche collaborative en se basant sur des études sur le développement de ces comportements chez l'enfant.

Conclusion

Les différentes théories présentées posent de manière claire qu'une tâche effectuée de manière collaborative n'a pas les mêmes exigences qu'une tâche individuelle ou même participative.

Nous avons vu que dans plusieurs cas étudiés, e.g. teamwork et HRI/OS, la fonction du système consistait pour le robot à trouver le bon niveau d'autonomie en fonction du contexte. Nous étions dans un contexte d'autonomie ajustable telle que défini par [MTM03] "agents dynamically varying their own autonomy, transferring decision making control to other entities (typically human users) in key situations". La question est alors de savoir quand effectuer le transfert de contrôle d'un agent (e.g.le robot) vers un autre agent (e.g. l'homme). Or nous constatons que dans notre contexte la question est substantiellement différente et consiste plutôt à déterminer si il faut et quand il faut que le robot interagisse avec l'homme et si nécessaire prenne des initiatives envers lui. L'homme et le robot vont partager un même environnement et vont être amené à exécuter des tâches en interaction l'un avec l'autre, ils vont devoir gérer un "*turn-taking*" et maintenir un ensemble de croyances et devenir des partenaires comme nous l'avons vu avec l'apprentissage collaboratif chez Breazeal. La question est alors d'équiper le robot avec des capacités de décision et d'action lui permettant d'exécuter des tâches en interaction avec l'homme. Nous adoptons ce concept que nous nommerons : interaction ajustable. Cela nous a poussé dans la définition d'une architecture générale munie d'un noyau décisionnel de supervision et de plans d'action et de communication prenant en compte l'intentionalité et l'acceptabilité. Dans ce cadre, nous considérons non seulement comme importante la décision d'agir mais également la manière d'agir.

Nous avons vu également avec SharedPlans et Collagen, la difficulté de combiner, d'entrelacer la réalisation d'une action et la communication nécessaire à son exécution. Nous allons voir comment nous traitons ce problème en essayant de coupler à chaque tâche un modèle de la communication nécessaire.

Enfin, la plupart des systèmes "traduisent" l'homme sous forme d'un agent logiciel, l'homme mettant à jour cet agent logiciel utilisé ensuite par le robot. Dans notre travail, nous avons défini que l'ensemble des connaissances du système était porté par le robot et que par conséquent, elles étaient filtrées (et parfois même biaisées si les systèmes de perception le sont) par la perception du robot i.e. elles sont dépendantes des capacités de perception et d'interprétation du robot.

Deuxième partie

Un cadre pour la réalisation de tâches collaboratives en robotique

Introduction

Dans cette partie, nous allons nous intéresser au cadre que nous avons défini pour la réalisation de tâches collaboratives en robotique. Dans un premier chapitre, nous parlerons de l'architecture générale développée pour prendre en compte l'interaction au sein de l'architecture LAAS. Ensuite, nous approfondirons l'étude du système d'exécution et de supervision de tâches Shary ("Supervision for Human Adapted Robot I(Y)nteraction") que nous avons développé. Enfin, nous présenterons une manière de modéliser la communication et l'étude que nous avons effectué concernant la liaison de notre système à un système de dialogue.

Chapitre 3

Une architecture de contrôle pour un robot interactif

3.1 Introduction

3.1.1 Supervision de tâches en robotique

La réalisation d'une tâche en robotique est dépendante d'un contexte. Ce contexte et sa perpétuelle évolution nécessite une adaptation constante du comportement du robot. Superviser une tâche robotique c'est potentiellement déterminer toutes les issues possibles des actions, tous les contextes envisageables et prévoir le traitement à réaliser dans chacun des cas. Le but est de rendre le robot autonome, i.e. si l'on demande au robot de réaliser une tâche, il l'exécute et entreprend si besoin toutes les procédures d'adaptation nécessaires. Le résultat pourra être un succès si la tâche à accomplir est terminée à l'issue des actions mises en route par le robot ou un échec si le robot conclut, selon ses capacités et l'état de l'environnement que la tâche est irréalisable ou encore si un cas non-prévu ou ingérable se produit. Dans tous les cas, le système de supervision devra, dans la mesure du possible, s'assurer de laisser le système dans un état dénué de danger pour le robot lui-même et les personnes qui l'entourent, e.g. arrêter les mouvements d'un bras manipulateur et le remettre en position sûre, arrêter les moteurs et mettre les freins si l'on doit arrêter la base.

3.1.2 Introduction de la collaboration

Lorsque la tâche est de nature collaborative, c'est à dire qu'elle engage non seulement le robot mais aussi une autre personne (ou même un autre robot), de nouvelles exigences apparaissent. En effet, à côté de la réalisation de la tâche elle-même, vont devoir se produire un ensemble d'actions reliées à la mise en place de la tâche, à sa conduite et à sa terminaison. Prenons par exemple la tâche *GiveObject(ObjectName,PersonName)* pour le robot Jidowanki, tâche pour laquelle le robot doit donner l'objet *ObjectName* à la personne *PersonName* se trouvant à sa proximité. Nous pouvons décrire cette tâche d'une manière "robot-centrique" où le robot va devoir suivre un ensemble prédéfini d'étapes : tendre l'objet à la personne, attendre que la personne prenne l'objet, remettre le bras à sa position initiale. Réalisant cette suite d'actions, la tâche *GiveObject* serait réalisée. Cependant si la tâche est définie de cette manière, que se passe-t-il si l'homme ne suit pas exactement le protocole défini par l'enchaînement des tâches, e.g. si la personne ne prend pas l'objet, le robot attend il ou abandonne-t-il la tâche ? Qui impose le rythme ? Comment l'homme est-il averti étape par étape des intentions du robot ? Le robot exhibe-t-il

de telles intentions ? Dans ce cas, la tâche est réalisée en boucle ouverte sans que l'on donne au robot la possibilité d'avoir ou de percevoir un retour d'information de la part de l'homme et de manière symétrique de donner des informations en ce sens à l'homme.

Ainsi, à côté de la gestion du processus d'évolution de la tâche, il faut gérer le processus d'évolution de l'interaction. Ces processus ne sont pas disjoints. Par exemple, prendre en compte la position, l'attitude de l'homme lors de la planification de la trajectoire du bras, et adapter constamment la vitesse et l'accélération lors de sa réalisation participent de ces deux composantes.

La théorie de l'intention jointe décrite à la section 1.1 page 25, énonce que deux agents ont un but commun persistant (ou *joint persistent goal*) si ils ont la croyance mutuelle :

- que le but n'est pas encore réalisé,
- qu'ils souhaitent que le but soit réalisé,
- que tant qu'ils ne possèdent pas la croyance commune que le but est terminée (*achieved*), irréalisable (*unachievable*) ou n'est plus pertinent (*irrelevant*), le but ne peut être abandonné.

Cette dernière condition indique qu'il est de la responsabilité des agents engagés dans le but d'assurer la communication pour atteindre la croyance commune concernant l'état de la tâche. Étant donné que nous ne pouvons nous engager que sur les actes effectués par le robot [6], i.e. nous ne pouvons nous engager sur l'éxécution d'un acte par un autre agent, nous analyserons ce que le robot doit faire pour satisfaire cette définition.

Concernant notre tâche *GiveObject*, le minimum de communication qu'il faut réaliser va consister à s'assurer que la personne est prête à interagir et à la prévenir lorsque la tâche est terminée. Ainsi lors de l'exécution d'une tâche en collaboration avec l'homme, le robot doit non-seulement avoir les moyens de réaliser la tâche mais aussi de gérer le flot d'interaction pour communiquer ses intentions et ses croyances et de monitorer l'activité de son partenaire. Ceci de manière analogue au cas du dialogue [Cla96] où l'on peut distinguer le dialogue au sein de la tâche elle-même et la communication nécessaire à sa gestion.

Communiquer c'est partager l'information, obtenir des croyances partagées va conduire à la communication par l'intermédiaire d'actes de communication. La communication va pouvoir se faire par le dialogue, i.e. la communication verbale, mais également par l'intermédiaire de toutes les possibilités de communication disponibles : gestes, mouvements, affichage, etc. De plus comme nous l'avons vu avec Clark page 40, contribuer à la communication ne doit pas se restreindre à la *negative evidence* - évidence que nous n'avons pas été entendu ou compris, mais aussi à la *positive evidence* par l'obtention d'accusé de reception (*acknowledgement*), de (*relevant next turn*) ou par une attention continue (*continued attention*).

La communication est un processus bilatéral. Dans notre cas elle va considérer deux acteurs : le robot et l'homme. La communication peut être à l'instigation de l'un ou de l'autre de manière indifférente.

3.1.3 Challenges

Le contexte de l'interaction homme-robot nécessite la prise en compte explicite de l'homme à tous les niveaux. Il va notamment impliquer des communications entre l'homme et le robot pour partager les connaissances concernant la tâche à réaliser, en cours de réalisation ou réalisée. Pour exécuter une tâche collaborative notre système de supervision va devoir réaliser non seulement la tâche elle-même mais devra également monitorer l'activité de l'homme pour pouvoir détecter

[6] même si nous considérerons que l'homme réagit de manière "habituelle"

l'apparition d'actes de communication et produire la réponse appropriée. Par exemple, dans notre tâche *GiveObject*, si lorsque le robot tend le bras pour donner l'objet à l'homme, celui-ci détourne la tête (e.g. commençant à parler à une autre personne) le robot doit pouvoir interpréter cette perte d'attention comme une communication (en l'occurrence une perte de communication) et la traduire par exemple comme une suspension de la tâche en cours et agir en conséquence en arrêtant le mouvement du bras tant que l'homme ne regarde pas en direction du robot. Il est donc nécessaire de prendre en compte l'homme au niveau décisionnel.

Nous ne pouvons restreindre la communication à un schéma prédéfini et statique. Par exemple, la communication peut être plus ou moins équilibrée entre les interlocuteurs selon qu'il y ait ou non ou leader à la tâche, ou elle peut être plus ou moins nécessaire selon les connaissances qu'ils ont l'un de l'autre.

Notre ambition est de permettre la définition d'un tel schéma de manière adaptée à la tâche (et à chacune de ses sous-tâches), à l'agent avec lequel va avoir lieu la communication et au contexte courant de la tâche. Nous proposons un moyen de définir ces schémas de communication et nous proposons un schéma basé sur notre traduction de la notion de croyance mutuelle.

Cependant, nous restreignons notre études aux communications concernant le déroulement d'une tâche : sa mise en route, sa suspension, sa reprise, son arrêt.

Notre système permet la gestion d'un ensemble de tâches, défini par un ensemble partiellement ordonné de sous-tâches auxquelles nous rattachons des schémas de communication adaptés.

3.2 Architecture

3.2.1 L'architecture LAAS

L'architecture LAAS [ACF+98] pour systèmes autonomes, présentée figure 1, a été développée de manière incrémentale durant un grand nombre d'années et est mise en oeuvre sur l'ensemble des robots mobiles du LAAS. Il faut noter que les aspects généricité et programmabilité de cette architecture permettent une mise en oeuvre rapide et une bonne intégration des systèmes utilisés (GenoM ([FHC97], [FHM05]), OpenPRS ([ICAR96],[Ing05]),...). Cette suite logiciel comprend trois niveaux :

- Le niveau décisionnel : Ce plus haut niveau intègre les capacités délibératives par exemple : produire des plans de tâches, reconnaître des situations, détecter des fautes, etc. Dans notre cas, il comprend :
  - un exécutif procédural OpenPRS qui est connecté au niveau inférieur auquel il envoie des requêtes qui vont lancer des actions (capteurs/actionneurs) ou démarrer des traitements. Il est responsable de la supervision des actions tout en étant réactif aux événements provenant du niveau inférieur et aux commandes de l'opérateur. Cet exécutif a un temps de réaction garanti,
  - un planificateur et/ou une bibliothèque de plans,
  - une base de faits.
- Le niveau fonctionnel : Le plus bas niveau comprend toutes les actions et fonctions de perception de base de l'agent. Ces boucles de contrôle et traitements de données sont encapsulées dans des modules contrôlables (développés avec GenoM). Chaque module fournit des services et traitements accessibles par des requêtes envoyées par le niveau supérieur ou un autre module. Le module envoie en retour un bilan lorsqu'il se termine correctement ou est interrompu. Ces modules sont complètement contrôlés par le niveau supérieur et leurs contraintes temporelles dépendent du type de traitement qu'ils ont à gérer (servo-contrôle, algorithmes de localisation, etc.).

– Le niveau de contrôle des requêtes (défini dans l'architecture mais non utilisé au sein de nos instanciations sur Jido et Rackham) : Situé entre les deux niveaux précédents, le R2C « Requests and Replies Checker » [IP02] vérifie les requêtes envoyées aux modules fonctionnels (par l'exécutif procédural ou entre modules) et l'utilisation des ressources.

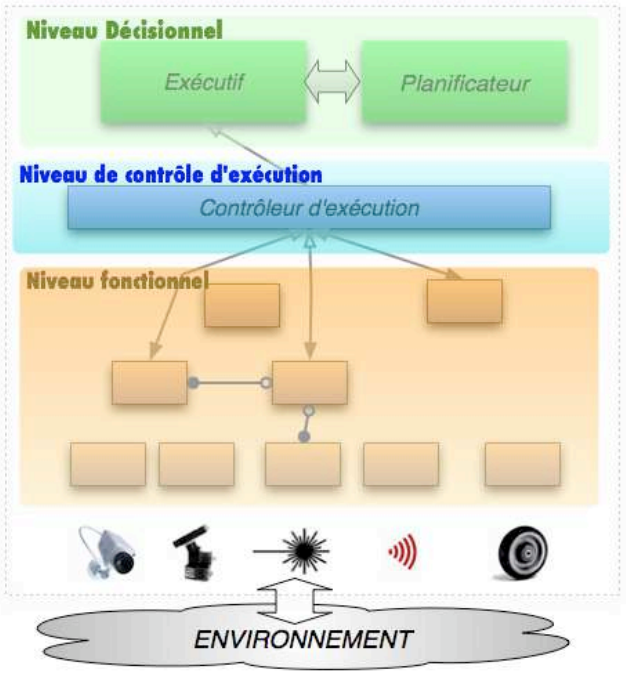


FIG. 1 – L'architecture LAAS

3.2.2 Une architecture adaptée à l'interaction

Dans notre contexte l'homme et le robot partagent un espace commun et échangent des informations à travers différentes modalités.

L'interaction peut se produire à la demande de l'homme ou à l'initiative du robot si la situation lui semble adéquate. Dans les deux cas, le robot aura une tâche à accomplir.

L'architecture définie devra couvrir l'établissement d'un but commun, l'affinage des tâches qui serviront à l'accomplir mais aussi l'établissement et le maintien de la communication entre l'homme et le robot pour réaliser ce but.

Pour réaliser cela, nous avons défini un ensemble de composants nécessaires à la supervision et à la réalisation de tâches collaboratives ([CMAC05, CA05, ACM+06a]). La figure 2 montre l'ensemble de ces composants qui s'intègrent au sein de la couche délibérative de l'architecture LAAS. Ces composants sont :

l'Agenda qui va s'assurer de la cohérence de l'ensemble des tâches en cours et la gérer,

la base de faits qui va comprendre deux éléments :

– l'état du monde modélisé par le robot qui va contenir des données a priori (carte de l'environnement, objets, meubles,...) et des données contextuelles (position et propriétaire des objets, état actuel de l'environnement,...)
– la base de données des agents (nommé IAA pour InterAction Agent) avec lesquels le robot pourra potentiellement interagir, à laquelle est associée l'IAA manager qui va

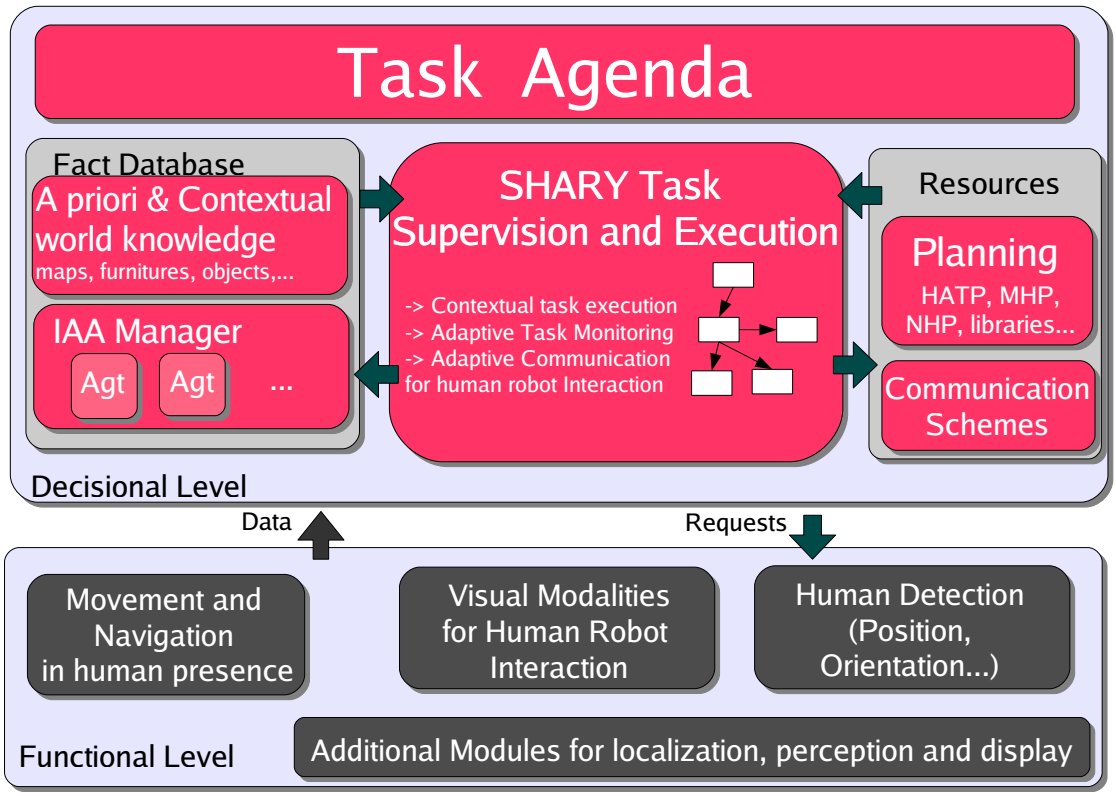


FIG. 2 – Une architecture adaptée à l'interaction ([CA05, ACM+06a]) : l'**Agenda** contient la vision globale des tâches en cours de réalisation ou à accomplir, la **base de faits** comprend d'une part les informations sur le monde et les informations sur les agents (InterActionAgent) avec lesquels le robot va interagir, le système d'exécution et de supervision **Shary** va assurer la prise en charge de l'exécution des tâches : leurs affinements et la communication pouvant intervenir lors de leurs déroulements, à l'aide des **ressources** : planificateurs et schémas de communication.

gérer la mise à jour de la base. A l'intérieur de cette base, le robot représentera un agent particulier.

Les données arrivant dans cette base de faits proviendront des informations données par le niveau fonctionnel qui auront été interprétées.

les ressources disponibles :

- d'une part en terme de planification [ACM+06b], e.g. HATP (*Human Aware Task Planner*, [MCA07, MCAR07]), NHP et MHP (*Navigation and Manipulation in Human Presence*, [SAS+05, SCMU+06, SUAS06]) qui vont calculer des plans acceptables pour l'homme ou même des bibliothèques de plans déjà définis,
- et d'autre part les schémas de communication disponibles pour gérer l'interaction au sein d'une tâche.

le système de supervision et d'exécution Shary (*Supervision for Human Adapted Robot Y(I)nteraction*) qui va gérer l'ensemble des tâches en cours, leurs exécutions et les communications associées,

Nous allons maintenant revenir sur chacun de ces composants.

Agenda

On peut donner plusieurs tâches au robot, impliquant potentiellement plusieurs personnes. A chaque instant ces tâches peuvent être en cours d'exécution, en attente d'exécution ou suspendus. Pour gérer cet ensemble de tâches, leurs compatibilités, les instants où il est nécessaire de les mettre en oeuvre (ou de les arrêter/suspendre/reprendre), bref pour décider ce que le robot doit faire et quand il doit le faire, nous avons défini un composant logiciel ayant un vue d'ensemble de tout ces éléments : c'est l'**Agenda**.

Basé sur les informations présentes dans la base de données concernant l'état de chacune des tâches qu'elles soient en cours ou non et concernant l'état de chacun des agents, l'**Agenda** va déterminer quelles tâches il est nécessaire d'activer, de suspendre, de reprendre ou d'abandonner pour assurer la cohérence de l'ensemble des tâches à réaliser.

L'**Agenda** va :

- être capable de déterminer si une tâche est réalisable ou non par le robot à un moment donné ou même plus tard, c'est donc à ce composant que le système d'exécution va s'adresser quand une prise de décision doit être effectuée,
- être à l'origine des prises d'initiatives du robot : l'**Agenda** aura pour mission d'exhiber de nouvelles tâches rendues pertinentes par le contexte d'exécution et les tâches déjà présentes.

Base de faits

La base de faits que nous définissons va contenir d'une part les informations données a priori et/ou collectées par le robot sur son environnement et décrivant l'état du monde et son évolution. Par exemple, le robot a accés aux données géométriques et topologiques de son environnement (cartes, segments,...) et des objets qui s'y trouvent.

D'autre part, nous décrivons les informations concernant les agents avec lesquels le robot peut interagir. Chaque homme rencontré par le robot est représenté par un agent (nommé IAA pour InterAction Agent). Une base d'agents connu à priori est disponible. Lorsque le robot rencontre et/ou interagit avec un agent, son état est mis à jour. Il faut bien noter que cette mise à jour est effectuée par le robot lui-même et non par un système extérieur, ainsi les données disponibles pour le robot concernant les autres agents sont filtrés par la perception du robot. L'agent représenté n'est en aucun cas rattaché à l'agent lui-même mais représente la perception

qu'à le robot de cet agent (potentiellement à travers des signaux plus ou moins explicites envoyés par lui).

Les informations stockées vont porter sur des données telles que la position géométrique et/ou topologique (si elles sont connues), les éventuels objets possédés,... ; mais également sur les capacités, préférences, besoins de l'agent en terme d'actions, de perception et d'interaction. Enfin l'état de chacune des tâches dans lequel l'agent est impliqué ou a été impliqué est rendu disponible. Ces informations seront utilisées par le superviseur et par les planificateurs de tâches et de mouvements pour produire des comportements adaptés.

Le robot est lui même représenté dans cette base sous la forme d'un agent pour faciliter la prise en compte des données le concernant.

Ressources

Les ressources comprennent les systèmes de planification disponibles qui seront appelées à chaque niveau de la tâche et les schémas de communication qui auront été définies.

Au sein d'une architecture adaptée à l'interaction, nous considérons qu'un planificateur doit non seulement fournir des plans rapidement mais qu'il doit également s'assurer que le plan retenu est socialement acceptable. Nous avons défini que pour produire de tels plans il faut que le planificateur prennent en compte les aptitudes et les préférences des partenaires humains du robot mais aussi certaines règles sociales afin que le(s) plan(s) retenu(s) ne donnent pas de sensations d'inconfort aux partenaires humains du robot. De plus, le planificateur devra être capable de considérer un certain nombre de contraintes que les partenaires humains du robot pourraient imposer sur la facon de procéder pour réaliser les buts courants.

HATP (*Human Aware Task Planner*), MHP (*Manipulation in Human Presence*) et NHP (*Navigation in Human Presence*) sont des planificateurs spécifiquement conçus pour prendre en compte les considérations évoquées ci-dessus.

Par exemple, HATP (pour Human Aware Task Planner) est un planificateur basé sur le formalisme HTN (Hierarchical Task Network). Afin de ne retenir parmi les plans possibles que les plans qui sont les plus "socialement" acceptables, HATP évalue chaque plan selon un critère social prenant en compte les aptitudes et les préférences des différents agents mais aussi un ensemble de règles sociales prédéfinies. Pour chaque plan réalisant les buts courants, HATP lui attribue un score calculé à partir de ce critère, plus ce score est faible plus le plan est de meilleur qualité. L'objectif de HATP devient alors de trouver le plan réalisant les buts courants et ayant le score le plus faible possible. De plus, HATP est capable de prendre en considération les potentielles contraintes imposées par les partenaires humains du robot sur la manière de procéder. Pour cela, HATP prend en entrée une structure partielle de plan et la complète afin d'aboutir au plan le plus socialement acceptable et vérifiant les "souhaits" des partenaires humains du robot.

Les ressources contiennent également la définition des schémas de communication dont nous parlerons plus en détail au prochain chapitre. Ces schémas de communication représentent les plans de communication qui serviront à gérer le déroulement de la communication et notamment la gestion des engagements dans la tâche et leur maintien.

Supervision et Exécution : Shary

L'**Agenda** va nous fournir les tâches à exécuter/suspendre/arrêter/reprendre, le contexte d'éxécution est défini au sein de la base de faits, les plans de communication et d'actions sont rendus disponibles, il nous reste à rendre possible l'exécution/suspension/arrêt/reprise des tâches. Shary est le système de supervision et d'exécution de tâches adapté à l'interaction homme-robot

que nous avons développé. Shary permet de gérer des tâches individuelles (où seul le robot est impliqué) et des tâches jointes (où le robot et un autre agent sont impliqués). Il nous a paru pertinent d'exhiber pour chacune des tâches, son exécution proprement dite et toute la communication qui peut intervenir autour de cette exécution, ce que nous avons construit à partir de schémas de communication. Par conséquent, Shary se compose d'une mécanique d'exécution de tâches gérant d'une part la communication nécessaire au sein d'une tâche et d'autre part l'affinement de la tâche en sous-tâches selon un schéma de type HTN. Pour cela chaque tâche est définie par un plan de communication et un plan de réalisation auxquels sont associés des moniteurs de suivi. Avec ce système, nous disposons d'un système de supervision et d'exécution de tâches hiérarchiques avec prise en charge de la communication à chacun des niveaux des tâches.

3.3 Conclusion

Nous avons défini une architecture générale pour la prise en compte de l'interaction au sein d'une architecture robotique existante. Nous avons défini un composant permettant d'avoir une vision générale des tâches à accomplir : l'**Agenda**, une base de faits et un système de mise à jour associé permettant un accès au contexte d'exécution de ces tâches, un système de supervision **Shary** assurant la prise en charge de l'exécution de ces tâches, leur affinement et la communication pouvant intervenir lors de leur déroulement, et enfin les ressources en terme de plan d'actions et de communication utilisées par le système d'exécution.

Après la définition de cette architecture, nous avons porté notre travail sur le système de supervision et d'exécution Shary. Dans le prochain chapitre, nous allons détailler ce système. Ensuite, nous analyserons une manière de modéliser la communication nécessaire autour de l'exécution d'une tâche jointe.

Chapitre 4

Shary

L'objectif du système de supervision Shary est de permettre l'exécution de tâches par le robot dans un contexte d'interaction avec l'homme. Le système se compose de deux éléments : d'une part une base de données regroupant les informations collectées par le robot ainsi que les informations qui lui sont données à priori, et d'autre part une mécanique d'exécution de tâches. Cette mécanique d'exécution gère d'une part la communication nécessaire au sein de la tâche et d'autre part l'affinement de la tâche en sous-tâches selon un schéma de type HTN mais avec une séparation entre le contexte de réalisation de la tâche et le contexte de communication qui l'entoure défini à l'aide de schémas de communication.

4.1 Description Générale

4.1.1 Tâches

Nous adoptons les notations trouvées dans la littérature pour définir les différents types de tâches que nous allons considérer. Tout d'abord on distingue les *tâches individuelles* et les *tâches jointes*. Les tâches individuelles sont accomplies par des individus et les tâches jointes par des ensembles d'individus. En ce qui concerne les tâches individuelles, on distingue : les *tâches autonomes* et les *tâches participatives*. Les tâches participatives sont des tâches individuelles réalisées au sein d'une tâche jointe. Une tâche jointe est constituée de tâches participatives. Ainsi un mouvement du bras réalisé pour déposer un objet dans une étagère sera considéré comme une tâche autonome alors qu'un autre réalisé pour donner un objet à l'homme constituera une tâche participative. De même, si on définit la tâche *"AllerA(PosFinal)"* comme la tâche permettant au robot de se rendre de sa position actuelle à la position finale "PosFinal", cette tâche réalisée dans la cadre d'une tâche de suivi du robot par l'homme ou de l'homme par le robot est une tâche individuelle participative alors que la même tâche réalisée par le robot seul est une tâche individuelle autonome. Nous considérerons les tâches autonomes comme *privées* et les tâches jointes et participatives comme *publiques*, dans le sens où elles auront à être partagées i.e. rendues publiques.

En définitive, nous définissons trois types de coopération pour les tâches : individuelle autonome, individuelle participative, jointe.

Nous adoptons une décomposition hiérarchique des tâches, cette hiérarchie peut être représentée sous la forme d'un arbre de tâches dont les feuilles sont les tâches atomiques et les branches les tâches abstraites qui nécessitent un découpage en sous-tâches. La figure 3 représente une manière de décomposer la tâche *"GiveObject(ObjectName,PersonName)"* dans un certain contexte, cette décomposition ne représente qu'une possibilité de réalisation de la tâche correspondant au

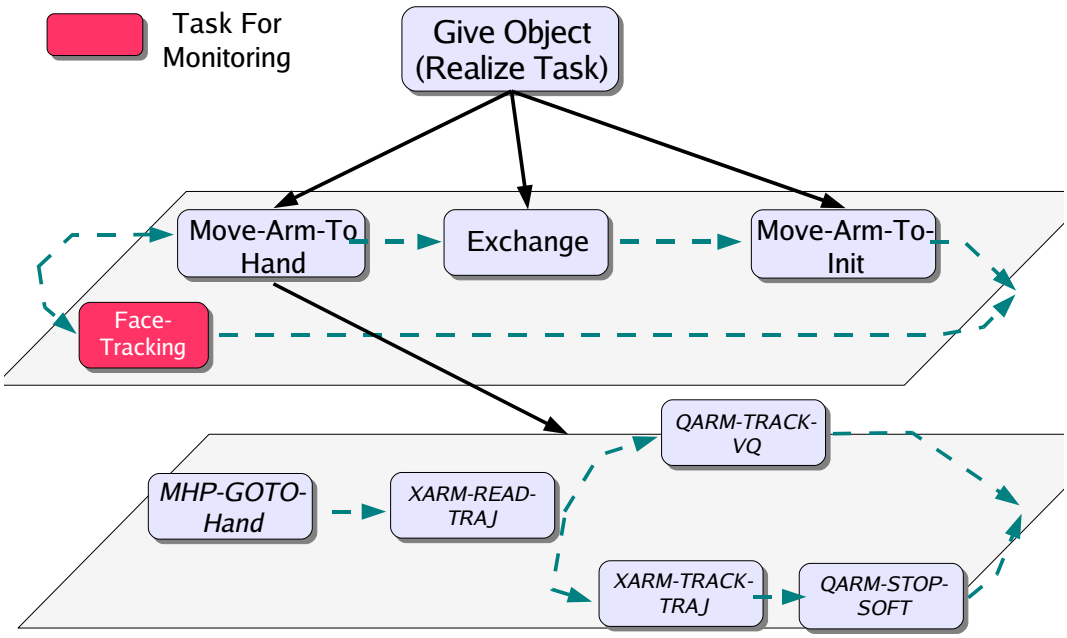


FIG. 3 – une décomposition hiérarchique de la réalisation de la tâche *"GiveObject(ObjectName,PersonName)"*

contexte actuel de la tâche. Un autre contexte pourra conduire à une autre décomposition, par exemple si l'homme ne peut ou ne veut pas prendre l'objet, le robot pourra le poser près de lui.

Le choix d'une décomposition hiérarchique a été motivé par deux raisons. D'une part, ce découpage permet une réaction adaptée à chacun des niveaux. Ainsi, un problème avec *"Move-Arm-To-Hand"* tentera d'être résolu à ce niveau, par exemple en trouvant un autre moyen de réaliser le mouvement du bras, avant d'être repercuté à un niveau supérieur si le problème persiste. Cela permet d'être plus réactif et facilite la réutilisation de tâches déjà définies. D'autre part, la volonté de permettre au robot d'extraire des informations compréhensibles par l'homme à partir de son plan. En maintenant l'entière décomposition de son plan, le robot est capable de communiquer à/avec l'homme son contexte courant d'exécution.

La table 4.1 décrit notre définition d'une tâche. Concernant les états possibles d'une tâche, nous reprenons la notation de la théorie de l'intention jointe : *unachieved* (*en cours*), *achieved* (*terminée/réalisée*), *impossible* (*impossible*), *irrelevant* (*non-pertinente*), en y ajoutant la possibilité *stopped* (*arrêtée*). Lorsque la tâche est en cours d'exécution, son état est à *unachieved* (*en cours*). Son état passe à *achieved* (*terminée*) (resp. *impossible* (*impossible*), *irrelevant* (*non-pertinente*), *stopped* (*arrêtée*)) si elle est terminée (resp. impossible, non-pertinente, arrêtée)), une tâche est terminée (resp. impossible, non-pertinente, arrêtée) non-seulement si la réalisation de la tâche est terminée (resp. impossible, non-pertinente, arrêtée), mais aussi que toute la communication nécessaire autour de cette tâche est terminée (i.e. le fait que la tâche soit terminée (resp. impossible, non-pertinente, arrêtée)) a été communiqué si nécessaire au partenaire de la tâche.). Ce choix peut être rapproché de la notion de but persistant joint (cf section 1.1.2.0 page 27) où le but ne peut être abandonné tant que tous les participants ne sont pas alertés de l'état

TAB. 4.1 – Définition d'une tâche instanciée

attribut	**valeur**
id	un identifiant unique est alloué à chaque tâche
task-name	nom de la tâche
task-coop	type coopératif de la tâche *possibilités :* *individual autonomous,* *individual participative,* *joint*
task-state	état de la tâche *possibilités :* *unachieved, achieved, irrelevant,* *impossible, stopped*
task-com	schéma de communication

de la tâche.

Le paramètre task-com indique quel schéma de communication doit être utilisé par la tâche, nous reviendrons en détail sur la notion de schéma de communication dans la section qui lui est dédiée page 92. Pour le moment nous préciserons seulement que le choix du schéma de communication se fait lors de l'instanciation de la tâche, il dépend du type de tâche, du type de coopération, du partenaire et du contexte courant d'exécution.

4.1.2 Actes de communication

Au sein d'une tâche, nous distinguons des actes de communication, ces actes et leurs enchainements représentent la communication qui peut exister au sein d'une tâche autour et lors de sa réalisation.

Un acte de communication représente l'échange d'information entre deux agents qui peut être réalisé par l'intermédiaire d'un dialogue, aussi bien que par un mouvement ou une combinaison de plusieurs modalités. Dans notre cas, les actes de communication pourront être émis par le robot ou par les agents avec qui il est amené à interagir (dans ce cas le robot doit se doter des moyens de percevoir et d'interpréter ces actes). Cet échange permet de partager les connaissances concernant la tâche à réaliser (ou en cours de réalisation) et de se mettre d'accord sur les conditions d'exécution de la tâche et sur son évolution. Par exemple, lorsque le robot doit réaliser une tâche avec l'homme, le robot peut être amené à demander son accord à l'homme, cependant en fonction des connaissances a priori disponibles sur l'homme, ses préférences et le contexte courant de la tâche, cette communication sera explicite ou implicite.

Un acte de communication (cf table 4.2) est défini par un nom (*name*) qui caractérise l'objet de la communication, une étape (*step*) qui caractérise son évolution au cours de l'interaction ainsi que par son état si l'acte est initié par le robot (nous verrons plus tard que cette assymétrie provient d'un décalage dans la réception de l'information, e.g. le robot sait qu'il a terminé une tâche avant de le communiquer à l'homme alors que lorsque le robot est mis au courant que l'homme a terminé il reçoit directement l'acte de communication).

Si nous définissons par exemple l'acte de communication **ask-task**, les étapes vont correspondre d'abord à la réalisation de l'acte de communication (**ask-task** *act*) qui pourra être suivi

de la réception d'un accusé de réception (**ask-task** *acknowledge*) de la part de l'interlocuteur qui pourra amener l'auteur de l'acte (**ask-task** *act*) à confirmer l'acte (**ask-task** *confirm*). Nous voyons pourquoi nous avons du définir un acte non-seulement par un nom mais également par une étape pour qualifier son évolution.

Dans notre optique de communication relative au contrôle de l'exécution d'une tâche, nous avons défini les actes suivants :

- **ask-task** : proposer une tâche. Cet acte définit le début d'une interaction où l'homme demande au robot ou le robot à l'homme si il souhaite participer à une tâche. Cet acte va correspondre à la phase d'établissement des engagements (*establish commitment phase*) rencontré précédemment section 1.1.3.0 page 28.
- **propose-plan** : proposer un plan pour réaliser la tâche. Cet acte définit qu'il est nécessaire de se mettre d'accord sur un plan, i.e. l'engagement n'est plus alors à obtenir sur la tâche mais sur le plan pour la réaliser.
- **modify-plan** : proposer une modification du plan courant. Cet acte est identique au précédent sauf qu'ici il y a remise en cause d'un plan déjà défini.
- **give-up** : abandonner la tâche. Cet acte va correspondre à l'annonce de l'abandon de la tâche car l'agent est dans l'impossibilité d'accomplir la tâche.
- **cancel** : annulation de la tâche. Cet acte va correspondre à l'annonce d'un abandon volontaire de la tâche devenu non-pertinente.
- **end_task** : terminaison de la tâche. Cet acte va correspondre à l'annonce de la terminaison, de la fin de la réalisation de la tâche par l'agent concerné.
- **realize-task** : réalisation de la tâche (et si besoin annonce du début de cette réalisation).

Bien entendu ce dernier acte : **realize-task** peut ne pas être considéré en soit comme un acte de communication, mais comme son nom l'indique comme un acte de réalisation de la tâche. Cependant, le fait de commencer à réaliser une tâche peut exhiber de la communication dans le sens où cela peut montrer un engagement dans la tâche. Ensuite, considérer la réalisation même de la tâche au même titre que les autres actes de communication permet d'homogénéiser la gestion des actes. Le plan de la tâche dont il est question lors des **propose_plan** et **modify_plan** correspond au plan de réalisation de la tâche (i.e. le plan de **realise_task** *act*).

Nous reviendrons plus tard sur les étapes définis pour chacun des actes, ceux donnés en exemple dans le tableau correspondent à un schéma particulier que nous avons défini qui permet d'intégrer une notion de degré de compréhension au sein des actes, i.e. différence entre *clear* (*clair*) et *not clear* (*non-clair*). Ainsi lorsque le robot tend un objet à l'homme, si l'homme s'en va cela peut être interprété comme un acte clair de renoncement, alors que le fait que l'homme reste devant le robot sans prendre l'objet va nécessiter une clarification, mais nous reviendrons sur ces notions par la suite. Nous verrons également pourquoi les étapes diffèrent selon que l'initiateur de la communication soit l'homme ou le robot.

Les actes et les étapes présentés ici ne sont que des exemples, toute liberté est laissée au programmeur de définir ceux qu'il souhaite.

Nous appelons un enchaînement d'actes un schéma de communication, un acte étant relié à une tâche nous parlerons d'Act_X_Task quand il sera question d'un acte de communication donné au sein d'une tâche donnée.

TAB. 4.2 – Définition d'un acte de communication : à gauche dans le cas où le robot est l'auteur de l'acte, à droite dans le cas où l'homme est l'auteur de l'acte

attribut	**valeur**	
act-sender	la personne réalisant l'acte	
	robot	*human*
act-addressee	le destinataire de l'acte	
	human	*robot*
act-name	nom de l'acte	
	exemples	
	ask_task, propose_plan, modify_plan, realize_task, end_task, suspend_task, resume_task,	
act-step	étape de l'acte	
	exemples	
	act, acknowledge, refute, answer confirm	*act clear, act not clear, confirm, refute, answer clear, answer not clear, acknowledge*
act-state	état de l'acte	
	possibilités	
	unachieved, achieved, irrelevant, impossible stopped	none

4.1.3 Schéma de communication

Un schéma de communication définit un enchaînement d'actes de communication selon deux fonctions :

$$\begin{aligned} \mathcal{F}: \quad Act_{current} &\rightarrow Act_{expected1} \times \ldots \times Act_{expected} \\ \mathcal{G}: \ Act_{current} \times Act_{incoming} &\rightarrow Act_{next} \end{aligned}$$

La fonction F définit la liste des actes attendus en retour de l'acte en cours de réalisation. La fonction G permet de récupérer le prochain acte à réaliser en fonction de l'acte en cours de réalisation et de l'acte perçu.

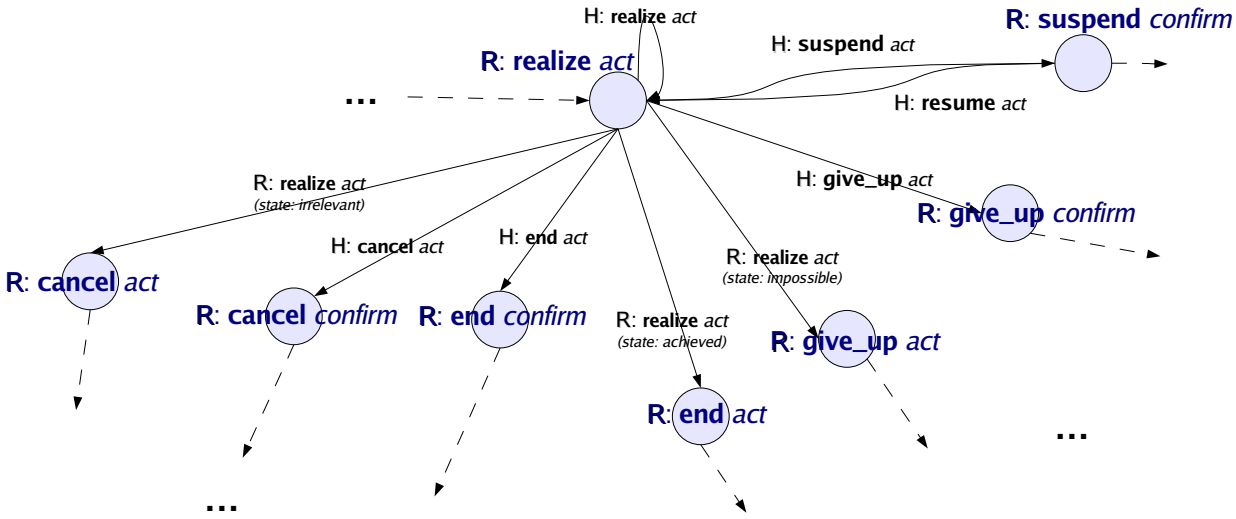


FIG. 4 – Une partie d'un schéma de communication : R ou H caractérise l'auteur de l'acte de communication.

La figure 4 représente une partie d'un schéma de communication. Si nous utilisons un tel schéma lors de la phase de réalisation $R:$ *realize act* de notre tâche "GiveObject(ObjectName,PersonName)", $\mathcal{F}$ sera défini de la façon suivante :

$$\begin{aligned} \mathcal{F}: \quad \mathbf{R}: \textbf{realize } act \quad \rightarrow \quad & \mathbf{R}: \textbf{realize } act\ (state: irrelevant) \times \\ & \mathbf{R}: \textbf{realize } act\ (state: impossible) \times \\ & \mathbf{R}: \textbf{realize } act\ (state: achieved) \times \\ & \mathbf{H}: \textbf{suspend } act \times \\ & \mathbf{H}: \textbf{giveup } act\ \times \\ & \mathbf{H}: \textbf{end } act \times \\ & \mathbf{H}: \textbf{cancel } act \end{aligned}$$

Nous voyons que les actes attendus qu'il faudra monitorer sont de deux types : tout d'abord les actes correspondants à un changement d'état de l'acte réalisé par le robot, d'autre part les actes attendus de la part de l'homme.

Si l'homme s'en va, le robot pourra interpréter cela comme un abandon, nous obtenons alors la fonction $\mathcal{G}$:

$$\mathcal{G} : \quad \mathbf{R} : \textbf{realize } act \times \mathbf{H} : \textbf{giveup } act \rightarrow \mathbf{R} : \textbf{giveup } confirm$$

Il est nécessaire de définir pour chaque schéma de communication, un état (ou acte) d'entrée et un ou plusieurs états de sortie. Les états de sortie sont prédéfinis et vont refléter l'état de la tâche : *impossible, achieved, irrelevant, stopped*. Ils permettent de faire remonter l'information au niveau supérieur.

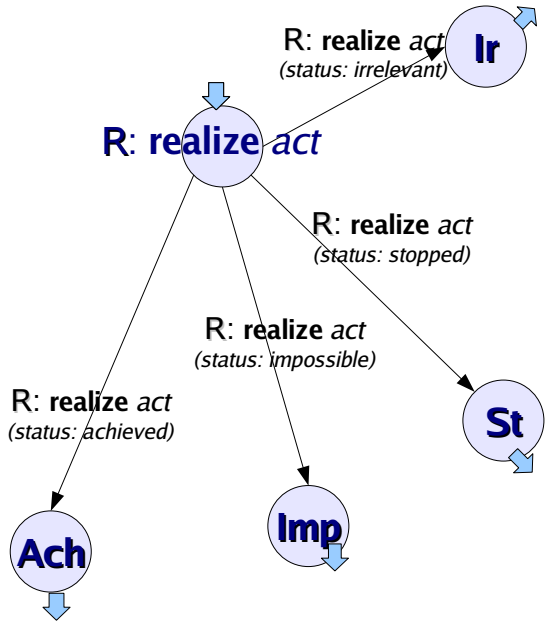


FIG. 5 – Un schéma minimum de communication.

Ainsi, un schéma minimum de communication est celui décrit figure 5. Dans ce schéma, l'état d'entrée correspond à un acte de réalisation de la tâche. En fonction du résultat obtenu : terminé (*achieved*), impossible (*impossible*), non-pertinent (*irrelevant*), stoppé (*stopped*), le schéma dirige vers les actes de sortie correspondant. Dans ce cas, le programmeur aura "juste" à coder l'acte **realize** *act* et les moniteurs correspondants à terminé (*achieved*), impossible (*impossible*), non-pertinent (*irrelevant*), stoppé (*stopped*), les actes de sortie sont déjà précodés et répercutent cette information au sein de la hiérarchie de tâches. Ainsi, pour une tâche qui se termine en *achieved* on passera aux tâches suivantes si il en existe, si c'est un autre résultat, l'information sera envoyée à la tâche immédiatement supérieure (i.e. l'état de l'acte en cours d'exécution prendra la valeur impossible (*impossible*) ou non-pertinent (*irrelevant*), le cas stoppé (*stopped*) est un peu différent car il faut s'assurer que toutes les tâches soient stoppées avant de déclarer l'acte stoppé).

En dehors de ces contraintes, le programmeur est libre de décrire les schémas de communication qu'il souhaite. Ceci peut par exemple permettre de décrire des schémas plus ou moins

collaboratifs. Le schéma utilisé pour une tâche étant instancié lors de l'exécution de la tâche, cette instanciation peut être dépendante de l'agent avec lequel va s'effectuer l'interaction, du contexte du courant (e.g. la position des agents), en bref de toute information qui sera rendue disponible.

Les schémas de communication sont définis de manière indépendante aux tâches. Cette indépendance permet la réutilisation des schémas de communication définis et permet de définir/choisir différents schémas de communication et donc différentes manières de réaliser la communication pour une même tâche.

4.1.4 Act_X_Task

Un acte de communication et le schéma auquel il appartient ne contient aucune information a priori sur la tâche.

Une tâche ne s'exécute qu'à travers l'exécution des actes de communication qui sont associés au schéma de communication qui lui correspond.

Un Act_X_Task représente l'instanciation d'un acte de communication au sein d'une tâche.

La définition d'un Act_X_Task est donnée table 4.3, elle rejoint la définition donnée par un acte de communication auquelle se rajoute un identifiant unique, la tâche associée et le niveau de l'ActXTask, i.e. nous distinguons les Act_X_Task atomiques qui feront directement appel au niveau fonctionnel et aux modules et les ActXTask abstraits qui seront décomposés en une liste partiellement ordonnée de sous-tâches. Cette liste représente la "recette" ou script pour l'exécution de la tâche.

4.1.5 Recette

Une recette représente la liste partiellement ordonnée des sous-tâches associés à un Act_X_Task. Cette liste de sous-tâches contient l'ensemble des tâches permettant d'exécuter l'Act_X_Task associé à l'ensemble des tâches nécessaires au monitoring de l'exécution, elle correspond à un niveau de décomposition et à un contexte donné d'exécution de l'Act_X_Task

Le plan de la tâche dont il est question lors des **propose plan** et **modify plan** correspond au plan de réalisation de la tâche (i.e. le plan de **realise task** *act*). La recette pour l'Act_X_Task **realise task** *act* correspond au plan de la tâche pour l'agent considéré.

Ainsi, la décomposition en sous-tâches défini figure 3 n'est en fait pas celui d'une tâche mais celui de l'Act_X_Task : R : **realize task***act* de la tâche "GiveObject(ObjectName, PersonName)", i.e. le plan de la tâche "GiveObject(ObjectName, PersonName)" pour ce qui concerne le robot. La recette associée se décompose en 4 sous-tâches : "Move-Arm-To-Hand", "Exchange" (qui est une tâche jointe) et "Move-Arm-To-Rest" qui sont nécessaires à l'exécution de l'Act_X_Task, alors que la tâche "Face-Tracking" est une tâche de monitoring. Il est à noter que nous rendons ici cette distinction visible mais qu'elle n'apparaît pas dans les définitions des recettes.

Nous partons du principe que ces deux éléments nous sont fournis soit par le système de planification soit qu'ils sont décrits au sein de la bibliothèque de recettes. Les tâches de monitoring des actes de communication de l'homme sont spécifiques à notre contexte d'interaction homme-robot, elles permettent en effet d'intégrer l'homme dans la boucle et d'avoir une vision de ses intentions. Cela peut être rapproché de la notion de "focus of attention" que l'on retrouve chez Grosz [GS86] (voir aussi la description de SharedPlans dans la bibliographie page 37). Considérer que ces deux éléments sont déjà définis n'est pas anodin, l'exécution et le monitoring ne formant pas nécessairement un ensemble disjoint. En effet, ils peuvent être amenés à partager des ressources communes. Par exemple la sous-tâche "Face-tracking" va utiliser des images et

TAB. 4.3 – Définition d'un Act_X_Task

attribut	**valeur**	
id	un identifiant unique est alloué à chaque Act_X_Task	
task-name	tâche à laquelle est associée l'Act_X_Task	
act-sender	la personne réalisant l'acte	
	robot	*human*
act-addressee	le destinataire de l'acte	
	human	*robot*
act-name	nom de l'acte	
	exemples *ask_task, propose_plan, modify_plan,* *realize_task, end_task, suspend_task,* *resume_task,*	
act-step	étape de l'acte	
	exemples	
	act, acknowledge, *refute, answer* *confirm*	*act clear,* *act not clear,* *confirm, refute,* *answer clear,* *answer not clear,* *acknowledge*
act-state	état de l'acte	
	possibilités	
	unachieved, achieved, *irrelevant, impossible* *stopped*	none
act-position	position dans schéma	
	possibilités *enter, in, exit*	
ActXTask-level	niveau d'exécution	
	possibilités *atomic, abstract*	

donc les caméras qui seront aussi utilisées par "Move-to-hand" pour tracker la position de la main de l'homme. La gestion de ces dépendances est laissée aux méthodes de planification ou à l'écriture des bibliothèques de recettes, l'exécutif suppose que la décomposition obtenue est indemne de tout conflit.

La recette correspondant à un Act_X_Task est une liste partiellement ordonnée de sous-tâches (et pas d'actes ou d'Act_X_Task). C'est lors de l'exécution de chacune des tâches et de manière dynamique que sont instanciés les actes de communication nécessaires à l'exécution de ces sous-tâches.

La figure 6 montre un exemple de procédure OpenPrs codant une recette. On notera que la partie `context` de la procédure OpenPrs ainsi défini peut contenir toutes les informations qui vont permettre de définir les contraintes associées à cette recette. Si l'on reprend les contraintes définies dans SharedPlans, nous aurons les contraintes concernant la réalisation de l'Act_X_Task, e.g. cette recette de l'Act_X_Task ne sera valide que si telle et telle proposition est vérifiée, les contraintes provenant du "contexte intentionnel", e.g. si la tâche concernée est une sous-tâche de telle autre tâche la recette sera celle-ci ou celle-là, ou selon l'agent avec lequel la tâche va être exécuter si elle est jointe, etc. En fait ces contraintes peuvent être instanciées à partir de toute information présente dans la base de faits.

4.1.6 Fonctionnement

Le fonctionnement général du système au niveau d'une tâche (qui s'insère au sein d'une hiérarchie de tâches) est décrit figure 7. Lors de la création de la tâche, le schéma de communication associé à la tâche est instancié : *Adapted Scheme* (en fonction de la tâche, du contexte et des agents concernés). A ce schéma est associé un acte d'entrée dans la tâche qui sera le premier acte à exécuter. La recette correspondant à cet acte (plus précisément à cet Act_X_Task) est instanciée lors de l'exécution à l'aide de la bibliothèque de recettes : *Recipe*. Les moniteurs nécessaires au suivi de la communication d'une part et de la tâche d'autre part sont également instanciés sous la forme générique d'attente d'actes de communication : *Expected Acts*. L'acte est alors exécuté : *Act Execution*. Au déclenchement d'un moniteur : *Incoming Act* (i.e. à l'arrivée d'un acte de communication attendu), on arrête l'acte en cours d'exécution puis on instancie la réponse associée à l'acte reçu en fonction du schéma de communication : *Next Act*. Son exécution s'effectue alors de la même façon.

Nous allons maintenant étudier plus précisément comment fonctionne cette mécanique d'exécution et décrire les algorithmes associés.

4.2 Mécanique d'exécution

Cette mécanique d'exécution gère d'une part la communication nécessaire au sein de la tâche et d'autre part l'affinement de la tâche en sous-tâches ou plus précisément de chacun des Act_X_Task en sous-tâches.

Le coeur de la mécanique est constitué par l'algorithme Launch-ActXTask qui va démarrer l'exécution d'un Act_X_Task, il est décrit figure 8.

L'algorithme démarre en vérifiant que les tâches précédentes (au même niveau de la hiérarchie) se sont terminées correctement : "verifyPreviousAct". Cette fonction est bloquante tant que le ou les actes précédents sont en cours. Si ils se terminent correctement (i.e. via l'état de sortie Achieved), l'acte peut être lancé, sinon l'acte est abandonné.

Ensuite, si l'Act_X_Task est de niveau atomique, son exécution est lancée par la fonction "Execute", sinon nous allons rechercher la recette associée à l'Act_X_Task : "Get-ActXTask-

```
(defop |RECIPE REALIZE_TASK ACT T-MOVE-ARM-TO-HAND|
  :invocation (! (GET-RECIPE
                   (id ID)
               (act-name  $ACT-NAME)
          (act-step  $ACT-STEP)
          (task-name $TASK-NAME) $REPORT))
  :context(&
   (EQUAL $ACT-NAME  REALIZE-TASK)
   (EQUAL $ACT-STEP  ACT)
   (EQUAL $TASK-NAME T-MOVE-ARM-TO-HAND)
          (ACT-X-TASK (id $ID)
                      (task-name $TASK-NAME)
                      (act-sender ROBOT)
                      (act-addressee NONE)
                      (act-name $ACT-NAME)
                      (act-step $ACT-STEP)
          )
  )
  :body(
(! (= $PARENT-ID ID))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;MHP GOTO HAND    ;;;;;;;;;;;;;;;;;;
        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(! (GENERATE-NEXT-ID $T-MHP-GOTO-HAND-ID))
(=> (TASK (id $T-MHP-GOTO-HAND-ID) (task-name T-MHP-GOTO-HAND )
                                          (params                    )
                                          (parent $PARENT-ID         )
                                          (previous (. .))))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;; XARM READ TRAJ  ;;;;;;;;;;;;;;;;;;
        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(! (GENERATE-NEXT-ID $T-XARM-READ-TRAJ-ID))
(=> (TASK (id $T-XARM-READ-TRAJ-ID) (task-name T-XARM-READ-TRAJ)
                                           (params                    )
                                           (parent $PARENT-ID         )
                                           (previous (.$T-MHP-GOTO-HAND-ID .))))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;; QARM TRACK VQ   ;;;;;;;;;;;;;;;;;;
        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(! (GENERATE-NEXT-ID $T-QARM-TRACK-VQ-ID))

(=> (TASK (id $T-QARM-TRACK-VQ-ID) (task-name T-QARM-TRACK-VQ)
                                          (params                   )
                                          (parent $PARENT-ID        )
                                          (previous (.$T-XARM-READ-TRAJ-ID .))))
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;; XARM TRACK TRAJ ;;;;;;;;;;;;;;;;;;
        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(! (GENERATE-NEXT-ID $T-XARM-TRACK-TRAJ-ID))

(=> (TASK (id $T-XARM-TRACK-TRAJ-ID) (task-type T-XARM-TRACK-TRAJ )
                                           (params                      )
                                           (parent $PARENT-ID           )
                                           (previous (.$T-XARM-READ-TRAJ-ID .))))
        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;; QARM STOP SOFT  ;;;;;;;;;;;;;;;;;;
        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(! (GENERATE-NEXT-ID $T-QARM-STOP-SOFT-ID))

(=> (TASK (id $T-QARM-STOP-SOFT-ID) (task-type T-QARM-STOP-SOFT   )
                                           (params                      )
                                           (parent $PARENT-ID           )
                                           (previous (.$T-XARM-TRACK-TRAJ-ID .))))

(! (= $REPORT RECIPE-OK))
)
  )
```

FIG. 6 – Cette figure représente le codage de la recette de l'Act_X_Task **realize** *act* de la tâche MOVE-ARM-TO-HAND. Cela se traduit par la construction d'une liste de tâches qui vont correspondre à la décomposition que l'on retrouve figure 3. L'ordre partiel est géré par le fait que l'exécution d'une tâche est subordonnée à la terminaison de la tâche la précédent.

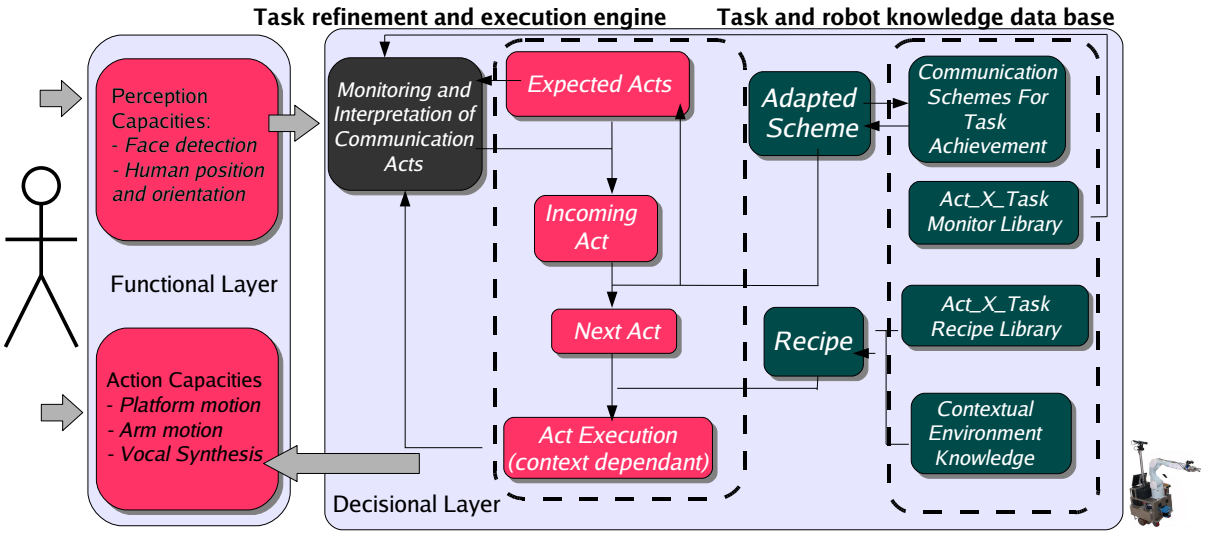


FIG. 7 – Description Générale du fonctionnement de Shary

Recipe" et nous lançons l'exécution des sous-tâches associées "Launch-ActXTask-Subtasks". Quelque soit le cas, les moniteurs correspondants sont enclenchés "Activate-ActXTask-Monitors", cela instancie la fonction **F** décrite au niveau des schémas de communication, et le système est mis en attente du déclenchement de l'un d'eux. Au déclenchement d'un moniteur (i.e. à l'arrivée d'un acte de communication), l'Act_X_Task en cours d'exécution est arrêté (ainsi que les sous-tâches associées).

On vérifie alors si l'état dans lequel on se trouve est un état de sortie. Si oui et qu'il s'agit de l'état de sortie *achieved*, on va rechercher si d'autres tâches sont définies à la suite dans la recette. Si il en existe, les tâches suivantes sont lancées "Launch-Task-Brothers". Si ce n'est pas un état de sortie, on va rechercher le prochain acte à exécuter dans le schéma de communication et on lance son exécution "Launch-ActXTask".

FIG. 8 – Description de l'algorithme de lancement d'un Act_X_Task, les algorithmes 1, 2, 3, 4, 5 présentent les autres algorithmes utilisés.

```
Invocation : Launch-ActXTask(ActXTask id) :
status = verifyPreviousAct(ActXTask id) ;
if status = OK then
    if (isAtomic(id.task − level)) then
        Execute(ActXTask id);
    else
        subtree = Get-ActXTask-Recipe(ActXTask id);
        Launch-ActXTask-Subtasks (TasksList subtree) ; /* vertical transition   */
    Monitor − Triggered = Activate-ActXTask-Monitors(ActXTask id);
    Stop-ActXTask-SubTasks (TasksList subtree);
    /* horizontal transition                                                    */
    if (isExit(id)) then
        if (id.act − name == ACHIEVED) then
            task − id = Get-Task (ActXTask id);
            tasks_list = Get-Task-Brothers(Task task − id);
            if (isNotEmpty(tasks_list)) then
                /* inter task transition                                        */
                Launch-Task-Brothers(TaskList tasks_list);
    else
        /* intra task transition                                                */
        next − act − id = Get-Next-Act(Monitor − Triggered);
        Launch-ActXTask(ActXTask next − act − id);
```

Algorithm 1: Launch-ActXTask Execution Algorithm

```
Invocation : Launch-ActXTask-Subtasks(TasksList subtree) :
tasks − list = Get-List-First-Tasks(TasksList subtree) ;
foreach task ∈ tasks − lists do
    act = Get-Task-First-Act(Task task − id);
    launchActXTask(ActXTask act);
```

Algorithm 2: Launch-AxtXTask-Subtasks Execution Algorithm

```
Invocation : Launch-Task-Brothers(TasksList tasks − list) :
foreach task − id ∈ tasks − list do
    act = Get-Task-First-Act(Task task − id);
    launchActXTask(ActXTask act);
```

Algorithm 3: Launch-AxtXTask-Brothers Execution Algorithm

```
Invocation : Stop-ActXTask-Subtasks(TasksList subtree) :
tasks − list = Get-Active-Tasks(TasksList subtree) ;
foreach task ∈ tasks − lists do
    act = Get-Active-Act(Task task − id);
    Stop-ActXTask(ActXTask act);
```

Algorithm 4: Stop-AxtXTask-Subtasks Execution Algorithm

Invocation : Stop-ActXTask(ActXTask *id*) :
if *(isAtomic(id.task − level))* **then**
  Stop(ActXTask *id*);
else
  subtree = Get-ActXTask-Recipe(ActXTask *id*);
  Stop-ActXTask-Subtasks (TasksList *subtree*) ; `/* vertical transition */`

Algorithm 5: Stop-ActXTask Execution Algorithm

4.3 Ecrire une tâche avec Shary

4.3.1 Scénario

Dans le scénario choisi Jido propose à une personne se trouvant devant lui de prendre l'objet qu'il tient.

4.3.2 Description

Pour pouvoir décrire cette tâche avec Shary, il faut :
- décrire au moins un schéma de communication,
- pour chaque acte présent dans le(s) schéma(s) de communication pouvant être associé à une tâche : écrire les recettes des Act_X_Task correspondants si l'Act_X_Task est de niveau abstrait et l'appel aux modules si il est de niveau atomique.

Schémas de communication

Pour cette tâche, nous avons défini deux schémas de communication que l'on retrouve figure 9. Le schéma de gauche va servir aux tâches jointes, il représente un schéma basique de "*turn-taking*" pour la prise en compte de la communication au sein de ce type de tâches. Le schéma de droite est celui que nous avons déjà rencontré et qui représente l'instantiation d'un but persistant individuel.

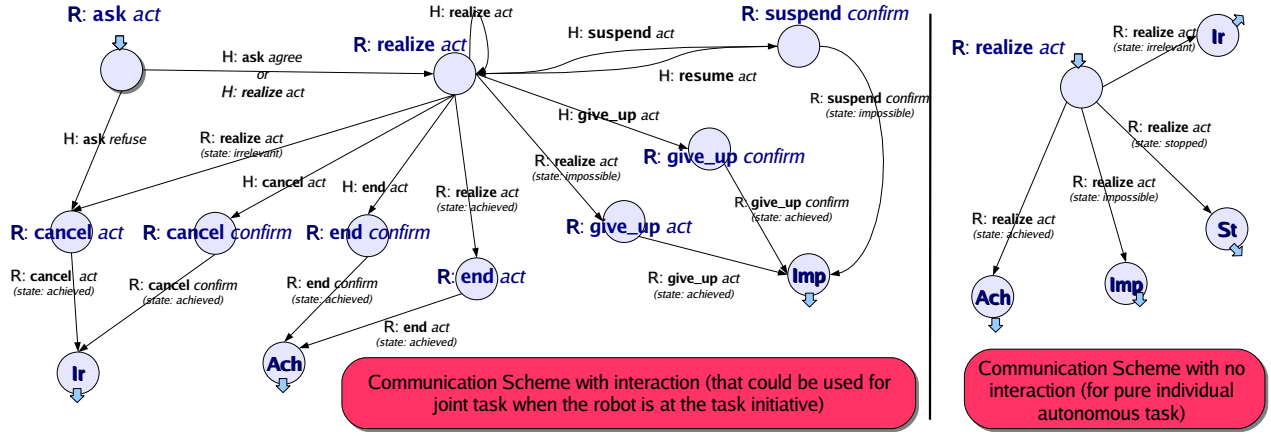


FIG. 9 – Les deux schémas de communication définis. Le schéma de gauche va servir aux tâches jointes et celui de droite aux tâches individuelles.

Tâches et schémas de communication

Le choix du schéma de communication associé à une tâche se fait dynamiquement en fonction du contexte courant de la tâche, de son type et des agents considérés. Si le programmeur souhaite rendre disponible un schéma de communication pour une tâche, il doit :

- concernant les actes de communication du robot :
  - si il s'agit d'un acte à exécuter : écrire pour chacun des actes définis un moyen de le réaliser, c'est à dire coder coder l'Act_X_Task formé. Par exemple pour la tâche jointe *Give* si l'on utilise le schéma de gauche de la figure 9 il faudra coder : (GIVE **R : ask** *act*), (GIVE **R : realize** *act*), (GIVE **R : suspend** *confirm*), (GIVE **R : cancel** *act*), (GIVE **R : cancel** *confirm*), (GIVE **R : end** *act*), (GIVE **R : end** *confirm*), (GIVE **R : give_up** *act*), (GIVE **R : give_up** *confirm*). Les Act_X_Task abstraits demanderont la définition d'une recette telle qu'on l'a expliqué section 4.1.5, si il s'agit d'une tâche atomique, il faudra coder son exécution.
  - si il s'agit d'un acte de monitoring de l'état de réalisation de l'acte : il faut coder la "traduction" du moniteur en terme de contexte. Ainsi, toujours pour la tâche jointe *Give* si l'on utilise le schéma de gauche de la figure 9 il faudra coder les moniteurs : (GIVE **R : realize_task** *act (state : irrelevant)*), (GIVE **R : realize_task** *act (state : achieved)*), (GIVE **R : realize_task** *act (state : impossible)*), (GIVE **R : suspend** *confirm (state : impossible)*), (GIVE **R : give_up** *confirm (state : achieved)*), (GIVE **R : give_up** *act (state : achieved)*), (GIVE **R : end** *confirm (state : achieved)*), (GIVE **R : end** *act (state : achieved)*), (GIVE **R : cancel** *confirm (state : achieved)*), (GIVE **R : cancel** *act (state : achieved)*). Traduire un moniteur consiste à déterminer sur l'arrivée de quel fait ou ensembles de faits dans la base de faits le moniteur sera réalisé. Par exemple, le fait que la personne soit en possession de l'objet peut être considéré comme la terminaison de la réalisation de la tâche *Give* par le robot et donc être associé à (GIVE **R : realize_task** *act (state : achieved)*) .
- concernant les actes de communication de l'homme, là encore il nous faut "traduire" les éléments de la base de faits correspondant. Là encore, toujours pour la tâche jointe *Give* si l'on utilise le schéma de gauche de la figure 9 il faudra coder les moniteurs : (GIVE **H : ask** *agree*), (GIVE **H : ask** *refuse*), (GIVE **H : realize_act** *act*), (GIVE **H : suspend** *act*), (GIVE **H : resume** *act*), (GIVE **H : give_up** *act*), (GIVE **H : end** *act*), (GIVE **H : cancel** *act*). Ainsi, le fait que l'on détecte que l'homme s'en va, peut être interprété comme un abandon de la tâche et être traduit par : (GIVE **H : give_up** *act*).

Une fois que l'ensemble de ces Act_X_Task sont rendus disponibles, la tâche peut être exécuter.

4.3.3 Exécution

Nous allons maintenant voir comment s'effectue l'exécution de cette tâche au sein de Shary.

Deux traces d'exécution sont présentées figure 10. La première séquence présente l'exécution nominal du scénario où Jido donne à l'objet à l'homme et celui-ci le prend. La deuxième séquence présente le même scénario mais avec une interruption. Les schémas entre les deux séquences représentent la trace d'exécution au niveau de Shary avec à gauche, ce qui se passe au niveau du robot et à droite se qui se passe au niveau de l'homme, les flèches représentant les échanges entre les deux. Cela représente la décomposition de la tâche en cours ainsi que les moniteurs déclenchés.

Pour commencer, Jido propose l'objet à l'homme, cela se traduit par : (GIVE **R : ask**

act), Act_X_Task qui lui même se décompose en un SPEAK dont l'exécution va amener à l'instanciation de (SPEAK **R : realize** *act*) car c'est une tâche autonome (on va donc utiliser le schéma de droite de la figure 9). Sur le schéma sont également représenté les moniteurs associés à l'Act_X_Task (GIVE **R : ask** *act*), ils se traduisent par :

- **H : ask** *refuse* : l'homme refuse la tâche,
- **H : ask** *agree* : l'homme accepte la tâche,
- **H : realize** *act* : l'homme commence à réaliser la tâche (e.g. l'homme commence à prendre la bouteille).
- **R : ask** *impossible* : la réalisation de l'Act_X_Task échoue, cela peut provenir par exemple d'un échec permanent de la synthèse vocale.

Le fait que l'homme reste devant le robot est considéré comme une acceptation de la tâche : **H : ask** *agree*, le robot passe donc à la réalisation de la tâche en déclenchant les moniteurs associés.

On peut remarquer une situation intéressante dans la seconde trace d'exécution. A un moment, l'attention de l'homme est perturbée par l'intervention d'un nouvel arrivant (la personne ne continue plus à regarder le robot). Le robot traduit cela par une suspension de la tâche par l'homme : (GIVE **H : suspend** *act*) qui va aboutir comme cela est défini par le schéma de communication à une confirmation de la suspension de la tâche par le robot : (GIVE **R : suspend** *confirm*). Quand l'attention de l'homme revient, (GIVE **R : resume** *act*), la réalisation de la tâche est relancée : (GIVE **R : realize** *act*).

Ensuite se réalise la tâche d'échange entre l'homme et le robot dans laquelle le robot va relâcher l'objet alors qu'il aura détecté la prise d'objet par l'homme. Une fois cet échange réalisé la tâche GIVE va se terminer (GIVE **R : end** *act*) le robot prévenant l'homme par l'intermédiaire d'un message.

Nous avons présenté ici le déroulement de cette tâche de manière symbolique au sein de Shary, nous reviendrons plus en détail sur l'implémentation de la tâche dans le chapitre consacré à Jido dans la partie expérimentations page 155.

4.4 Conclusion

Nous avons présenté ici Shary le système d'exécution et de supervision de tâches que nous avons développé. Ce système permet la gestion de la communication de la communication nécessaire au bon déroulement de la tâche et l'affinement des tâches en sous-tâches.

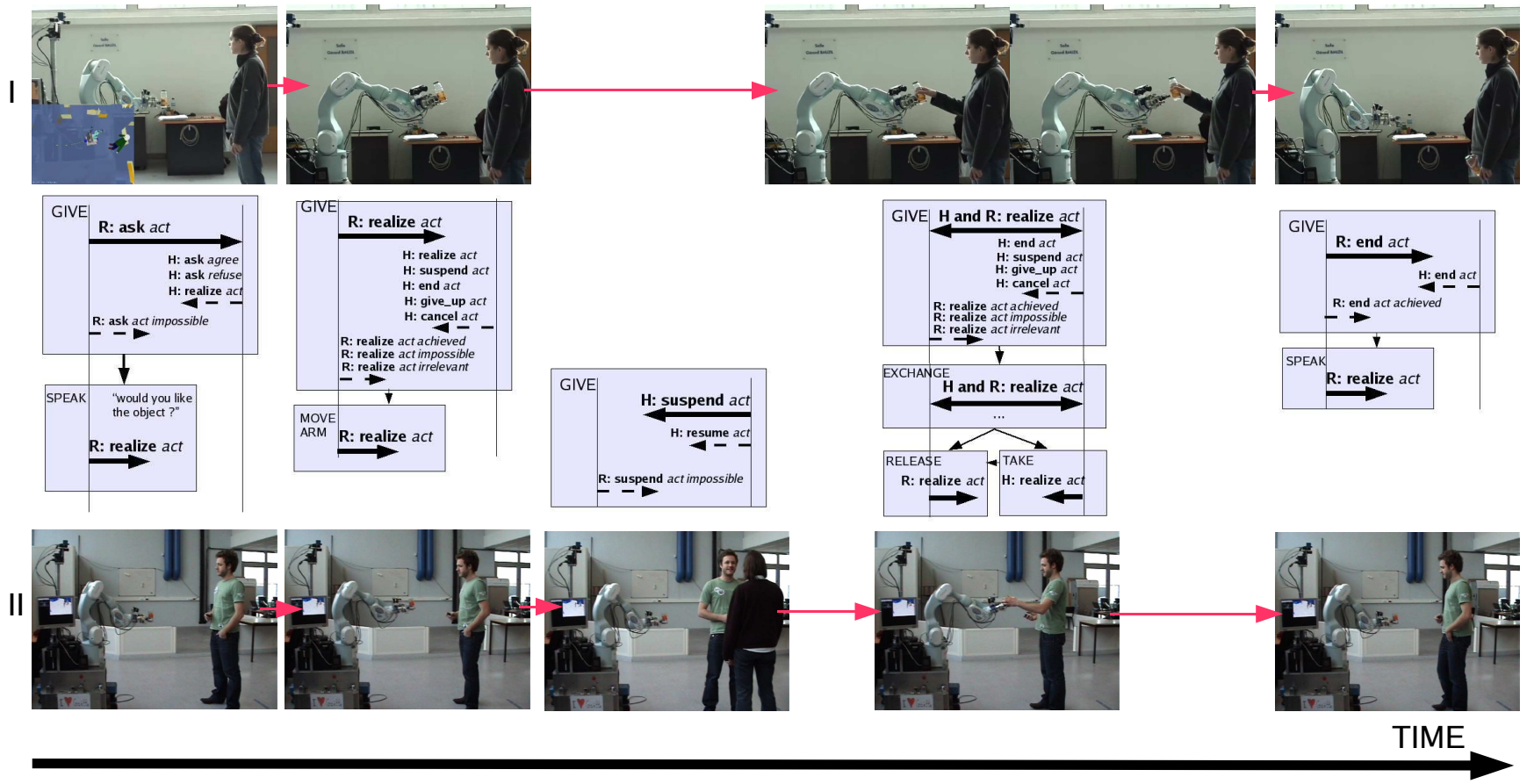


FIG. 10 – Dans cet exemple, on demande à Jido de donner une bouteille à la personne se trouvant près de lui. Jido attend qu'une personne se présente devant lui, i.e. reste assez longtemps debout devant lui. En utilisant la synthèse vocale, il demande alors à la personne si elle désire la bouteille. Si la personne accepte, Jido lui tend alors la bouteille . La personne doit alors prendre la bouteille (la prise est détectée par les capteurs d'effort sur la pince), et le robot lâche l'objet. Comme illustré par la seconde série d'images, il peut se produire une interruption dans l'exécution si l'homme est perturbé par une autre personne. Cependant si l'homme revient, le robot relance l'exécution de la tâche. Les vidéos correspondantes sont disponibles à cette adresse : http ://www.laas.fr/∼mransan/jido/data/en/videos.php

Chapitre 5

Modéliser la communication

Nous décrivons ici le schéma de communication que nous avons défini pour une tâche jointe, il est relié à notre traduction de la définition de croyance mutuelle.

Le schéma de communication que nous définissons peut être comparé au langage de discours artificiel de Sidner [Sid94] ou aux protocoles pour les actions jointes définies par [KHC02]. Cependant nos schémas essaient de prendre en compte les possibles incompréhensions (et leurs modélisations) et les procédures nécessaires à leur gestion. Ces schémas peuvent également être rapprochés des niveaux de *grounding* de Clark : *attend*, *identify*, *understand* and *consider*.

5.1 Croyance Mutuelle

La théorie de l'intention jointe (*joint intention theory*) établit qu'une action jointe ne peut être vue comme une collection d'actions individuelles mais que des agents travaillant ensemble doivent partager des croyances. Cette notion est représentée par la notion de croyance mutuelle que nous avons étudié section 1.1.2 page 26.

Nous avons vu section 1.3.4 page 43 qu'une notion similaire est décrite par Clark sous la forme du *grounding criterion* ou *common ground* par le fait que lorsque nous addressons un message à quelqu'un, il ne suffit pas de l'exprimer, il faut aussi s'assurer qu'il a été compris. Lors d'un dialogue, notre but est alors d'atteindre le *grounding criterion*, i.e. que nous et notre interlocuteur nous partagions la croyance que nous nous soyons compris.

Rappelons les définitions de croyance mutuelle MB (*mutual belief*) et de croyance mutuelle unilatérale BMB (*unilateral mutual belief*) d'après Kumar [KHCM02], nous avons :

$$
\begin{array}{rcl}
(BMB\ x\ y\ p) & \triangleq & (Bel\ x\ p\ \wedge\ (BMB\ y\ x\ p)) \\
(MB\ x\ y\ p) & \triangleq & (BMB\ x\ y\ p)\ \wedge\ (BMP\ y\ x\ p) \\
 & \triangleq & (Bel\ x\ p\ \wedge\ (BMB\ y\ x\ p))\ \wedge \\
 & & (Bel\ y\ p\ \wedge\ (BMB\ x\ y\ p)) \\
 & \triangleq & (Bel\ x\ p\ \wedge\ (Bel\ y\ p\ \wedge\ (BMB\ x\ y\ p)))\ \wedge \\
 & & (Bel\ y\ p\ \wedge\ (Bel\ x\ p\ \wedge\ (BMB\ y\ x\ p)))
\end{array}
$$

Selon cette définition pour obtenir la croyance mutuelle de p entre deux agents, nous avons besoin des croyances des deux agents. Même si le fait que cela soit défini par des croyances et non par des connaissances n'oblige pas à être sûr de leur véracité, nous considérons cette notion

difficile à modéliser dans notre contexte. En effet, nous considérons le robot comme un agent individuel ayant ses propres croyances, capacités de raisonnement et de perception. Dans ces conditions le robot et le système décisionnel qui lui est propre n'ont à proprement parler pas accès aux connaissances des autres agents.

Soit h un homme avec lequel le robot va interagir et r le robot lui même. Les croyances concernant l'homme auxquelles le robot a accès ne sont jamais $(Bel\ h\ p)$ mais $(Bel\ r\ (Bel\ h\ p))$, i.e. l'accès aux connaissances de l'homme se fait à travers la perception du robot. De la même manière nous n'aurons pas $(Bel\ h\ (Bel\ r\ p))$ mais $(Bel\ r\ (Bel\ h\ (Bel\ r\ p)))$. Nous retrouvons ces notions dans la définition de la croyance mutuelle unilatérale :

$$
\begin{aligned}
(BMB\ x\ y\ p) &\triangleq (Bel\ x\ p\ \wedge\ (BMB\ y\ x\ p)) \\
&\triangleq (Bel\ x\ p)\ \wedge\ (Bel\ x\ (BMB\ y\ x\ p)) \\
&\triangleq (Bel\ x\ p)\ \wedge\ (Bel\ x\ (Bel\ y\ p\ \wedge\ (BMB\ x\ y\ p))) \\
&\triangleq (Bel\ x\ p)\ \wedge\ (Bel\ x\ (Bel\ y\ p))\ \wedge \\
&\quad\ (Bel\ x\ (Bel\ y\ (Bel\ x\ p\ \wedge\ (BMB\ y\ x\ p)))) \\
&\triangleq (Bel\ x\ p)\ \wedge\ (Bel\ x\ (Bel\ y\ p))\ \wedge \\
&\quad\ (Bel\ x\ (Bel\ y\ (Bel\ x\ p)))\ \wedge \\
&\quad\ (Bel\ x\ (Bel\ y\ (Bel\ x\ (Bel\ y\ p\ \wedge (BMB\ x\ y\ p)))))
\end{aligned}
$$

Nous avons décidé de considérer une forme "tronquée" de cette croyance mutuelle unilatérale que nous nommerons UMB :

$$
\begin{aligned}
(UMB\ x\ y\ p) \triangleq\ & (Bel\ x\ p)\ \wedge\ (Bel\ x\ (Bel\ y\ p))\ \wedge \\
& (Bel\ x\ (Bel\ y\ (Bel\ x\ p)))\ \wedge \\
& (Bel\ x\ (Bel\ y\ (Bel\ x\ (Bel\ y\ p))))
\end{aligned}
$$

Cette définition appliquée à notre contexte d'interaction homme-robot est illustrée figure 11. Elle implique que nous donnions au robot (grâce à ses capacités de perception, de dialogue, de décision) des connaissances concernant :

- $(Bel\ r\ p)$: ses propres croyances (que nous pourrions assimiler à des connaissances si l'on considère que le robot connaît son état).
- $(Bel\ r\ (Bel\ h\ p))$: ses croyances concernant l'homme,
- $(Bel\ r\ (Bel\ h\ (Bel\ r\ p)))$: ses croyances concernant les croyances que l'homme a vis-à-vis de lui-même (i.e. le robot),
- $(Bel\ r\ (Bel\ h\ (Bel\ r(Bel\ h\ p))))$, ses croyances concernant les croyances de l'homme concernant les croyances du robot concernant l'homme.

La prise en compte de ces croyances fait clairement références au "perspective taking", i.e. la possibilité pour un agent de prendre la perspective d'un autre agent ([NF05, TCB+05]) que nous avons évoqué section 2.4.2 page 66.

Prenons maintenant deux examples pour illustrer ce que ces croyances peuvent signifier au sein d'une application robotique.

Premièrement, supposons que Rackham soit dans une pièce et que son système de détection de visages soit activé, ce qui signifie qu'il est capable de détecter les visages des gens qui sont dans le champ de vue de la caméra. Une personne h souhaitant interagir avec Rackham s'approche du robot. Si on considère le fait p : *h is detected by r*, à partir du moment où la personne est détectée par le robot, nous pouvons décider que $(Bel\ r\ p))$. Dans le cas où aucun autre retour n'est donné par le robot, aucune autre croyance ne peut être déduite de la situation.

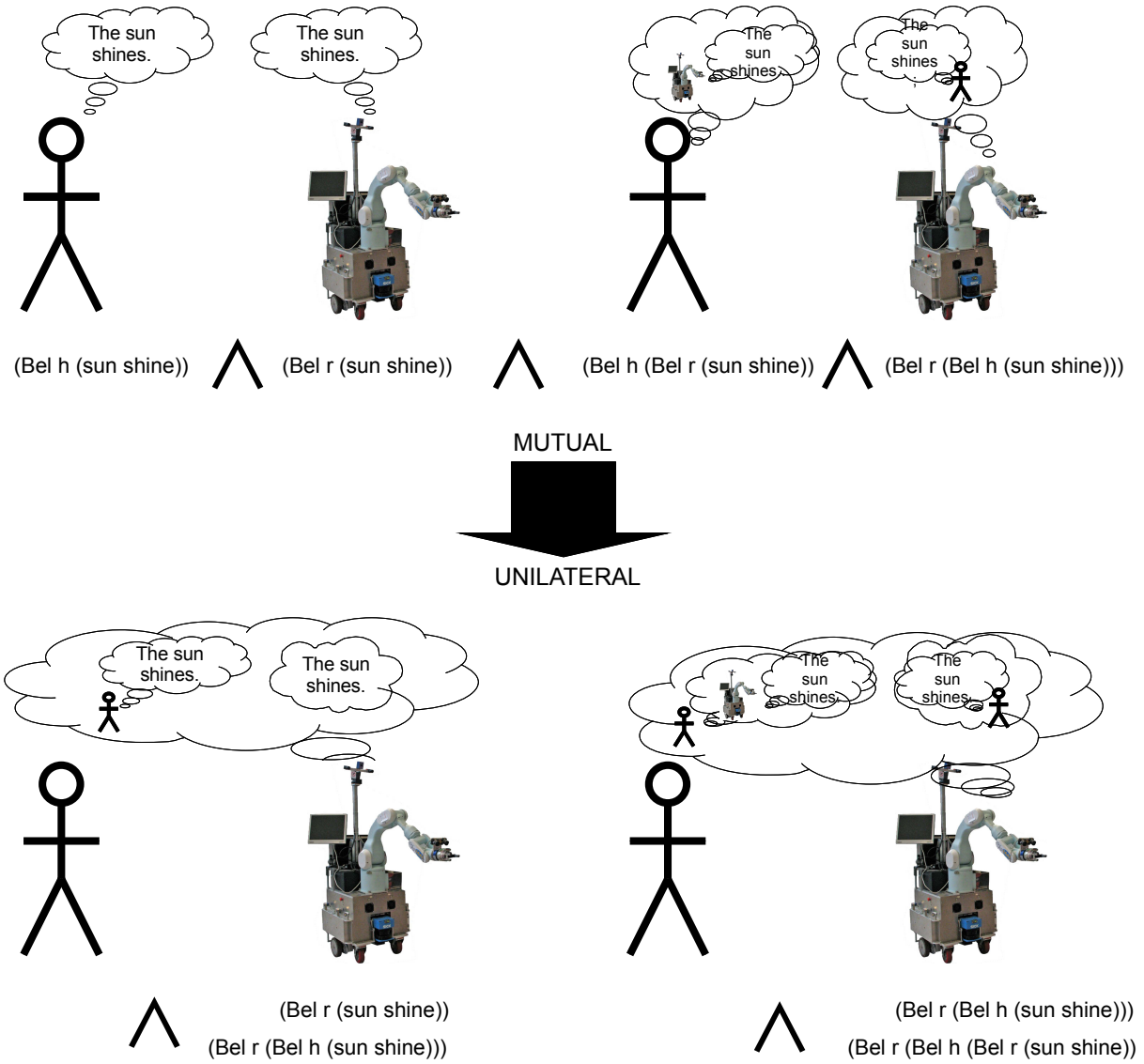


FIG. 11 – De la croyance mutuelle à la croyance mutuelle unilatérale : ce schéma représente la traduction effectuée entre la croyance mutuelle et la croyance mutuelle unilatérale que nous utilisons où toutes les croyances sont représentées à travers la perception du robot.

FIG. 12 – Rackham sending (or not) feedback to the people via the use of an external representation

Si le robot affiche maintenant des informations sur ses détections (par l'intermédiaire d'une représentation externe, par exemple en affichant le flux vidéo de la caméra, un carré s'affichant sur chacun des visages détectés comme montré figure 12), de nouvelles croyances peuvent être ajoutées. Si l'on présume que l'information ainsi affichée est vue par l'homme, nous ajoutons $(Bel\ r\ (Bel\ h\ p))$ car l'homme est prévenu qu'il a été détecté. Dans ce cas (car la croyance concerne une détection du robot), elle est équivalente à $(Bel\ r\ (Bel\ h\ (Bel\ r\ p)))$. La dernière croyance est $(Bel\ r\ (Bel\ h\ (Bel\ r\ (Bel\ h\ p))))$, elle représente le fait que l'homme est prévenu que le robot sait qu'il a l'information.

Un deuxième exemple peut être pris dans le contexte de guide à un moment où Rackham et le visiteur se sont mis d'accord sur la destination. Pour atteindre la destination désirée, le robot calcule une trajectoire, ainsi nous avons $(Bel\ r\ (trajectoryknown))$. Ce que nous avons observé lors de nos expériences c'est qu'il est intéressant de donner à l'homme une information sur la trajectoire. Etant donné que les visiteurs ne sont pas avertis du fonctionnement du robot : de ses contraintes cinématiques, de ses algorithmes de planification, les mouvements initiaux étaient souvent mal compris ou mal interprétés. La solution que nous avions trouvé était d'afficher la trajectoire calculée par le robot sur une carte de l'environnement affiché sur l'écran tactile pour bien montrer au visiteur que le robot allait l'amener à la bonne destination. Le fait d'afficher la trajectoire (à un endroit visible par l'homme) peut être interprété comme $(Bel\ r\ (Bel\ h\ (Bel\ r\ (trajectoryknown))))$. Si l'on considère que l'homme a intégré la trajectoire nous pouvons ajouter $(Bel\ r\ (Bel\ h\ (trajectoryknown)))$.

Ces deux exemples illustrent la nécessité pour le robot d'intégrer ce genre de croyance à un niveau décisionnel. Parfois, ces croyances sont nécessaires à la réalisation de la tâche : ainsi lorsque Jido bouge son bras en direction de l'homme, il doit s'assurer qu'il est perçu par lui. Cette information est de nature critique si l'on considère la sécurité de l'homme et doit donc être une précondition à tout mouvement du bras.

5.2 Déroulement d'un acte de communication

Nous avons défini un acte de communication avec un nom définissant l'objet de l'acte et une étape définissant son évolution. Nous allons maintenant proposer deux évolutions possibles pour un acte de communication, la première évolution définissant le cas où le robot est à l'origine de

l'acte de communication et le second le cas où c'est l'homme qui est à l'origine de la communication. Nous verrons ce que ces deux évolutions traduisent en terme de croyances et feront le lien avec la notion de croyance mutuelle que nous avons défini précédemment.

Nous nous inspirons de deux idées communément admises dans la littérature. La première est que lorsque l'on communique on s'attend à recevoir une réponse en retour (cette réponse pouvant relever de la *positive* ou de la *negative evidence*). De plus, nous connaissons plus ou moins ces réponses attendues et utiliser cette connaissance peut bien entendu aider le système. La deuxième idée est basée sur le fait que la communication repose sur un échange d'information, il n'est donc pas suffisant de considérer que l'information a été envoyée, il faut également s'assurer dans la mesure du possible que l'information ait été correctement comprise par l'interlocuteur.

Nous introduisons également une notion de clarté des actes de communication réalisés par l'homme. Cette clarté peut être reliée à la notion de convention. Une convention représente la manière dont une action est généralement exécutée, elle est aussi définie comme un moyen de coordination [Cla96]. L'un des problèmes dans notre cas est le manque de conventions existant aujourd'hui entre l'homme et le robot, i.e. il n'est pas évident de savoir ce qui peut être considéré ou non comme conventionel dans un contexte d'interaction homme-robot.

Une étape d'un acte peut être interprétée comme non-claire quand le robot a une idée de la signification de l'étape de l'acte reçu mais qu'il n'en est pas certain.

5.2.1 Communication R→H

Le premier cas considère que le robot est à l'initiative de la communication, il est décrit figure 13.

Si nous analysons les croyances à chacune des étapes. Au début, le robot a sa propre croyance concernant un attribut dont il considère qu'il a la valeur $val1$: $(Bel\ r\ (att\ val1))$. En exécutant l'acte de communication, il rend cette information disponible pour son interlocuteur. Ici peut prendre sens la notion de publicité de l'information, ainsi une information privée n'aurait pas a être rendu public et donc n'aurait pas à être communiquée.

Le robot attend au moins un accusé de réception indiquant que l'homme a reçu l'information ce qui se traduit par $(Bel\ r\ (Bel\ h\ (Bel\ r\ (att\ val1))))$.

Le robot peut également recevoir ce que nous appelons une réponse non-clair de la part de l'homme, ce qui se traduirait par $(Bel\ r\ (Bel\ h\ (att\ val2)))$. Cette réponse peut indiquer l'homme est d'accord ou non avec le robot concernant la valeur de l'attribut att, ainsi nous pouvons avoir $val1 = val2$ ou $val1 \neq val2$.

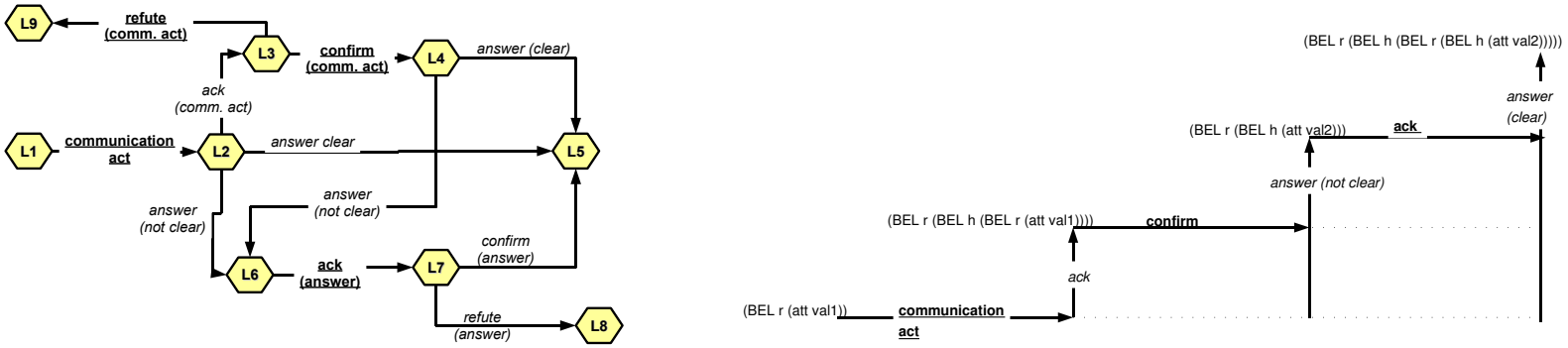
Cela traduit le fait que le robot pense que l'homme à une croyance $(Bel\ r\ (Bel\ h\ (att\ val2)))$ mais qu'il ne sait pas si c'est cette croyance que l'homme a souhaité lui donner, i.e., nous n'avons pas $(Bel\ r\ (Bel\ h\ (Bel\ r\ (Bel\ h\ (att\ val2)))))$ qui traduirait le fait que l'homme pense qu'il a donné l'information au robot qu'il avait l'information.

La dernière proposition relève que l'homme peut donner une réponse claire, qui ajouterait ce $(Bel\ r\ (Bel\ h\ (Bel\ r\ (Bel\ h\ (att\ val2)))))$ à l'ensemble des croyances.

Les croyances sont obtenues dans l'ordre :

1. $(Bel\ r\ (att\ val1))$,
2. $(Bel\ r\ (Bel\ h\ (Bel\ r\ (att\ val1))))$,
3. $(Bel\ r\ (Bel\ h\ (att\ val2)))$,
4. $(Bel\ r\ (Bel\ h\ (Bel\ r\ (Bel\ h\ (att\ val2)))))$.

Si $val1 = val2$, ces quatre croyances définissent $(UMB\ r\ h\ (att\ val1))$ comme expliqué section 5.1 page 105.



Exemple :
Le robot désire obtenir l'engagement de l'homme à participer à la tâche avec lui.

communicative act : "Voulez-vous participez à cette tâche avec moi ?"
ack communicative act : "Tu veux que je réalise la tâche avec toi ?"
confirm communicative act : "Oui !"
answer clear : "Ok, allons-y !"
answer not clear : Le robot détecte un signe de la tête qu'il interprète comme un signe affirmatif.
ack answer : "Alors c'est d'accord ?"
confirm answer : "Oui !"
refute answer : "Non !"

FIG. 13 – Déroulement de la communication et évolution du processus de grounding si le robot est l'instigateur de la communication. En **gras souligné** on retrouve les actes du robots en *italique* ceux l'homme. Il y aura croyance mutuelle (i.e. accord entre les agents) si $val1 = val2$ concernant la valeur de l'attribut considéré. Le message que tente de faire passer le robot est que l'attribut prend la valeur $val1$, l'homme peut ensuite montrer son accord sur ce fait, on aura alors $val1 = val2$ ou son désaccord i.e. l'homme peut considérer et donner l'information au robot que $val2 \neq val1$.

5.2.2 Communication H→R

Si maintenant l'homme est l'instigateur de la communication, nous considérons un déroulement légèrement différent décrit figure 14. L'une des différences notables par rapport au cas précédent, c'est qu'avant l'apparition de l'acte de communication réalisé par l'homme, il n'y a pas de connaissance a priori du robot.

Nous définissons également que le robot puisse avoir des difficultés d'interprétation des intentions de son interlocuteur, cela se traduit par la distinction entre :

- un acte de communication non-clair : le robot a alors une croyance concernant l'homme $(Bel\ r\ (Bel\ h\ (att\ val1)))$,
- un acte de communication clair : le robot a alors une croyance concernant l'homme $(Bel\ r\ (Bel\ h\ (att\ val1)))$ mais également la croyance que l'homme le sait : $(Bel\ r\ (Bel\ h\ (Bel\ r\ (Bel\ h\ (att\ val1)))))$.

Si l'acte de communication est considéré comme non-clair, le robot envoie un accusé de réception pour confirmer sa perception/compréhension. Il attend alors une confirmation qui arrivera ou non pour envoyer sa réponse.

Les croyances dans ce cas s'obtiennent dans cet ordre :

1. $(Bel\ r\ (Bel\ h\ (att\ val1)))$
2. $(Bel\ r\ (Bel\ h\ (Bel\ r\ (Bel\ h\ (att\ val1)))))$
3. $(Bel\ r\ (att\ val2))$
4. $(Bel\ r\ (Bel\ h\ (Bel\ r\ (att\ val2))))$

Nous observons également que le robot doit prendre une décision avant d'envoyer sa réponse, ce qui implique qu'un mécanisme de décision doit être implémenté pour inférer $(Bel\ r\ (att\ val2)))$, i.e. si $val2 = val1$ or $val2 \neq val1$. Par exemple quand l'utilisateur demande au robot d'effectuer une tâche, le robot doit évaluer sa capacité à la réaliser dans le contexte. Il envoie ensuite son accord ou son refus.

5.2.3 Exemple d'instanciation

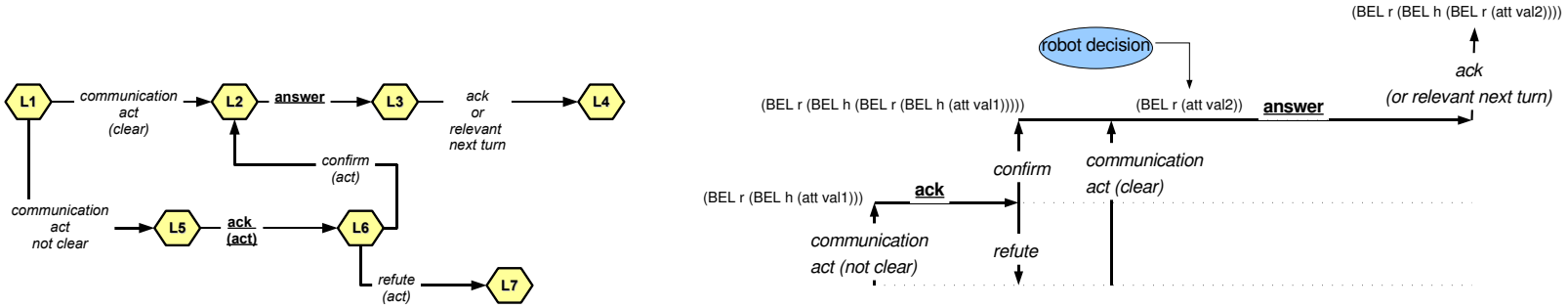
La figure 15 montre un exemple d'instanciation pour l'acte **ask** selon que l'homme ou le robot soit à l'initiative de l'acte.

5.3 Schéma de communication pour tâche jointe

Nous avons défini le déroulement possible d'un acte de communication. Nous rappelons que dans notre optique de communication relative au contrôle de l'exécution d'une tâche, nous avons défini les actes suivants :

- **ask-task** : proposer une tâche,
- **propose-plan** : proposer un plan pour réaliser la tâche,
- **modify-plan** : proposer une modification du plan courant,
- **give-up** : abandonner la tâche (e.g. car elle est impossible). Pour le robot, cela sera le moyen d'annoncer qu'il est dans l'impossibilité d'accomplir la tâche.
- **cancel** : annulation de la tâche (i.e. abandon volontaire, la tâche n'est plus pertinente),
- **end_task** : terminaison de la tâche,
- **realize-task** : réalisation de la tâche (et si besoin annonce du début de cette réalisation).

Mise à part pour l'acte **realize-task** pour lequel nous ne définissons pas de déroulement, tous les actes sont définis avec le déroulement proposé à la section précédente.



Example :
L'homme désire la suspension d'une tâche jointe.

communicative act clear : "Rackham, il faut suspendre la tâche"
answer : "Ok"
ack answer or relevant next turn : "Nous sommes d'accord"
communicative act not clear : L'homme arrête la réalisation de la tâche.
ack communicative act : "Souhaitez-vous suspendre la tâche ?"
confirm act : "Oui, il y a un problème."
refute act : "Non, non, on continue !"

FIG. 14 – Déroulement de la communication et évolution du processus de grounding si l'homme est l'instigateur de la communication. En **gras souligné** on retrouve les actes du robots en *italique* ceux l'homme. Il y aura croyance mutuelle (i.e. accord entre les agents) si $val1 = val2$ concernant l'attribut considéré.

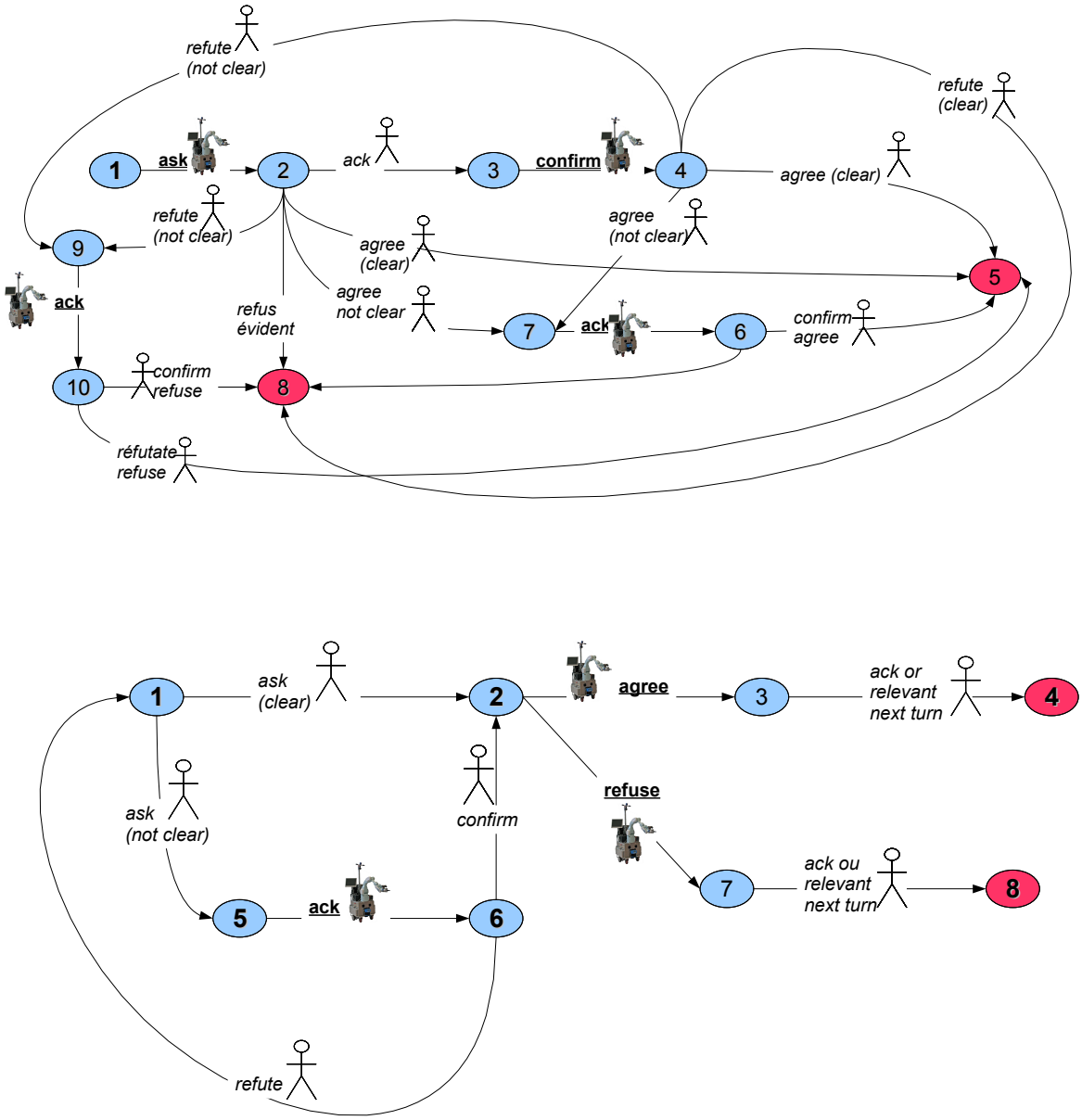


FIG. 15 – Ces deux schémas montrent l'instanciation de l'acte **ask-task** avec ses différentes étapes. Ils montrent le "*turn-taking*" d'une part lorsque le robot est l'instigateur de l'acte (en haut) et d'autre part lorsque l'homme est l'instigateur de l'acte (en bas).

Ensuite, nous établissons un ordre dans le déroulement de ces actes : **ask-task** d'abord une demande est effectuée pour connaître la décision concernant la participation à la tâche, **propose-plan** on se met d'accord sur le plan pour réaliser la tâche, **realize-task** ensuite on réalise la tâche, **modify-task** si au cours de l'exécution un nouveau plan est disponible, on le propose, **suspend-task** il est aussi possible de suspendre la tâche et ensuite **resume-task** de la reprendre, enfin il est possible de communiquer sur le fait que : **end_task** la réalisation est terminée, **cancel** la réalisation est annulée, **give-up** la réalisation est abandonnée.

Ensuite, il est bien entendu que nous définissons des raccourcis qui nous semblent pertinents, ainsi si l'on détecte que l'homme commence à réaliser la tâche alors qu'on vient de lui demander si il voulait y participer, nous considérons son engagement comme acquis et lançons également la réalisation de la tâche.

Cela définit un schéma basique de turn-taking pour la réalisation d'une tâche jointe.

5.4 Interaction avec le dialogue et la planification

Lorsque l'on parle communication, vient immédiatement la référence au dialogue. Nous avons conçu notre système en gardant à l'esprit qu'il serait nécessaire de le relier à un système de dialogue. Nous avons travaillé sur ce sujet en collaboration avec l'Université de Bielefeld. Nous présentons ici brièvement ces travaux, une version plus étendue peut être trouvée en annexe page 167. Le but de ce travail est d'établir un mécanisme pour permettre une coordination et un "*turn-taking*" flexible au cours des phases de négotiation entre les hommes et le robot. Cela peut se produire à plusieurs niveaux de la hiérarchie de tâches.

5.4.1 Contexte

Nous avons défini un premier protocole d'interaction et d'échange de données entre le dialogue et la supervision tel que la figure 16 le schématise. Le planificateur va intervenir ici comme une ressource appelée par le superviseur. Le système de dialogue va être chargé de la négociation avec l'utilisateur jusqu'à l'obtention d'une réponse claire telle que définie section 5.2.2 page 111. Ainsi l'on considère que le système de supervision, lorsqu'une réponse proviendra du dialogue, aura toujours accès à une réponse claire de l'utilisateur. La supervision et le dialogue vont interagir à propos de la tâche et des tâches jointes, et les tâches participatives, mais pas les tâches individuelles autonomes.

5.4.2 Scénario

Le scenario est le suivant : Jido attend dans une pièce. Luis arrive et demande à Jido de lui ramener un coca. Jido accepte. Il se rend à la cafeteria, trouve le coca et le ramène au bureau de Luis. Jido donne alors le coca à Luis.

Une variation de ce scénario peut être que lors du trajet retour de la caféteria, le robot croise Luis, un nouveau plan est alors généré considérant que l'échange peut avoir lieu de suite.

5.4.3 Déroulement

Les figures 17, 18, 19, 20 montrent des déroulements commentés et argumentés de cette tâche. Pour un souci de simplicité dans le schéma, l'étape de l'acte est inséré au début du nom de l'acte i.e AGREE_ASK_TASK est équivalent à (**ask-task** *agree*).

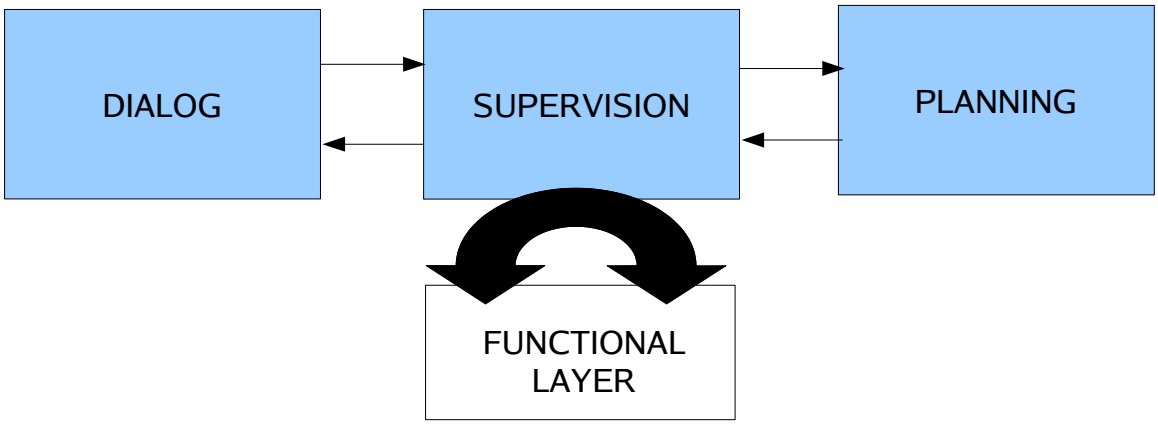


FIG. 16 – Le but de ce travail est d'établir un mécanisme pour permettre une coordination entre la supervision et le dialogue, la planification étant une ressource utilisable par la supervision

L'une des remarques que l'on peut faire sur l'intégration du dialogue, c'est qu'il nous faut prévoir la possibilité pour l'homme de donner plusieurs informations (et donc pouvoir prendre cela en compte au niveau des schémas de communication).

5.5 Conclusion

En fait, en décrivant les schémas de communication tel que nous les avons défini, nous faisons abstraction des croyances sous-jacentes sur lesquelles nous nous sommes appuyés. Nous pensons que ce lien peut être reconstruit et qu'il peut faciliter et aiguiser l'utilisation du schéma construit. Toutefois, il faut pour cela définir une représentation des connaissances permettant la prise en compte des différents point de vue définis par la croyance mutuelle. C'est à dire avoir une base de faits et des processus de raisonnement capables de gérer les différentes perspectives d'une même croyance, travail que nous n'avons pas encore effectué.

Une étude sur l'utilisation du schéma de communication décrit en lien avec le dialogue a été effectuée au cours d'une collaboration avec l'université de Bielefeld et résumée ici. Elle montre qu'à partir des schémas de communication nous pouvons établir des schémas de "*turn-taking*" cohérents dans notre cadre de gestion de tâches.

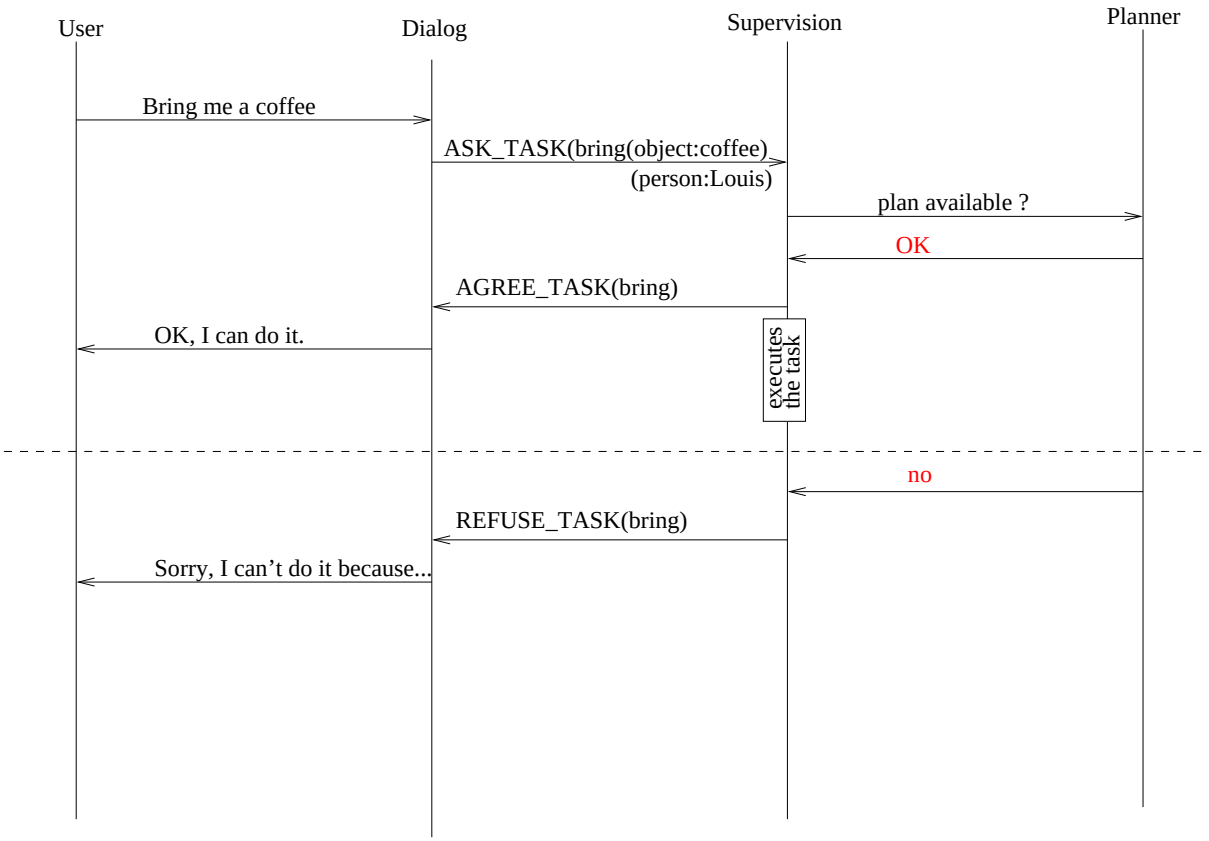


FIG. 17 – L'homme demande au robot de lui amener un café : ASK_TASK, le robot accepte : AGREE_ASK_TASK ou refuse : REFUSE_ASK_TASK selon sa disponibilité ou sa capacité à réaliser la tâche (pour cela il vérifie si un plan est possible ou non).

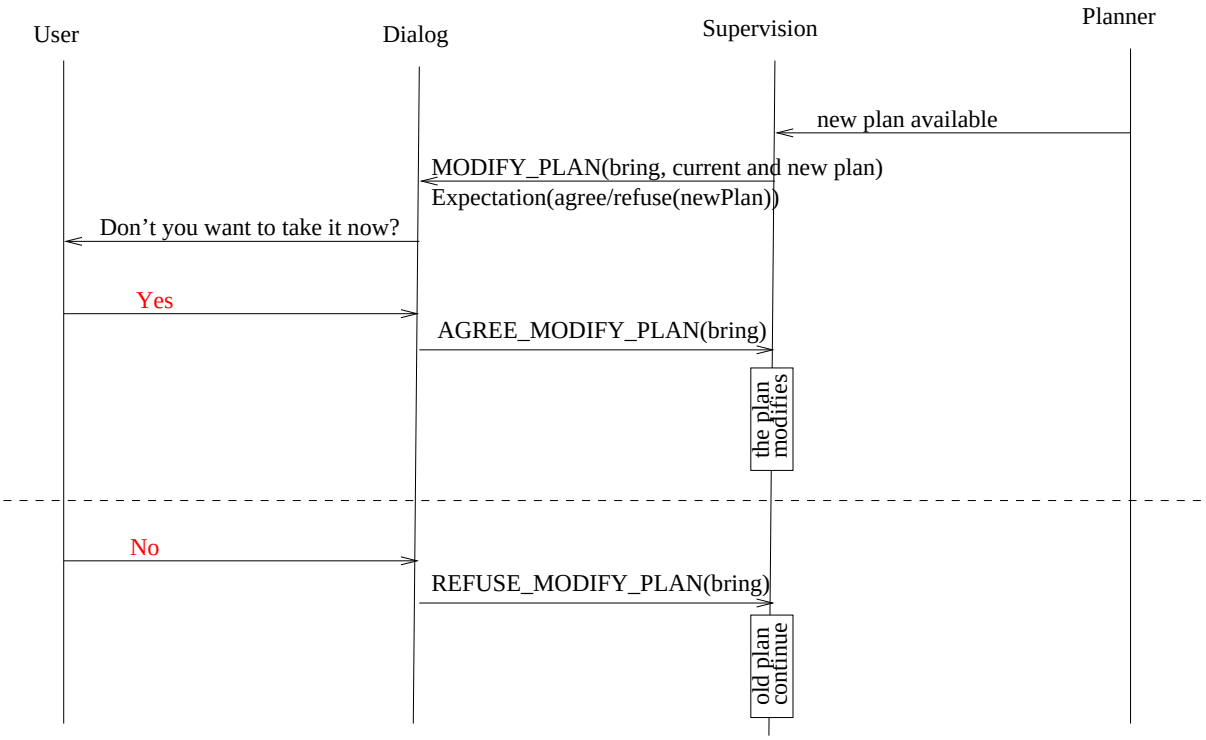


FIG. 18 – Le robot rencontre Luis avant d'atteindre l'endroit où il doit amener le café, cela induit la génération d'un nouveau plan. Le robot propose alors à l'homme de lui donner la café maintenant : MODIFY_PLAN. L'homme peut alors accepter : AGREE_MODIFY_PLAN ou refuser : REFUSE_MODIFY_PLAN. Selon la réponse, le nouveau plan sera utilisé ou non. Ceci montre une situation où c'est le robot qui va générer l'acte de communication et proposer une nouvelle manière de réaliser la tâche.

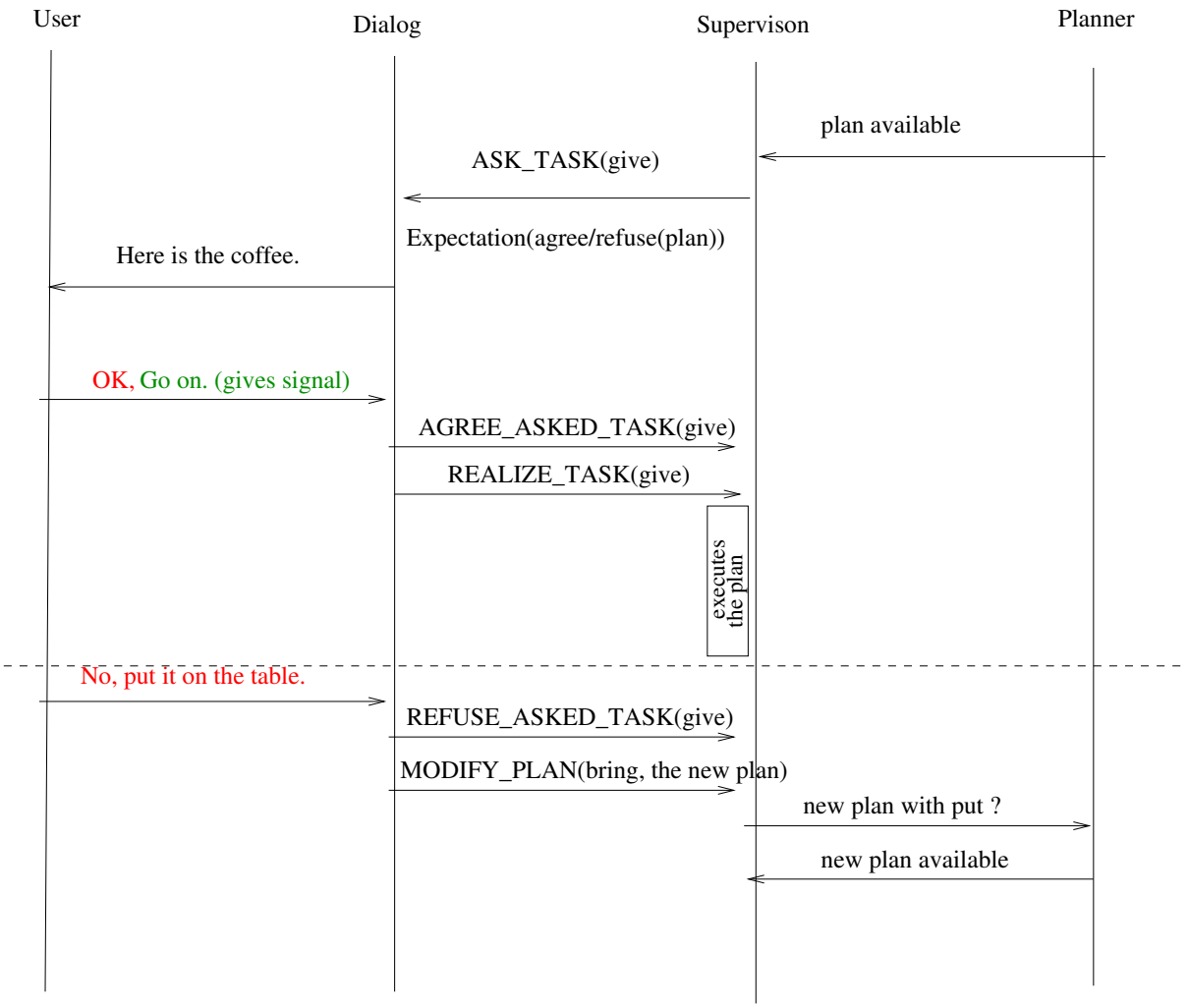


FIG. 19 – Arrivé près de Luis, le robot lui propose de lui donner son café : ASK_TASK. Dans le premier cas, l'utilisateur accepte : AGREE_ASK_TASK et l'échange a lieu. Dans le deuxième cas, l'homme refuse : REFUSE_ASK_TASK et propose une solution alternative : MODIFY_PLAN : le robot peut poser le café sur la table.

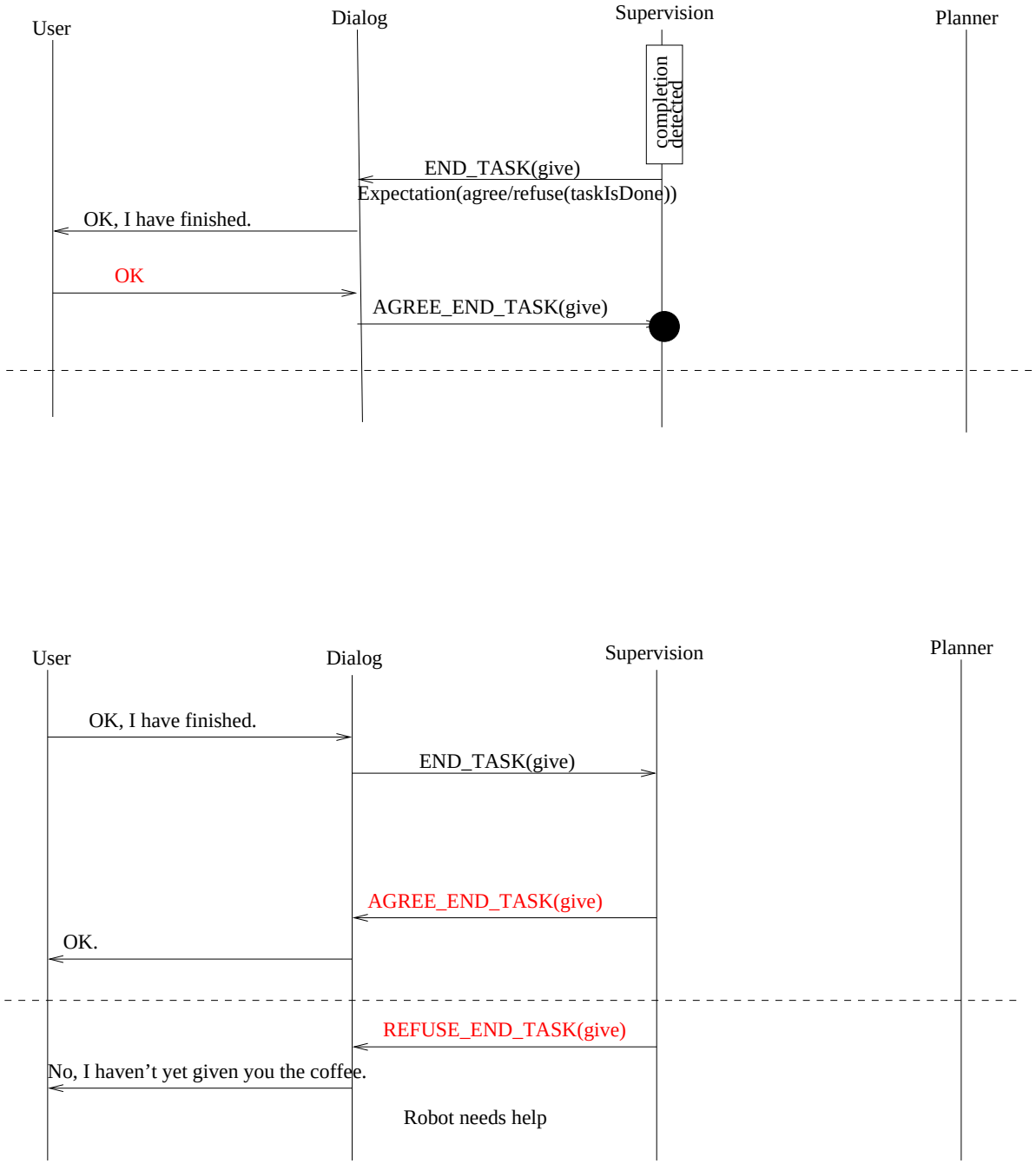


FIG. 20 – Deux cas de terminaison de tâches : Dans le premier cas, c'est le robot qui informe Luis que la tâche est terminée : END_TASK. Dans le deuxième cas, c'est Luis qui indique qu'il a terminé : END_TASK, le robot est en accord : AGREE_END_TASK ou pas REFUSE_END_TASK avec cette information.

Conclusion

Le but de notre travail est la prise en compte de l'interaction en général et de l'homme en particulier au sein de l'architecture LAAS.

Réaliser une tâche interactive ou collaborative nécessite la prise en compte d'un interlocuteur ou d'un collaborateur. Dans les théories ou les systèmes étudiés dans la première partie et dans la littérature en général, cet interlocuteur ou collaborateur devient un agent du système. Dans teamwork, à chaque agent est associé un proxy, composant logiciel intermédiaire qui va gérer la communication avec l'agent d'un côté et le système de l'autre. De même dans HRI/OS, nous retrouvons des "robot agents" et des "human agents" dont l'attribution des rôles est réalisée par un système central. Notre système se définissait dès le départ d'une manière différente, bien sûr il est nécessaire de définir des agents et notre système les définit en terme d'IAA pour InterAction Agent au sein duquel le robot lui-même est un agent particulier. Cependant c'est cet agent particulier qui va représenter le système, c'est cet agent particulier qui va assurer la mise à jour de la représentation des autres agents par l'intermédiaire de sa perception (et non pas l'agent lui même) et la communication va se dérouler entre l'agent robot et les autres agents (et non pas entre un système central et l'agent). Nous rejoignons ici le travail de Breazeal dans lequel le robot modélise non seulement ses croyances mais également les croyances de son tuteur. Dans notre système, ces données sont stockées sous forme d'IAA dans la base de faits.

La nature des données disponibles sur les agents va donner plus ou moins de possibilités au système. Les connaissances générales sur les positions géométriques et/ou topologiques, objets possédés,... et sur les capacités des agents permettent de planifier une tâche. Cependant, notre système permet de structurer les connaissances sur les tâches en cours et à réaliser pour le robot, cela va nous permettre de traduire des contraintes telles que celles mentionnés par "*intention-to*" dans Shared-Plans. Par exemple, le robot ne peut être disponible pour une nouvelle tâche car il en réalise déjà une. De plus, puisque les plans calculés par HATP nous donnent les actions du robot mais également celles de l'homme ou parce que nous aurons eu l'information, nous pouvons avoir accès aux tâches en cours de réalisation par l'homme et cela peut nous permettre de prendre en compte des contraintes du type "*intention-that*" : le robot ne va pas réaliser telle action car cela gènera la réalisation de telle autre action par l'homme et même permettre la génération de comportement d'aide : le robot va réaliser telle action car cela va aider l'homme à réaliser l'action prévue par leur plan commun. Ainsi, une fois les données rendues disponibles, ces contraintes peuvent être prises en compte à plusieurs endroits dans notre système : faire partie du contexte et donc des contraintes associées à une recette, être utilisées par HATP pour le calcul de son plan, et permettre à l'Agenda la génération de nouvelles tâches.

Il faut bien remarquer (et c'est une remarque qui existait déjà dans [GK96]) que ces données peuvent ne pas être correctes, ce qui est important c'est d'être capable de prendre en compte une modification de l'une d'elle. Bien entendu, cela va induire que le robot peut "se tromper".

La communication est inhérente à l'interaction, cette communication nécessite la définition d'un protocole de communication [KHCM02], d'une convention [Cla96], d'un moyen d'interpréter

la communication [Loc98]. Nous avons choisi de représenter cela sous la forme de schémas de communication. Ces schémas permettent l'interprétation de la communication et la génération des actes attendus. Les actes définis au sein de ces schémas vont constituer le langage disponible pour l'interaction (au même titre que les langages définis dans [FTB01] et [Sid94]). A travers les actes de communication que nous avons défini au sein de ces schémas, nous avons défini un langage particulier car nous avons choisi de nous restreindre à l'étude de la communication autour de la gestion d'une tâche, mais le système permet l'intégration d'autres actes de communication.

Ces schémas nous permettent de prendre en compte une grande variété de situations d'interaction. Ils permettent de coder des conventions qui peuvent être adaptées à chacun des agents avec qui l'on désire communiquer. Ils permettent également de gérer l'équilibre de la communication entre les interlocuteurs par exemple en fonction de la tâche : on peut imaginer qu'une tâche d'apprentissage pilotée par un homme se traduise par un schéma très directif de la part de l'homme.

Nous ne pouvons nous placer dans un système où l'on considère la communication comme parfaite, les agents forcément de bonne volonté, etc, comme on le retrouvait par exemple dans la théorie de l'intention jointe. Les schémas particuliers que nous avons définis, permettent de prendre en compte un processus de compréhension et d'expression des niveaux de consentement tel qu'on a pu le voir dans la théorie de l'action jointe. Nous proposons également une représentation des connaissances mises en jeu lors de ce processus, connaissances couvrant les connaissances du robot, et les connaissances du robot sur les autres agents mais également les connaissances que les autres agents peuvent avoir du robot, ceci pouvant permettre une mise en perspective lors de prise de décisions.

Une des particularités de notre système est que nous ne séparons pas la gestion opérationnelle et la gestion de l'interaction, i.e., nous n'avons pas un "*interaction manager*" découplé d'un "*task manager*", et un Act_X_Task représentant un acte de communication se décompose en sous-tâches au même titre que tous les Act_X_Task. Ainsi la réalisation d'un (**ask-task** *act*) pourra se traduire par un mouvement possiblement couplé avec de la parole, toutes les possibilités sont laissées au programmeur.

Il est intéressant de noter qu'il y a une certaine homogénéité dans la définition des buts en terme de buts persistants tels que définis par la théorie de l'intention jointe dans tous les systèmes rencontrés, que les buts soient individuels ou joints, de même on retrouve la décomposition hiérarchique.

Si maintenant nous revenons sur les 10 challenges mentionnés précédemment :

1. "pour être membre d'une équipe, un agent doit satisfaire la convention basique de s'engager dans une activité à faire en commun" : nous nou splaçons clairement dans ce cadre,
2. "les agents doivent être capables de modéliser les autres participants de manière adéquate" : dans ce cadre nous proposons un modélisation des agents en terme d'IAA, la mise à jour des données des agents s'effectuent par l'intermédiaire des perceptions du robot sans recours à des intermédiaires,
3. "les agents doivent être mutuellement prévisibles" : d'une part les schémas de communication permettent d'établir des schémas de convention avec les autres agents, et d'autre part notre traduction de la croyance mutuelle exhibe la prise en compte des croyances des autres agents,
4. "les agents doivent pouvoir être dirigés" : nos schémas de communication offrent la possibilité à l'homme d'agir sur la tâche en cours de diverses manières,
5. "les agents doivent être capables d'exhiber, de rendre visibles leurs intentions à leurs partenaires" : nous prenons cet aspect en compte à deux niveaux : dans les schémas de commu-

nication pour lesquels le robot “communique/informe” sur son état et celui de la tâche. Il est également pris en compte au niveau des planificateurs (HATP, HAMP) qui sont censés produire des plans “lisibles”.

6. “les agents doivent être capable d’interpréter les signaux envoyés par leurs partenaires” : nous avons également intégré cet aspect. Toute tâche collaborative au niveau du robot comprend non seulement des actions pour réaliser la tâche mais également pour “monitorer” l’homme et son implication dans la tâche commune.
7. “les agents doivent avoir les moyens de négocier sur leur but” : cet aspect est traité au niveau du planificateur. Dans notre schéma de communication, nous avons plusieurs actes qui permettent cette négociation (**ask-task**, **propose-plan**, **modify-plan**).
8. “les algorithmes de planification et de gestion de l’autonomie doivent supporter la collaboration” : là aussi nous apportons une première réponse au niveau des planificateurs que nous développons et qui raisonnent explicitement sur l’homme.
9. “les agents participent à la gestion de l’attention des autres agents (possiblement en les prévenant quand un événement important se produit)” : nos schémas de communication le prennent en compte explicitement. Pour le moment, son élaboration est le fait du programmeur.
10. “les agents doivent pouvoir contrôler le coût de leurs activités coordonnées”, HATP aussi bien que HAMP utilisent des critères de coûts pour guider la recherche d’un plan acceptable. Il serait également envisageable d’introduire des notions de coûts au niveau des schémas de communication, ou dans le choix des recettes (choix des médias de communication utilisés par exemple).

Nous n’avons pas (encore) envisagé explicitement la prise en compte de plans et/ou d’informations partiels, cela et une meilleure prise en compte des nouvelles informations à leur arrivée pourraient clairement bénéficier au système. En ce sens, l’introduction d’actes de communication tel que : **inform** et **get-info** est envisagé à court terme.

Une autre piste de développement est la prise en compte des croyances de manière explicite au niveau des schémas de communication.

Troisième partie

Expérimentations

Introduction

Dans cette partie, nous allons étudier plus précisément les différentes mises en oeuvre que nous avons effectués sur les robots du LAAS. Dans un premier chapitre, nous allons présenter ces robots et les différentes fonctionnalités dont ils disposent. Ensuite, nous présenterons de manière plus précise Rackham et l'expérimentation qui s'est déroulé à la Cité de l'Espace. Enfin, nous reviendrons sur Jido et sur la tâche qui consiste à apporter un objet à quelqu'un.

Chapitre 6

Robots et Fonctionnalités

Dans ce chapitre, nous allons présenter les robots qui ont servi à notre étude : Rackham et Jido ainsi que les différentes fonctionnalités présentes sur ces plate-formes. Ces fonctionnalités sont présentées figure 1, elles regroupent des capacités de localisation, de navigation, de mouvement, ainsi que des capacités d'interaction.

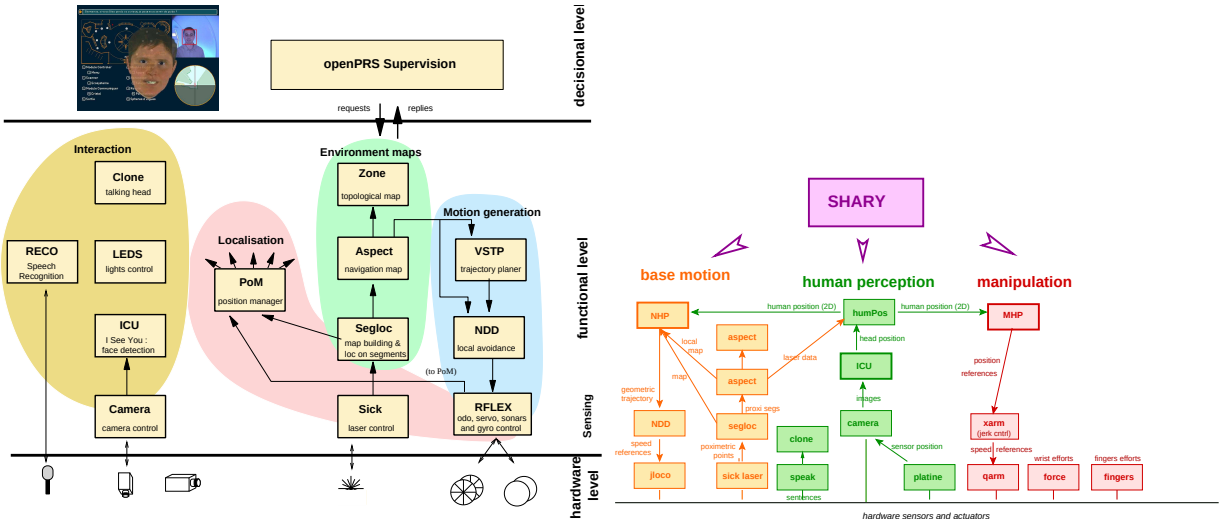


FIG. 1 – Deux instanciations de l'architecture LAAS

6.1 Présentation des robots

Nous avons été amenée à utiliser deux robots au cours de ce travail. Rackham qui a servi à l'expérimentation de la Cité de l'Espace et Jido que nous utilisons dans le cadre du projet Cogniron[7].

6.1.1 Rackham

Rackham (figure 2) est un robot de type B21r fabriqué par la société iRobot. La base mesure 1,20 m de haut et à un diamètre de 52 cm. Elle supporte un mât équipé d'une sorte de casque

[7] http ://www.cogniron.org

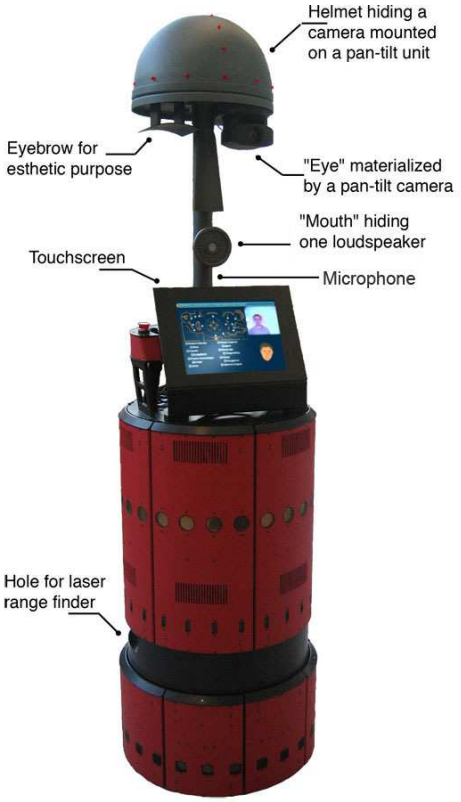


FIG. 2 – Rackham et ses équipements

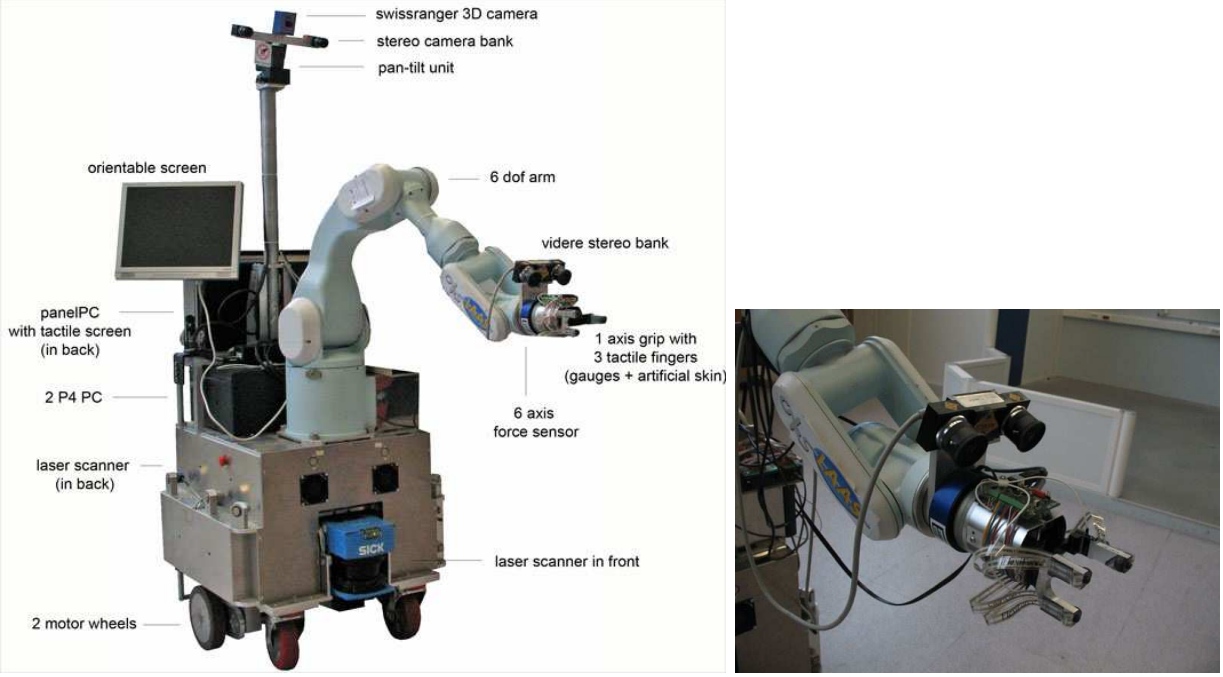


FIG. 3 – Jido et ses équipements

réalisé pour les besoins de la Cité de l'Espace. Il est équipé de deux pcs (un mono CPU et un bi-CPU P3 850 MHz) et d'ethernet sans fil. L'équipement standard a été complété d'une caméra Sony EVI-D70, d'un écran tactile ELO touch, d'une paire de haut-parleurs, d'un gyroscope, d'un capteur laser Sick et d'un micro.

6.1.2 Jido

Jido (cf figure 3) est composé :
- d'un mat surmonté d'une platine pan-tilt supportant un banc stéréo et une caméra Swiss-Ranger,
- un bras à 6 degrés de liberté (6R) équipé d'un capteur d'effort 6 axes au niveau du poignet et d'une pince à un axe munie de trois doigts dotés de capteurs tactiles,
- un banc stéréo Video est fixé sur le poignet (dirigé vers l'extrémité du bras),
- deux lasers Sick à l'avant et à l'arrière du robot
- un panel PC disposant d'un écran tactile,
- un écran orientable pour fournir des retours visuels à l'homme interagissant avec le robot.

6.2 GenoM

Il a déjà été question de l'architecture LAAS section 3.2.1 page 81. Avant de parler des fonctionnalités disponibles sur les robots, nous précisons la notion de module et décrivons brièvement GenoM.

`GenoM` (Generator of Modules) [FHC97][FHM05] est un outil de développement pour des robots, permettant de générer des *modules* destinés au niveau fonctionnel de l'architecture LAAS. Il fournit divers services, tels que des moyens de communication entre les modules, des interfaces de communication pour un superviseur ou pour un opérateur par un langage de scripts.

Un module `GenoM` comporte les éléments suivants :

- Des *tâches d'exécution* exécutent, de manière périodique ou non, les fonctions de traitement codées par le concepteur du module. Il peut y en avoir plusieurs fonctionnant en parallèle. Ces fonctions de traitement sont soit permanentes, soit déclenchées par une *requête* adressée au module par l'intermédiaire d'un programme disposant de la bibliothèque d'accès aux services d'un module `GenoM`.
- Une *tâche de contrôle* est chargée de réceptionner les requêtes adressées au module, de vérifier la validité des paramètres, de gérer les conflits ou encore de retourner un bilan sur l'activité en cours.
- Des *posters*, mis à jour régulièrement par les tâches d'exécution, contiennent des informations mises à disposition des autres modules voire d'autres applications telles que celles du niveau décisionnel. Chaque module dispose d'une bibliothèque d'accès aux posters des autres modules actifs. Ainsi, un module de localisation peut par exemple mettre à disposition des autres modules qui en ont besoin la position courante du robot dans un poster.

6.3 openPRS

`openPRS` [Ing05] est un logiciel développé au LAAS-CNRS dédié à la supervision et au contrôle "haut-niveau" de robots mobiles autonomes, et qui exploite un langage consacré à cet usage, le langage *PRS* [ICAR96]. Il dispose de moyens de communication avec `GenoM`, et en particulier d'une bibliothèque d'interface d'accès aux services d'un module (envoi de requêtes à un module, réception des répliques, lecture d'un poster). Une OP (Operation Procedure) d'`openPRS` permet ainsi d'envoyer les requêtes nécessaires à tous les modules impliqués pour la réalisation d'une action.

6.4 Fonctionnalités

6.4.1 Localisation

Géométrique

Plusieurs modules sont impliqués dans la localisation.

Sur Rackham, `rflex` est le module qui s'interface avec les logiciels fournis par iRobot et qui exporte dans un poster la position odométrique du robot, corrigée par la mesure du gyroscope. Cette position fournit une bonne estimée des mouvements du robot. Sur Jido, l'odométrie est fournie par le module `jloco`.

Pour se localiser dans l'environnement, les robots utilisent un laser SICK (controlé par le module `sick`), qui exporte les données laser de manière brut ou sous forme de segments. Le module `segloc` fait correspondre ces segments avec les segments préalablement enregistrés sous la forme d'une carte de l'environnement (un exemple de carte est visualisable figure 4) par une procédure de SLAM classique.

Nous disposons donc de deux éléments d'indication de position du robot : l'un nous informant sur le déplacement relatif du robot, l'autre sur sa position absolue dans l'environnement.

Les différentes positions exportées par `rflex` ou `jloco` et `segloc` sont fusionnées à l'aide du module `pom` (pour "position manager"). Ce module est capable d'intégrer des positions obtenues à des fréquences différentes et de propager les positions à partir de leurs instants d'acquisition. Le module `pom` permet de centraliser les données de position du robot et d'exporter une et seule position qui servira de référence pour tous les modules et le superviseur.

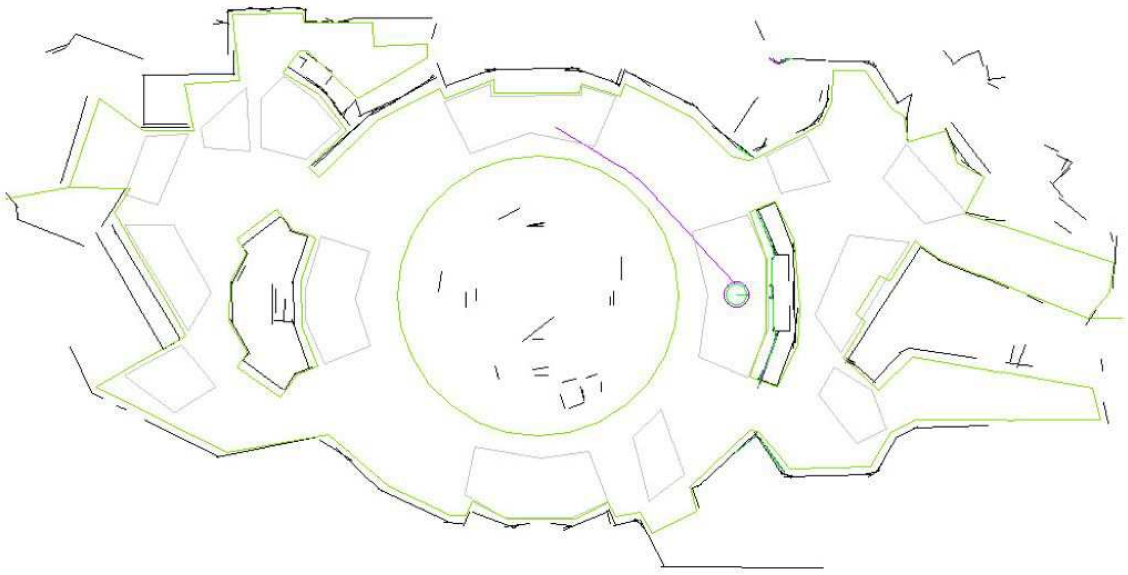


FIG. 4 – Localisation, la carte de l'exposition Mission Biospace : les segments noirs représentent la carte à l'aide de laquelle le robot va se localiser. Les zones matérialisées en vert représentent les zones "VIRTUAL_OBSTACLES" tandis que les formes bleues représentent les zones "TARGETS"

Topologique

Le module `zone` permet de connaître la position topologique du robot en fonction de la position géométrique. Pour cela il est possible de définir des fichiers de "zones".

Par exemple sur Rackham, on peut retrouver :

- un fichier "TARGETS" zones : décrivant les zones d'arrivée du robot,
- un fichier "VIRTUAL_OBSTACLES" zones : décrivant les zones considérées comme des obstacles pour le robot i.e. des zones "dangereuses" pas toujours "visibles" par le robot (comme une table dont on ne verrait que les pieds)
- un fichier "SPECIALS" zones : décrivant des zones où le robot doit adopter un comportement particulier (nous avons par exemple ajouté ce type de zone au niveau des portes pour que le robot ralentisse à ces endroits).

Le module `zone` peut surveiller constamment l'entrée ou la sortie du robot dans une de ces zones et exporte cette information dans un poster. Un exemple de ces zones est montré figure 4.

6.4.2 Détection des obstacles

Bien entendu la detection des obstacles est une fonction critique tant du point de vue de la sécurité des personnes que celle du robot lui-même.

Sur Rackham, après avoir tenté d'utiliser les ultras-sons (inutilisables car trop bruyant) et les bumpers sur les portes (qui étaient trop sensibles, spécialement ceux se trouvant sur les portes inférieures), le seul capteur réellement utilisable était là-encore le capteur SICK. C'est la même chose sur Jido où pour l'instant seul le laser situé à l'avant est utilisé.

Cependant, le laser n'effectue des coupes que sur 180 degrés vers l'avant et à une hauteur approximative de 40 cm. Pour palier partiellement à cette limitation, les données laser sont intégrées au sein d'une carte locale par le module `aspect` et filtrées en utilisant les connaissances provenant de la carte globale (les segments et les obstacles virtuels (cf section 6.4.1)), de plus

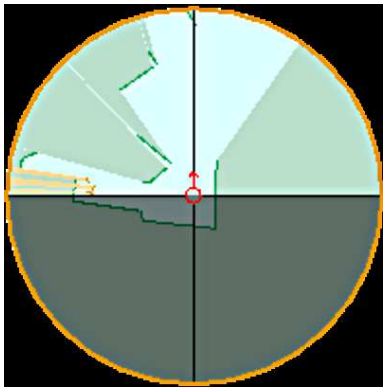


FIG. 5 – Représentation du module aspect : les segments verts correspondent aux obstacles virtuels, les segments jaunes aux obstacles dynamiques

nous n'autorisons pas le robot à effectuer des déplacements en reculant.

Le module `aspect` exporte toutes les 40ms une carte locale des alentours du robot. Cette carte va représenter l'espace libre autour du robot ainsi que les obstacles divisés en deux catégories : les obstacles statiques (qui appartiennent à l'environnement ou les obstacles virtuels) et les obstacles dynamiques (personnes marchant près du robot ou obstacles non-intégrés à la carte globale).

Cette représentation permet au module d'informer le superviseur quand le robot est entouré d'obstacles imprévus mais va également et surtout servir à l'exécution de la trajectoire (cf section 6.4.4).

6.4.3 Détection des hommes

Deux modules permettent la détection des hommes, l'un à l'aide de la vision `icu`, l'autre à l'aide des données fournies par le laser `humPos` (possiblement couplées avec des données fournies par `icu`)

Détection à l'aide de la vision : icu

Le module `icu` inclut différentes modalités de detection et de suivi des hommes par la vision ([BLH04, BLD06, GBLS07]). Ces modalités sont les suivantes :

FACE_DETECTION détection de visages,

MOTION_BLOB_DETECTION détection de mouvement (nécessite que le robot soit à l'arrêt car cette modalité est basée sur des différences d'images),

FACE_TRACKING suivi de visages,

BODY_TRACKING suivi de "body" i.e. se base sur la couleur du visage couplée à un blob couleur initialisé sur le vêtement de l'homme,

FACE_RECOGNITION reconnaissance de visages (illustré figure 6).

Toutes ces modalités peuvent être accédées de manière individuelle par le superviseur. Cependant le module inclut un mode automatique permettant le switch entre les modalités jugées les plus pertinentes.

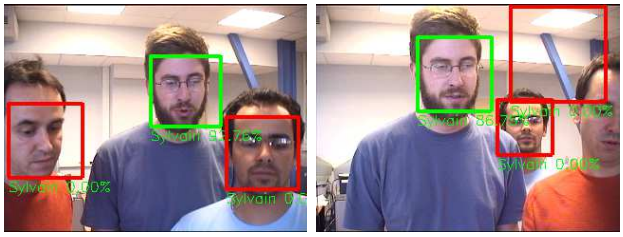


FIG. 6 – Les visages détectés (en rouge) et reconnus (en vert) avec les probabilités associées. La personne recherchée était Sylvain.

Détection à l'aide du laser : humPos

Le module `humPos` permet de détecter les hommes autour du robot, on peut voir des exemples de détections sur la figure 7. La détection s'effectue en trois étapes [SCMU+06] :

- détection des jambes à l'aide du module `aspect` : les segments obtenus par `aspect` qui ne correspondent pas à des segments de l'environnement connu (i.e. à des segments de la carte) sont considérés comme étant potentiellement des jambes. Chaque segment est taggé avec une certaine confiance comme un homme potentiel,
- filtrage des données du sick : parmi les points obtenus directement par le laser, ceux correspondant aux jambes fournissent un motif caractéristique qui permet d'effectuer un filtrage. Le résultat de ce filtrage est ensuite comparé au résultat de l'étape précédente. Les segments ayant des points faisant partis des jambes ont alors une grande probabilité d'appartenir à un homme.
- détection à l'aide d'`icu` : si les hommes trouvés sont également détectés par la vision, leur probabilité de présence est incrémentée de façon importante.

De plus, les mouvements de l'homme dans l'environnement sont étudiés pour trouver la direction des hommes détectés. Si un hommme ne se déplace plus pendant une longue période, il est éliminé de la liste. Au contraire, on attache à une forme se déplaçant une forte probabilité d'être un homme.

La liste des hommes détectés (positions, orientations et probabilités de présence) est mise à disposition dans un poster.

6.4.4 Trajectoire et Mouvement

Nous expliquons maintenant comment sont réalisés le calcul et la réalisation de trajectoires sur nos robots. Nous présentons d'abord une navigation réactive permettant l'évitement d'obstacles et ensuite une méthode de navigation prenant explicitement l'homme en compte.

Navigation réactive : NDD et VSTP

Concernant Rackham et sa fonction de guide, le robot est sensé se diriger vers des zones "cibles" définies dans le fichier TARGET zone (cf section 6.4.1.0). Jido peut utiliser les mêmes modules.

Les mouvements du robot impliquent 3 modules :

`rflex` qui gère le contrôle des moteurs et transfère les vitesses de consigne au microcontrôleur sur Rackham,

`jloco` effectue le même travail sur Jido,

`ndd`] intègre une procédure d'évitement local des obstacles basé sur l'algorithme du Nearness Diagrams [MOM04]. Les zones de visibilité sont fournies par la carte locale du module `aspect` (cf6.4.2).

`vstp` (Very Simple Trajectory Planner) se base sur un graphe de visibilité optimisé grâce à des tables de hashages[8]. Un graphe de visibilité principal est précalculé sur la base des segments de la carte globale. Les obstacles dynamiques peuvent être ajoutés ou enlevés par la suite en temps réel par le superviseur.

La stratégie utilisée par la suite consiste à calculer une trajectoire à l'aide de `vstp` qui va constituer un fil d'Ariane pour l'algorithme de `ndd` : les sommets des lignes brisées constituent les sous-buts. Le superviseur a besoin d'intervenir uniquement si `ndd` se trouve bloqué et ne fait aucun progrès vers le but. Dans ce cas, plusieurs stratégies peuvent être utilisées : calculer une nouvelle trajectoire prenant en compte les nouveaux obstacles, attendre, démarrer une interaction avec les personnes autour, etc. Le mouvement est considéré comme terminé lorsque le robot se trouve à l'intérieur de la zone TARGET (cf section 6.4.1). La vitesse maximum donnée au robot est généralement de 0.4 à 0.6 $m.s^{-1}$ mais cette vitesse est dynamiquement adaptée par `ndd` en fonction de la proximité et du nombre des obstacles détectés.

Navigation en présence de l'homme : NHP

Le module `nhp` implémente le planificateur HAMP (*Human Aware Motion Planner* [SAS+05, SUAS06]). Ce planificateur calcule un chemin dit "acceptable" pour l'homme en réalisant des raisonnements sur l'état de l'homme : sa position, son orientation, son champ de vue, son accessibilité mais aussi éventuellement sa posture et son activité.

3 critères ont été définis à la suite de "*user studies*" :

Critères de sécurité le robot doit éviter d'entrer en collision avec l'homme dans l'environnement et doit si possible éviter de passer près de lui. Ce critère est représenté sous la forme d'une gaussienne centrée sur l'homme.

Critères de visibilité le robot, si cela est possible, doit préférer rester dans le champ de vue des hommes. Ce critère prend en compte les champs de vue des hommes et permet de rendre plus acceptable les mouvements du robot. La grille de visibilité est construite de manière à minimiser l'effort que fait l'homme pour garder le robot dans son champ de vue.

Critères des zones cachées Le robot, si cela est possible, doit éviter d'apparaître brusquement près de l'homme pour éviter tout effet de peur ou de surprise. Pour cela des coûts sont associés aux zones cachés derrières les obstacles présents dans l'environnement. Ces coûts sont inversement proportionnels à la distance entre l'homme et le robot.

Chacun de ces critères est représenté par une grille 2D attachée à chaque homme contenant un ensemble de coûts dépendant de l'état des hommes présents dans l'environnement, mais aussi possiblement de l'homme lui-même. Les grilles sont ensuite combinées sous la forme d'une grille finale dans laquelle est calculée le chemin ayant le moindre coût pour le robot à l'aide d'un algorithme A*. Le chemin ainsi calculé est sans collision et réalisable par le robot tout en satisfiant les 3 critères.

La figure 7 montre des exemples de calcul de trajectoires par `nhp` tandis que la figure 8 montre des exemples grilles calculées à partir des critères définis.

Le chemin calculé est ensuite envoyé au module d'exécution (`rflex` sur Rackham et `jloco` sur Jido) qui va contrôler la vitesse des roues. Lorsque le robot se déplace, si une personne

[8]VSTP est distribué librement : http ://softs.laas.fr/openrobots/.

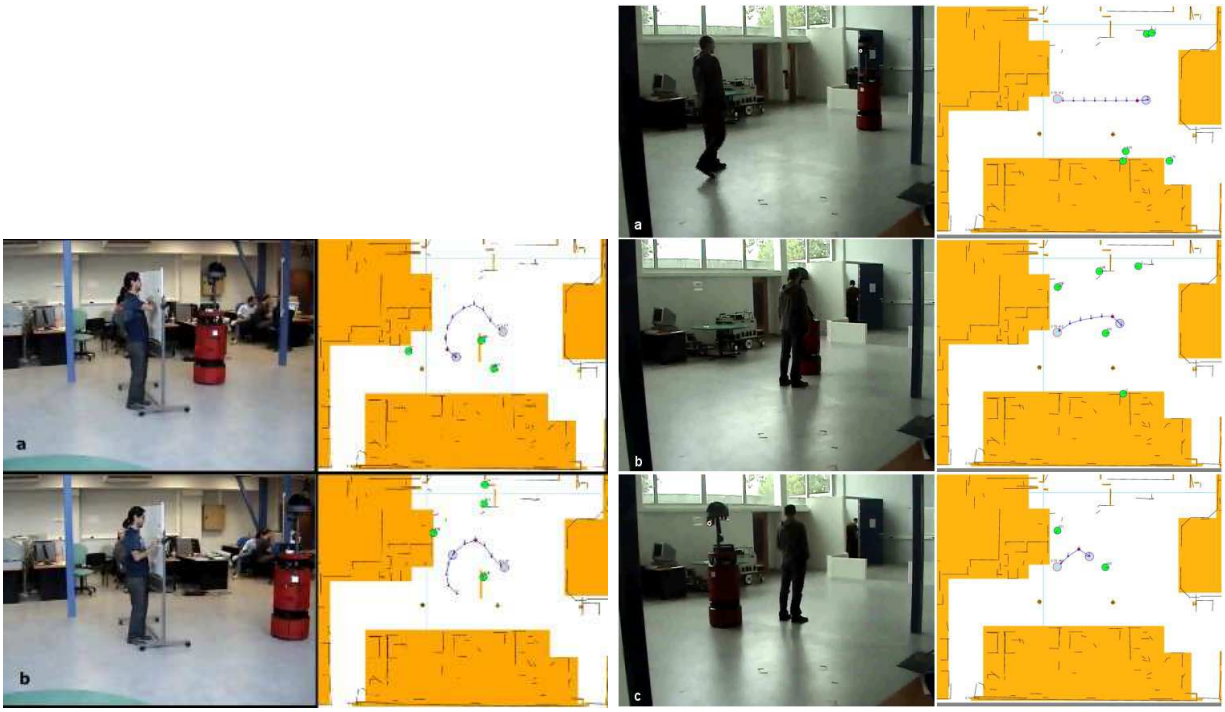


FIG. 7 – NHP : exemples de calcul de trajectoires par `nhp` à l'aide des données de `humPos` (les ronds représentent les hommes détectés)

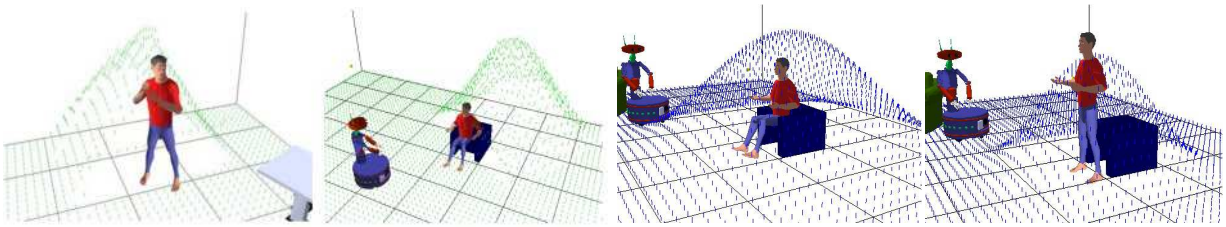


FIG. 8 – NHP : exemples grilles calculées à partir des critères définis : sécurité - visibilité - zone cachée

est détectée ou perdue, ou si une personne change de position ou d'orientation, le planificateur recalcule un nouveau chemin. Pendant l'exécution du chemin, la partie replanifiée est contrainte sur une petite partie avec l'ancien chemin calculé ce qui permet une transition plus naturelle entre l'ancien et le nouveau chemin.

6.4.5 Manipulation

La présence d'un bras manipulateur sur Jido a amené le développement d'une part de modules d'interface et aussi d'un planificateur de trajectoires adaptées à l'interaction.

Modules d'interface

Un certains nombre de modules ont été développé pour le pilotage et l'interfaçage du bras manipulateur :

`qarm` est un module interface avec la carte Mitsubishi de commande du bras PA10 au niveau des coordonnées articulaires. Il offre un accès aux fonctions de la librairie palib founie par Mitsubishi. Il donne accès à la position et à l'état du bras via un poster. Il peut lire un poster afin de faire un suivi de vitesses articulaires.

`xarm` exploite `qarm` pour commander le bras en coordonnées cartésiennes. Il implémente un planificateur temporel de trajectoire pour faire du suivi de trajectoire. La trajectoire peut provenir d'un fichier ou d'un poster. Le planificateur utilise une limitation du jerk, de l'accélération et de la vitesse pour générer des mouvements souples ([HAS06]). Un autre mode de fonctionnement consiste à incorporer ce planificateur de trajectoire dans la boucle de commande en temps réel.

`fingers` est le module qui gère l'ouverture et la fermeture des doigts de la pince. Ses posters fournissent une mesure de la position des doigts et des efforts exercés.

`force` lit les efforts du capteur de force à six composantes situé au niveau du poignet du bras et maintient à jour un poster de force. Il permet de choisir le niveaux de filtrage parmi ceux fournis par la carte d'acquisition et intègre un calcul de transformée de Fourier. Il peut exprimer les forces dans un repère quelconque.

Manipulation en présence de l'homme `MHP`

Le module `MHP` est basé sur la planificateur HAMP (Human Aware Manipulation Planning). Ce module permet de calculer des trajectoires pour le bras manipulateur en prenant en compte la présence de l'homme. Le planificateur HAMP, Move3D et un solveur de cinématique inverse généralisée sont utilisés.

Le planificateur prend en entrée un modèle de l'homme et la configuration du robot comprenant la configuration de la base et la configuration du bras. A partir d'un modèle 3D enregistré de l'environnement, une "*roadmap*" est construite comprenant des points de passage pour le robot (et son bras) acceptable pour l'homme. Le planificateur calcule le modèle cinématique inverse de manière itérative en calculant des mouvement locaux entre différents noeuds de la roadmap jusqu'à ce que la configuration finale soit atteinte. Si durant le mouvement du bras, la position du but bouge, le planificateur s'adapte à ce changement.

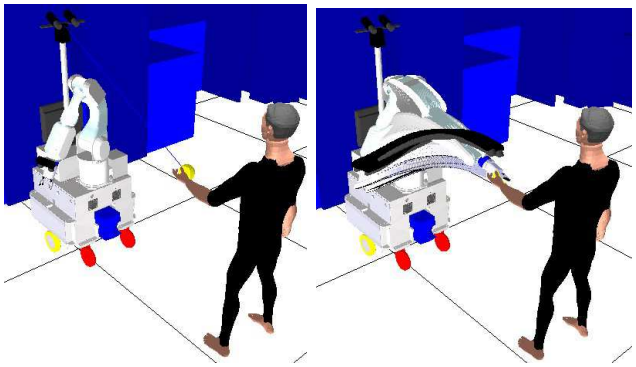


FIG. 9 – MHP : un exemple de mouvement calculé par MHP

6.4.6 Clone parlant

Une synthèse vocale enrichie d'un visage 3D animé affiché sur l'écran est utilisé. Ce clone parlant montré figure 10 a été développé par l'Institut de la Communication Parlée[9]. Il est basé sur un modèle articulatoire très précis des postures d'un locuteur parlant sur lequel est projeté une texture 3D. A partir d'un texte donné, la synthèse va produire un fichier "voix" coordonnées avec des informations concernant les mouvements du visage (machoire, dents, lèvres, etc.).

La direction des yeux et de la tête peuvent être contrôlées dynamiquement.

6.4.7 Reconnaissance vocale

Un système de reconnaissance vocale du français développé à l'IRIT[10] par l'équipe SAMOVA a été intégré pour permettre aux gens d'interagir de manière plus naturelle avec le robot par l'intermédiaire de commandes simples. Ce système est basé sur le système de reconnaissance open-source Julian[11].

Trois ressources linguistiques sont nécessaires au processus de reconnaissance : les modèles acoustiques (les phonèmes), un lexique des mots à reconnaitre associé à leurs prononciations et une grammaire proposant un ensemble de règles d'assemblage de ces mots et de leur signification.

La prononciation des mots a été obtenue à partir de BDLEX et ses ressources lexicales et phonologiques disponibles à l'IRIT. Les modèles acoustiques ont été récupérés directement des précédents travaux de l'équipe Samova de l'IRIT. Ils ont été développés pour être indépendants du locuteur et ne nécessitent pas d'adaptation en fonction de l'environnement acoustique, d'une tâche spécifique,... (ie pour être le plus générique possible). Lorsqu'une personne parle, sa voix est enregistrée, le système cherche la séquence de mots qui convient le mieux en fonction de la grammaire. Ensuite, la phrase la plus probable est convertie en une commande envoyée au superviseur.

6.4.8 L'interface graphique

Les systèmes d'interface utilisés précédemment sur les robots étaient davantage des interfaces robot-programmeur que des interfaces robot-utilisateur. Elles permettaient de manière visuelle

[9] http ://www.icp.inpg.fr
[10] Institut de Recherche en Informatique de Toulouse
[11] http ://julius.sourceforge.jp/en/julius.html

FIG. 10 – clone parlant de l'ICP Grenoble

de vérifier le bon fonctionnement du robot. Rackham évoluant dans un endroit public, l'interface se devait d'être plus interactive et montrer un choix d'informations pertinentes pour l'utilisateur.

Parmi les informations disponibles, on peut citer : la carte de l'environnement (avec la position du robot, sa trajectoire, sa destination), les informations du laser montrant les éléments perçus par le robot comme présenté figure 5, le flux vidéo de la caméra (avec les éventuelles informations de détection de visages). De plus d'autres informations peuvent être affichées sous la forme de texte ou de question et la synthèse vocale couplée au clone parlant est disponible. La figure 11 montre un exemple d'affichage telle qu'il était proposé aux visiteurs de la cité de l'Espace.

La pertinence et l'importance relative de chacune de ces informations changent au cours du temps et plus précisément du contexte dans lequel va se trouver le robot. Lorsqu'il s'agit

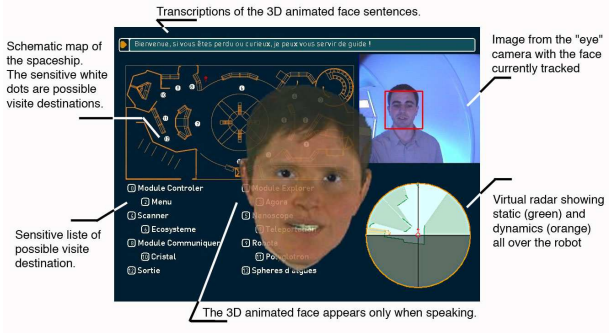


FIG. 11 – Un exemple d'affichage de l'interface

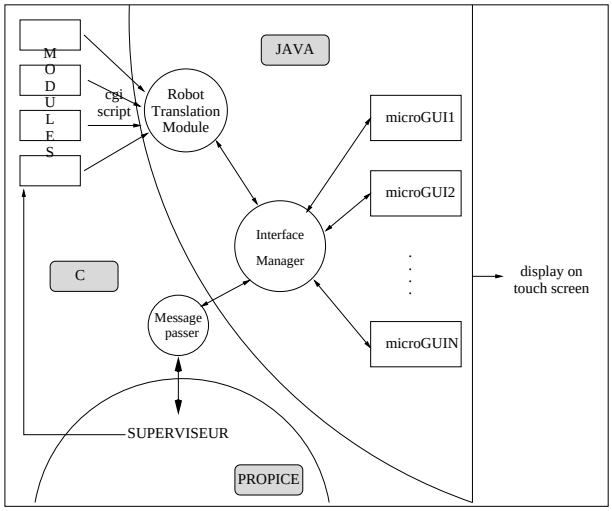


FIG. 12 – intégration de l'interface graphique

d'interaction, il peut être intéressant de mettre en exergue certaines informations lorsqu'elles prennent de l'importance. On peut ainsi donner de l'importance au clone parlant lorsqu'il donne des explications ou mettre l'accent sur les éléments perçus par le laser pour expliquer un éventuel blocage.

De par sa fonction de coordinateur, le superviseur connaît : l'état courant du robot et de son environnement (attente de mission, navigation, évitement d'obstacles, détection d'une personne,...). Ces informations sur le contexte peuvent permettre d'enrichir et de contrôler de façon fine les modalités d'interaction avec les visiteurs. Ce contrôle des interactions est réalisé par le superviseur lui-même. Pour cela, nous avons développé une interface graphique modulaire et dynamiquement configurable par le superviseur.

Dans un premier temps nous allons expliquer les différents éléments du système et les liens qui existent entre eux. Ensuite, nous expliquerons les possibilités que nous avons définies pour la gestion de l'interface par le superviseur.

Définition des composants d'interface

Le schéma 12 présente le système défini. L'interface est décomposée en éléments d'interface que l'on a appelé MicroGUI (pour micro Graphical User Interface). Une MicroGUI correspond à une fenêtre. Elles sont en général définies pour donner un retour visuel à l'utilisateur, et éventuellement lui proposer une interaction. On peut citer par exemple un radar affichant l'environnement du robot (MicroGUI présentée figure 5), ou encore une carte correspondant à ce que le robot connaît de son environnement, sa position, la trajectoire qu'il prévoit de suivre, etc. Ces MicroGUIs sont décomposés en deux parties : une vue et un contrôleur qui permettent de discerner le traitement graphique de la communication.

Le Message Passer permet la discussion avec le robot. C'est par lui qu'on reçoit les ordres provenant du superviseur, et par son intermédiaire que s'effectuent les retours au superviseur du robot.

Les "data" correspondent à des données exportées par les différents modules du robot. Aujourd'hui GenoM produit des cgi pour chaque module et permet d'accéder aux données en passant par *picoweb*, un mini serveur web développé pour nos applications. Ces "Data" utilisent donc ce biais pour récupérer le contenu des données (en XML), parsent ce contenu et stockent les

informations pour qu'elles puissent être utilisables dans les différentes microGUIs.

Le RobotTranslationModule consiste à procurer une interface simple pour la gestion des "Datas". Cette classe contrôle toutes les "Datas" instanciées à un moment donné, grâce au langage défini plus loin.

Enfin, l'InterfaceManager répartit au niveau des MicroGUIs, les informations reçues d'une part par le Message Passer, d'autre part par les RobotTranslationModule.

Sur cette base, les microguis qui ont été développées sont les suivantes : une carte de localisation stylisée, une carte locale basée sur les données du module `aspect`, le clone, la vidéo et d'autres concernant les messages : l'une permettant d'avoir un log, une autre permettant d'afficher un message et une autre permettant de poser une question.

Le langage

Concernant les composants Nous avons déjà indiqué qu'il pouvait être intéressant d'afficher ou non un composant, par exemple il est inutile de continuer à afficher la liste des zones si la destination a déjà été choisie. Nous avons également remarqué lors de nos expériences à la Cité de l'Espace, qu'il pouvait être utile de pouvoir limiter à certains moments les possibilités offertes à l'utilisateur, par exemple désactiver le clic sur la carte lorsqu'une destination a déjà été choisie ou a contrario de les étendre, par exemple si le robot se perd, il peut demander au visiteur de l'aider à se relocaliser en cliquant sur cette même carte.

Sur la base de ces constatations, nous avons défini un ensemble de messages permettant de paramétrer :

- l'interactivité du composant (est-il "cliquable" ?),
- l'affichage ou non du composant,
- le choix de la taille et de la position du composant,
- l'affichage ou non d'une information au sein du composant (ex : la trajectoire sur la carte).

Cela signifie qu'aucun des composants d'interface ne peut s'afficher ou afficher une information sans qu'il en ait reçu l'ordre du superviseur.

Concernant la lecture et la mise à jour des données Auparavant la lecture des données des modules nécessaires aux microguis se faisaient de manière automatique par lecture régulière des posters.

Nous avons trouvé plusieurs limitations à cette solution. D'une part, l'information est alors toujours présente/affichée même si elle n'a plus lieu d'être, par exemple quel est l'intérêt de continuer à afficher une trajectoire alors qu'elle est terminée, or pour l'interface il était alors impossible de savoir que cette information n'était plus pertinente. D'autre part, on peut vouloir faire des calculs sans forcément vouloir afficher l'information, par exemple calculer plusieurs trajectoires et n'afficher que celle que l'on aura choisi.

Pour ces raisons, nous avons également ajouté la possibilité pour le superviseur de paramétrer la lecture des données i.e. lecture au coup par coup ou de manière régulière et dans ce dernier cas la période de rafraichissement. Tous ces paramétrages peuvent se faire de manière dynamique au cours de la session.

Un exemple de microgui : "Dialog"

Dans une première version, nous avions défini de manière stricte les composants de type question (e.g. un composant microGUI par question et les réponses aux questions renvoyées au superviseur était prédéfini). Or au fur et à mesure des expériences, le nombre des questions

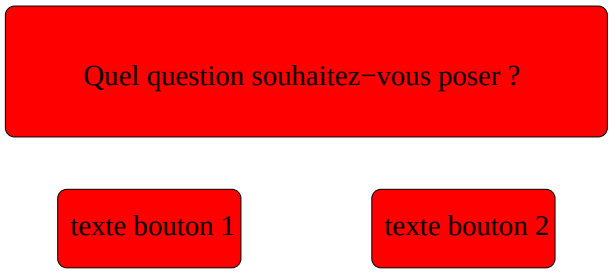


FIG. 13 – Un exemple de dialogue

devenait de plus en plus important et il n'était pas pratique de devoir recoder un composant en java à chaque nouvelle question. Pour cette raison, nous avons défini un composant "'Dialog" pour lequel, à partir du superviseur, on peut : écrire le texte de la question, choisir le nombre de réponses possibles et les messages renvoyés au superviseur pour chacune de ces réponses. Un tel composant d'interface peut être défini par le superviseur en envoyant le message suivant :

```
(MICROGUI ConfirmBool
(‘‘Quel question souhaitez-vous posez ?’’
(couleur de l’arrière plan de la microGUI)
(‘‘texte du bouton 1’’
(appui-bouton-1))
(‘‘texte du bouton 2’’
(appui-bouton-2))))
```

L'affichage au niveau de l'écran tactile sera alors semblable à la figure 13.

Dans notre cadre, la possibilité d'afficher ou non ce composant peut permettre d'effacer une question si le superviseur considère qu'elle est devenue inutile (soit sur la base de données perceptuelles, soit sur un timeout). Par exemple, si le robot perd le visiteur de vue et qu'il demande : "Êtes-vous là ? (oui non)", il peut effacer la question si la detection de visages lui indique qu'une personne est de nouveau là.

De plus, la possibilité de choisir la couleur d'arrière fond permet de mieux caractériser l'importance du message.

Chapitre 7

Rackham à la Cité de l'espace

7.1 Introduction

Nous présentons ici les travaux que nous avons effectués sur le robot Rackham[12] et plus particulièrement dans le cadre d'une expérimentation qui s'est déroulé à la Cité de L'Espace de Toulouse.

Dans un premier temps, nous présentons la démonstration et son déroulement. Ensuite, nous présentons le système de supervision que nous avions défini pour piloter cette démonstration. Nous analyserons ensuite les résultats de cette expérimentation avant de conclure.

7.2 Description de la démonstration

7.2.1 L'interface

Comme indiqué figure 14 le robot affichait de manière permanente :

- une carte avec un plan de l'exposition et un point sur cette carte montrait sa position mise à jour périodiquement.
- un retour visuel de la détection de visages effectuée par icu était donné aux visiteurs par l'intermédiaire de l'affichage du flux vidéo agrémenté d'un carré sur chacun des visages détectés (rq : seule la détection de visages était disponible).
- une carte montrant les éléments détectés par le laser du robot (i.e. la carte construite par le module `aspect`).

7.2.2 L'architecture fonctionnelle

L'architecture fonctionnelle utilisée pour Rackham est présentée figure 15. Avec cette architecture Rackham est capable :

- de se localiser dans l'environnement de manière géométrique (positionnement en x, y, théta dans un repère fixé au sein de l'environnement) et de manière topologique, un certain nombre de zones avaient été définies autour des points d'intérêts de l'exposition, l'environnement tel qu'il était modélisé est illustré figure4.
- de calculer une trajectoire (avec `VSTP`) et de la réaliser (avec `NDD`).
- de détecter les visages à l'aide de la caméra à l'aide du module `ICU`, la caméra étant orientable vers l'avant ou vers l'arrière selon les besoins.

[12] http ://www.laas.fr/robots/rackham/

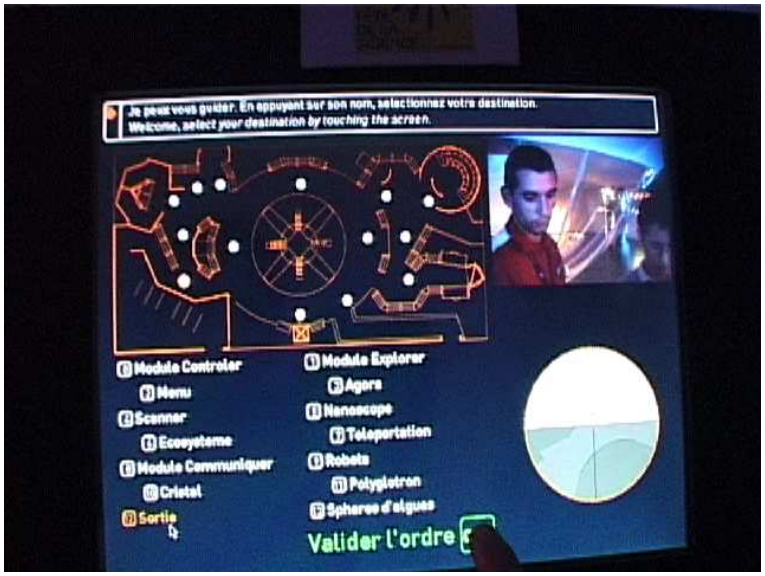


FIG. 14 – Interface montré par le robot

– de parler à l'aide du clone `clone`.
– d'effectuer de la reconnaissance vocale `reco`.

7.2.3 Scénario

Le plan de la démonstration comprenait deux grandes parties : tout d'abord le robot recherchait un visiteur avec lequel interagir : "*search interaction*" puis il se proposait de guider la personne rencontrée et réalisait la tâche si elle était d'accord : "*do guide*".

Lors de la phase de recherche d'interaction "*search interaction*", le robot circulait dans l'exposition. La détection d'un visiteur pouvait se faire de plusieurs façons :

– détection d'un visiteur devant le robot :
  – détection d'un visage avec la détection de visage du module `icu`, la caméra étant alors tournée vers l'avant,
  – détection d'un blocage du robot avec le module ndd (si le robot n'arrive plus à avancer le module ndd émet un signal de blocage).
– détection d'un visiteur à l'arrière du robot :
  – détection d'un appui sur l'écran tactile, un message invitant les gens à appuyer étant affiché,
  – détection d'une interpellation du robot grâce à la reconnaissance vocale (cependant il fallait que la personne soit proche du robot pour que cette modalité fonctionne).

Une fois la personne détectée, si besoin le robot se retournait vers le visiteur et dans tous les cas retournait la caméra en direction de l'écran tactile. A partir de ce moment, une vérification permanente de la présence de l'homme était lancée par l'intermédiaire de la détection de visages, un retour visuel de cette détection était donné aux visiteurs par l'intermédiaire de l'affichage du flux vidéo agrémenté d'un carré sur chacun des visages détectés.

Le robot se présentait alors : "Bonjour, je suis Rackham et je peux vous guider dans l'exposition". Et proposait ses services, au choix : "Cliquez sur la carte pour choisir une destination." ou "Dites-moi ou vous voulez aller." en affichant un plan de l'exposition.

Le visiteur choisissait alors sa destination, le robot demandait une confirmation. La présence

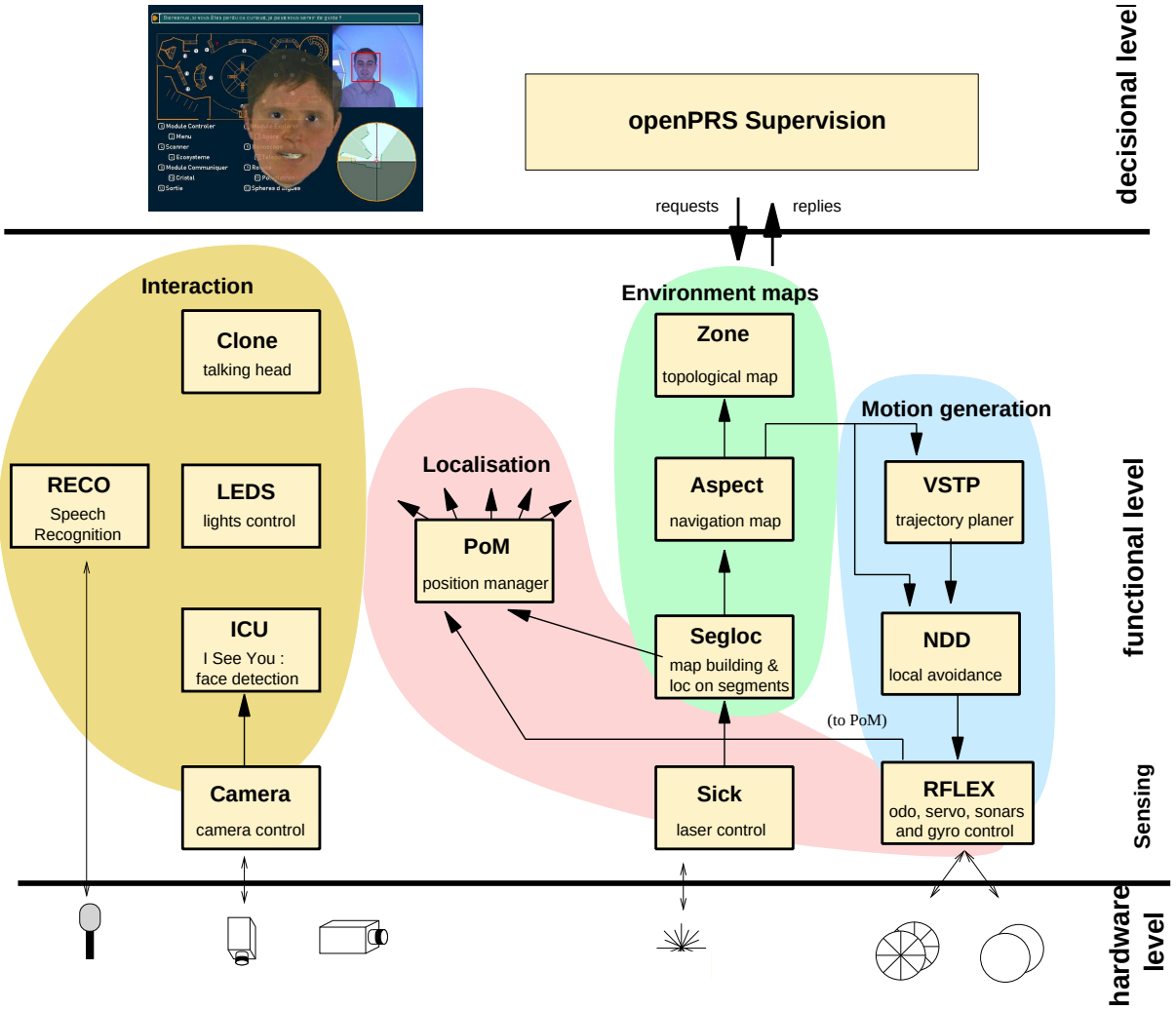


Fig. 15 – Architecture complète du robot Rackham

de cette confirmation vient de nos premières expériences où nous nous sommes aperçus que les visiteurs étaient un peu déroutés de voir partir le robot de suite après avoir fait leur choix. Cette confirmation nous permettait une meilleure compréhension et coordination avec les visiteurs.

Une fois la destination confirmée, le robot indiquait sur l'écran tactile ce qu'il réalisait : "je planifie le chemin" et ensuite affichait la trajectoire ainsi calculée. Le robot se tournait alors dans la direction vers laquelle il allait partir, disait "Suivez-moi" et commençait à réaliser sa trajectoire.

Au cours de la navigation, plusieurs cas pouvait se produire :

- perte de la localisation du robot : lorsque le robot se perdait, le robot s'arrêtait, expliquait qu'il tentait une relocalisation (en demandant si possible qu'on laisse l'espace libre devant lui). Si cette relocalisation fonctionnait, le robot redémarrait sa mission où il l'avait laissé. Si le robot ne pouvait se relocaliser lui-même, une carte de l'environnement plus précise d'affichait, montrant tous les segments perçus par le sick, on demandait alors au visiteur d'aider le robot à se relocaliser en proposant une position.
- blocage du module de navigation ndd : quand la foule était trop dense ou le passage trop étroit, le robot ne pouvait plus avancer et la navigation était arrêtée, le robot indiquait alors qu'il recalculait un nouveau chemin (prenant en compte les visiteurs présents devant lui), l'affichait et repartait en invitant le visiteur à le suivre de nouveau. Si les blocages étaient répétés ou si aucune autre trajectoire ne pouvait être trouvée, Rackham demandait qu'on lui laisse le passage : "Laissez-moi passer s'il vous plait". L'environnement étant assez sombre des leds avaient été placées sur le casque du robot. Ces leds clignotaient plus ou moins vite suivant l'espace libre calculé par le module `aspect`, ce qui permettait un retour visuel pour les visiteurs.
- demande d'arrêt à l'aide de l'interface ou de la reconnaissance vocale : le visiteur pouvait demander un arrêt du robot, cet arrêt pouvait être temporaire (i.e. suivi d'une reprise de la mission en cours) ou définitif (i.e. le robot partait à la recherche d'une nouvelle interaction),
- si la détection de visages perdait le visiteur au cours de la mission, le robot s'arrêtait et affichait le message : "où êtes vous ?", si la personne était de nouveau détectée (nous ne pouvions à l'époque pas savoir si la détection portait sur le même visiteur qu'au départ ou non, mais on supposait que tout visage rencontré derrière le robot était le visage du visiteur), Rackham indiquait "Vous revoilà !" et repartait. Si il n'y avait aucune detection pendant un temps défini, la mission était abandonnée.
- arrêt d'urgence : l'arrêt d'urgence était naturellement placé de manière visible sur le robot pour s'assurer qu'il soit accessible en cas de problèmes (un arrêt d'urgence radio était disponible pour la personne de la cité de l'espace surveillant le robot), mais il est alors sujet à tentation et son utilisation était plus intempestive qu'utile... cependant nous ne pouvons pas nous permettre de prendre un tel signal à la lègère. Lors d'un appui sur l'arrêt d'urgence, le robot s'arrêtait et indiquait que l'arrêt d'urgence avait été déclenché. Seul un nouvel appui sur l'interface tactile permettait le redémarrage du robot.
- d'autres arrêts plus critiques au niveau fonctionnel étaient possibles, nous avons eu notamment des problèmes de blocage du sick, ces erreurs n'offrant pas de possibilités de reprise sûr, le robot indiquait alors qu'il lui fallait l'aide de l'opérateur qui avait en charge de cesser la démonstration et de la relancer

Arrivé à destination, Rackham disait : "Nous sommes arrivés" et présentait l'élément de l'exposition en une phrase ou deux. Ensuite, il reproposait une interaction au visiteur qui décidait de recommencer ou non.

7.3 Description du système de supervision : l'avant Shary

Nous décrivons ici le système de supervision défini avant Shary qui a consisté en notre première implémentation de la théorie de l'intention jointe.

Dans ce système chaque homme entrant dans le champ du robot était pris en compte comme un agent avec qui le robot pouvait collaborer ou interagir. Pour être homogène, le robot lui-même était considéré comme un agent. Ainsi dans les explications nous utiliserons le terme agent si l'on se réfère de manière indifférente à un homme ou au robot, sinon le terme robot sera utilisé.

7.3.1 Espace d'état

L'espace d'état était composé de trois variables :
- l'engagement des agents dans la tâche (=AgentCo (exemple : RobotCo)) : engagé *committed*, non-engagé *uncommitted* (default),
- la croyance des agents concernant l'état de la tâche (=AgentTS (exemple :RobotTS)) : en cours *unachieved* (par défaut), terminé *achieved*, non-pertinent *irrelevant*, impossible *impossible*,
- progrès du robot dans la tâche (=RobotP) : inactif *idle*, commencement *start*, planification *planning*, en cours de réalisation *in task*, en cours d'arrêt *stopping*, arrêté *stopped*

7.3.2 Définition d'une tâche

En considérant la différence entre le cas individuel et le cas joint, et la granularité disponible entre les tâches et les sous-tâches, nous avons défini quatre types de tâches :

tâche racine *root task* correspond à l'implémentation actuel de l'agenda et est limité à une tâche unique,

tâche individuelle *individual task* une tâche qui peut être décomposée mais qui ne concerne que le robot,

tâche jointe *joint task* une tâche qui peut être décomposée mais qui va concerner le robot et un autre agent,

activité *activity* une tâche non-décomposable qui correspond à une routine de bas-niveau ou une activité atomique pour l'homme telle qu'il est possible de l'observer pour l'homme.

Une tâche est définie par :
- son plan courant : qui correspond à l'ensemble des activités et des sous-tâches que le robot doit exécuter pour réaliser la tâche, des exemples de plans sont donnés figure 16
- un ensemble de moniteurs qui vont suivre le progrès de la tâche et vérifier si la tâche est terminée *achieved*, non-pertinente *irrelevant* ou impossible *impossible*. Des exemples de moniteurs sont données figure 17.

Dans la version actuelle du système, les moniteurs sont passifs, i.e. un moniteur ne peut pas être la cause d'une action du robot. Ils utilisent juste les informations provenant du déroulement de la tâche pour conclure. Les moniteurs sont une manière de gérer la multi-modalité, en effet il est possible d'associer un nombre arbitraire de moniteurs à une valeur (à un état) donnée correspondant à différentes modalités.

L'affinage de la tâche est réalisé par l'intermédiaire d'une phase de replanification. Dans cette version, les plans sont stockés dans une bibliothèque de plans. La figure 18 illustre la structure dynamique de la tâche et le contexte de planification.

```
(=>(PLAN (ROOT)
        (.   (. Robot (INITIALISATION) .)
             (. Robot (LOOKING FOR INTERACTION)
             (. Robot Agent (DOSERVICE) .) .))
```

```
(=>(PLAN (GO_TO $destination)
    (.   (. Robot (FIND TRAJECTORY $destination) .)
         (. Robot (MOVE $destination).)
         .) .))
```

FIG. 16 – Exemples de plans de tâches stockés dans la bibliothèque de plans

```
(=>(MONITOR (GO TO $destination) $who1 $who1 ACHIEVED OK (GOTOACHIEVED)))

 ; watch BRAKESON
(=>(MONITOR (GO TO $destination) $who1 $who1 IRRELEVANT BRAKES (RFLEXTRACKSPEEDSTARTBRAKESON)))

 ; watch ASPECTFAILED
(=>(MONITOR (GO TO $destination) $who1 $who1 IMPOSSIBLE ASPECT_FAILED (ASPECTFROMPOSTERCONFIGFAILED)))

; watch people blocking the robot
; few time
(=>(MONITOR (GO TO $destination) $who1 $who1 IMPOSSIBLE PEOPLE (ATBLOCKED)))
(=>(MONITOR (GO TO $destination) $who1 $who1 ACHIEVED NEXT_TO (ATNEXTTO)))

; delay
(=>(MONITOR (GO TO $destination) $who1 $who1 IMPOSSIBLE DELAY (ELAPSED-TIME (TIME) 600)))

; segloc
(=>(MONITOR (GO TO $destination) $who1 $who1 IMPOSSIBLE LOCLOOP_FAILED (SEGLOCLOCLOOPFAILED)))
```

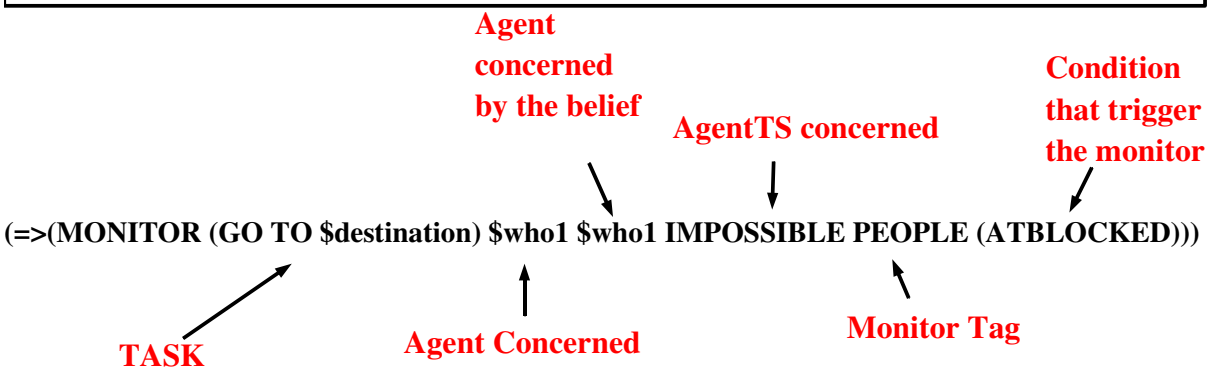


FIG. 17 – Syntaxe pour le codage des moniteurs associés à la tâche (GO TO $dest)

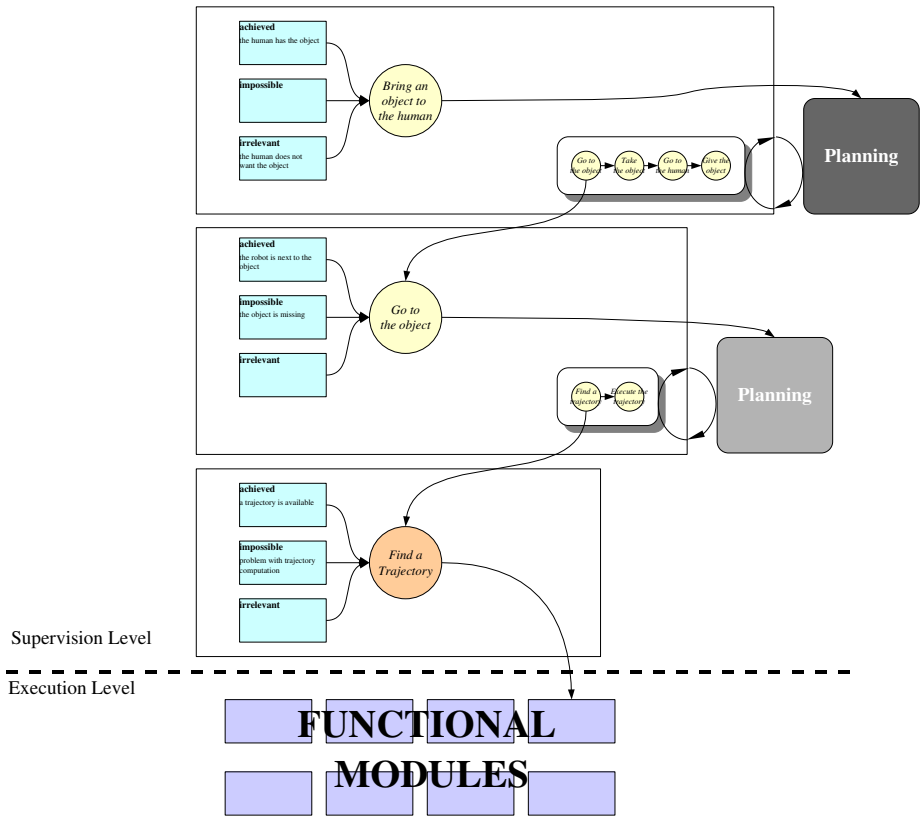


FIG. 18 – La structure hiérarchique de tâches gérée par le superviseur

L'activité de planification associée à une tâche est un processus continu, il vérifie de manière périodique la faisabilité de la tâche et fournit les prochaines tâches à réaliser en fonction du contexte courant. Les événements détectés par les moniteurs à un certain niveau sont propagés à travers la tâche et ses sous-tâches et occasionnent des créations/destructions de sous-tâches et des replanifications.

7.3.3 Tâches jointes

Chaque tâche jointe suit un schéma identique. Le plan d'une tâche jointe est divisée en 4 sous-tâches comme illustré figure 19 :

PRETASK correspondant à la phase d'établissement des engagements,

DOTASK correspondant à l'exécution de la tâche,

POSTTASK correspondant à la terminaison de la tâche,

CHECKTASK correspondant aux traitements des problèmes, potentiels pouvant survenir concernant la croyance mutuelle entre l'homme et le robot,

Chacune des ces sous-tâches peut être définie comme une tâche individuelle, jointe ou une activité.

Concernant les tâches jointes, plusieurs types de moniteurs peuvent être définis. La figure 17 montre la tâche individuelle (GO TO $dest) où les moniteurs sont définis pour $who1 concernant $who1, i.e. les moniteurs concernant les croyances du robot sur lui-même. Pour une tâche jointe,

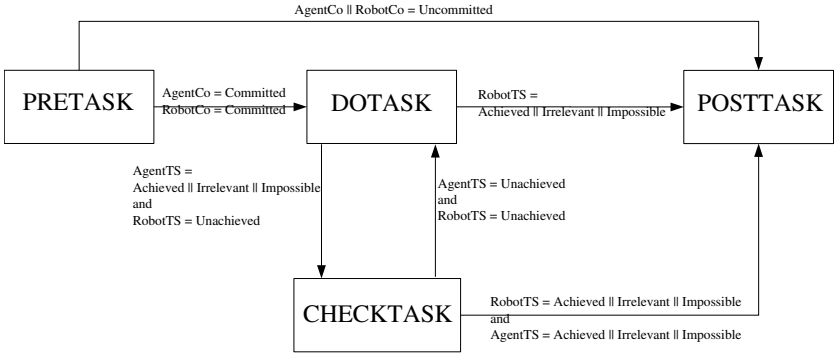


FIG. 19 – Une tâche jointe est divisé en 4 sous-tâches : (1) PRETASK : correspondant à la phase d'établissement des engagements, (2) DOTASK : correspondant à l'exécution de la tâche, (3) CHECKTASK : correspondant aux traitements des problèmes, potentiels pouvant survenir concernant la croyance mutuelle entre l'homme et le robot, (4) POSTTASK : correspondant à la terminaison de la tâche.

il est possible de définir des moniteurs concernant les croyances d'un agent vis à vis d'un autre agent.

7.3.4 Résumé et Leçons

Nous avons développé et implémenté un système de supervision permettant la gestion de tâches individuelles, jointe et d'activités. Chaque tâche est définie par un plan et des moniteurs associés. Un plan correspond à une séquence de sous-tâches et/ou d'activités. Les moniteurs servent à établir si une tâche est terminée *achieved*, impossible *impossible* ou stoppée *stopped*.

Le système peut être contrôlé à différents niveaux au même moment. Si un événement est détecté à un certain niveau, le système est capable de le prendre en compte à ce niveau et d'appliquer les solutions adaptées (éventuellement en propageant aux niveaux supérieurs ou inférieurs).

Le superviseur est écrit en OpenPRS. Les plans proviennent d'une bibliothèque de plans et seul le robot peut proposer une tâche.

Le système a été testé lors de la dernière visite de Rackham à la Cité de l'espace pour la réalisation du scénario présenté section 7.2.3 page 146.

Il constitue notre premiere tentative de construction d'un système basé sur la théorie de l'intention jointe. Il a montré des résultats encourageants, notamment sur la facilité de définir des tâches et des moniteurs, sur la réutilisation de morceaux de tâches déjà définis (ainsi le système a été utilisé au LAAS pour une tâche consistant pour Rackham à proposer ses services et à la conférence RO-MAN en Angleterre) et sur la souplesse qu'apportait l'encadrement de l'interaction (par exemple l'utilisation du CHECK_TASK lorsqu'un problème survenait). Cependant, le schéma d'interaction défini (i.e. la figure 19), nous est vite apparu trop restreint car il ne permettait pas de rajouter des étapes d'interaction, c'est pourquoi il nous est apparu important de rendre possible la définition de nouveau schémas en fonction de la tâche et/ou du contexte.

7.4 Résultats

7.4.1 Quantitatifs

Entre mars 2004 et novembre 2006, Rackham a effectué 9 visites à la "Cité de l'Espace"[13]. Au fil de ses visites, sur un total d'une centaine de jours, Rackham a été en interaction effective avec environ 3000 visiteurs pour 200 heures de fonctionnemment.

L'objectif[14] de nos premières visites étaient principalement la validation des algorithmes de localisation et de navigation, ainsi que leurs intégrations au niveau de la supervision. En effet, il s'avère très différent d'avoir des algorithmes qui marchent et des algorithmes qui fonctionnent pendant des heures de manière robuste. Le tableau suivant présente les résultats obtenus en février 2005

résumé visite du 7 au 20 février 2005 :
Les durées sont exprimées en heures et le nombre de requêtes représentent le nombre de requêtes du superviseur aux modules fonctionnels.

day	missions	distance	duration	requests	speed
1	40	395	1 :25	2801	0.51
2	49	555	1 :32	2719	0.56
3	44	487	1 :18	2557	0.62
4	82	851	3 :32	4338	0.44
5	82	881	2 :28	4209	0.58
6	70	739	1 :49	3609	0.56
7	85	884	2 :14	4338	0.50
8	71	815	2 :24	3984	0.53
9	55	663	1 :31	3154	0.60
10	78	912	2 :29	4742	0.49
11	71	872	2 :08	4214	0.54
12	91	994	2 :49	4632	0.53
	818	**9048 m**	**25 :39**	45297	**0.54**

Nous avons ensuite pu nous pencher sur l'interface graphique, puis la supervision, puis sur l'intégration de composants telle que la reconnaissance vocale pour arriver à la démonstration finale telle qu'elle est expliquée section 7.2.3.

Au cours de cette dernière visite, Rackham a effectué "un service minimum" puisque le personnel de la Cité n'était disponible qu'une heure et demi par jour, cependant sur les 4 jours, Rackham a effectué 143 missions en 3h38 (88 missions (61%) utilisant la reconnaissance vocale pour communiquer avec le robot dont 70 (49%) totalement) et 70 recherches d'interaction accomplies en 1h11, ce qui nous donne 75% mission 25% recherche. Ces résultats montrent une bonne robustesse de la reconnaissance vocale car l'environnement de la Cité de l'Espace était particulièrement bruyant (i.e. présence de bruit d'ambiance de vaisseau et bruit des autres éléments d'exposition)

[13] du 22/03/2004 au 09/04/2004, du 24/05/2004 au 04/06/2004, du 08/07/2004 au 21/07/2004, du 04/10/2004 au 17/10/2004, du 07/11/2004 au 21/11/2004, du 07/02/2005 au 21/02/2005, du 02/05/2005 au 16/05/2005, du 01/08/2005 au 16/08/2005, du 21/11/2005 au 27/11/2005.

[14] avant même cela, la première tâche que nous avons du réaliser consistait à permettre un démarrage aisé du robot pour permettre au personnel de la Cité de l'Espace de l'utiliser en dehors de notre présence, nous avons développé un système simple permettant de localiser le robot à l'aide de l'interface graphique.

7.4.2 Qualitatifs

La qualité de l'interaction obtenue est difficile à quantifier. Cependant, les expériences nous ont montré qu'il était important de maintenir un contact entre le visiteur et le robot, i.e. que le visiteur sache que le robot savait qu'il était là, par exemple en montrant un retour vidéo et/ou en montrant qu'un départ de l'homme est perçu. De plus, le robot doit donner de manière continu des informations sur ce qu'il est en train de faire, cela a à voir avec la lisibilité, il est important que l'homme comprenne ce que le robot est en train de faire pour pouvoir le suivre (ainsi si le robot s'arrête pour un problème quelconque : blocage de la navigation, perte de localisation, etc..., il doit expliquer et montrer que cela ne remet pas en cause l'ensemble de la mission).

7.5 Conclusion et travaux futurs

Cette expérimentation a constitué une première étape au sein du projet Cogniron et a constitué le coeur du projet Robea HR+. Elle a montré sa robustesse non-seulement en fonctionnant à la Cité de l'Espace ([CFA+05]) mais également en fonctionnant pour les visiteurs du laboratoire et même lors d'une présentation lors d'un workshop à Londres ([CFA+06]).

Ces travaux sur la fonction de guide continuent au laboratoire au sein du projet européen URUS pour le développement de plusieurs robots guides en environnement urbain (rues piétonnes).

Chapitre 8

Jido, un robot de service

8.1 Introduction

Nous présentons ici les travaux que nous avons effectués sur le robot Jido[15] et plus particulièrement dans le cadre de la tâche consistant à donner un objet à un homme.

Dans un premier temps, nous présentons la démonstration et son déroulement. Ensuite, nous expliquerons son implémentation au sein de Shary. Enfin, nous conclurons sur les travaux futurs.

8.2 Description de la démonstration

8.2.1 Présentation de l'architecture fonctionnelle

L'architecture fonctionnelle de Jido est présentée figure 20. Au sein de cette architecture nous allons utiliser les fonctionnalités permettant à Jido :

- de se localiser dans l'environnement de manière géométrique (positionnement en x, y, théta dans un repère fixé au sein de l'environnement).
- de calculer une trajectoire pour le bras avec `MHP` et de la réaliser avec `qarm` et `qarm`.
- de détecter un effort sur la pince (et donc une prise de l'objet par l'homme) grâce aux modules `force` et `fingers`, et d'ouvrir/fermer la pince grâce au module `fingers`.
- d'utiliser le module `ICU` pour, en se basant sur la détection de visages, savoir si l'homme regarde le robot ou non.
- de détecter les hommes présents dans l'environnement en se basant sur les données du laser et du module `ICU` grâce au module `humPos`
- de parler à l'aide du clone `clone` et de `speak`.

8.2.2 Scénario

Nous nous basons sur le scénario déjà défini précédemment. Le robot Jido doit donner l'objet qu'il porte à une personne qu'il aura détecté près de lui.

Jido attend d'abord l'arrivée d'une personne. Si la personnne ne s'arrête pas, Jido ne fait rien. Si la personne s'est arrêtée, le robot s'approche et propose l'objet : "Voulez-vous la bouteille ?". Si la personne reste devant le robot, Jido considère cela comme un accord et tend la bouteille à l'homme.

[15] http ://www.laas.fr/∼mransan/jido/

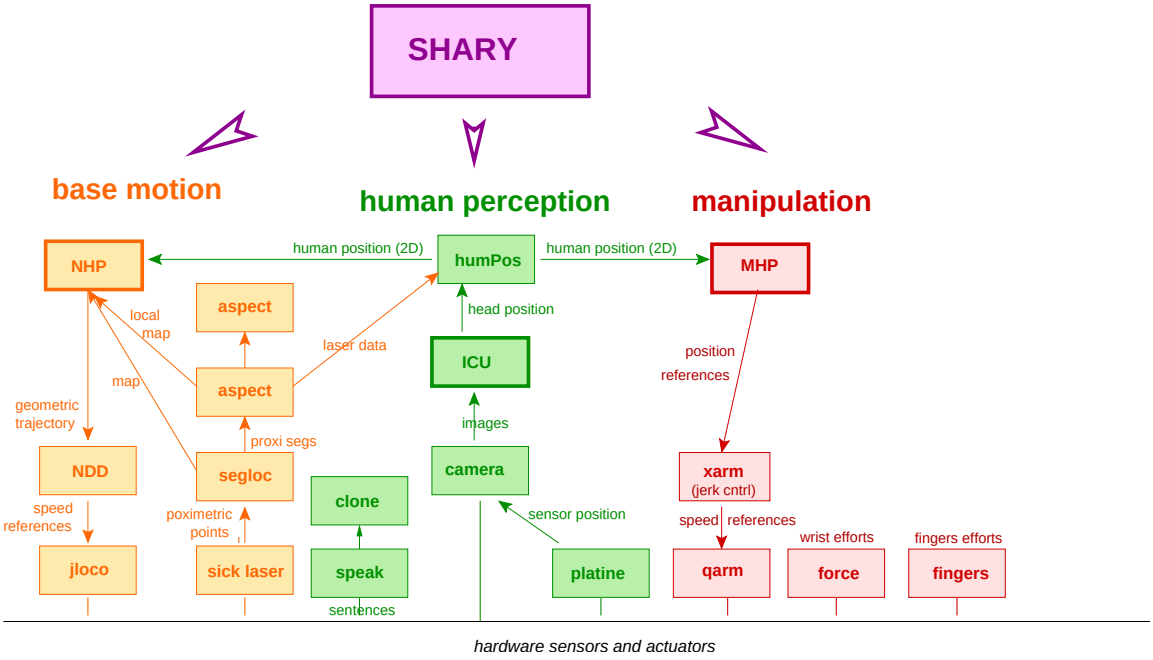


FIG. 20 – L'architecture logicielle de Jido

Pendant ce temps là le robot détecte si la personne est toujours là bien sûr mais aussi si elle regarde dans sa direction (vu qu'il va être amené à bouger le bras) grâce à la détection de visages. Si le robot détecte que l'homme ne regarde plus dans sa direction, il suspend la tâche et attend soit le retour de l'homme à la tâche, soit son départ (bien entendu un timeout permet de ne pas rester bloqué sur ces conditions).

Une fois le mouvement du bras terminé, le robot indique : "Vous pouvez prendre la bouteille."[16]

Si l'homme prend la bouteille, le robot le détecte grâce à ses capteurs, il indique : "c'est bon je vais lâcher" et lâche effectivement la bouteille. Ensuite, il remet le bras à sa position initiale, dit "au revoir" et s'en va.

Si l'homme ne prend pas la bouteille, le robot comprend cela comme un possible d'abandon, il demande une confirmation : "Vous ne voulez pas la bouteille ?", si l'homme persiste à ne pas prendre la bouteille, le robot abandonne la tâche, sinon on reprend la tâche.

Une des variations du scénario se base sur la reconnaissance de visages. On demande à Jido de donner l'objet à une personne en particulier. Si la personne détectée n'est pas la bonne personne, Jido ne donne pas l'objet.

8.3 Utilisation de Shary

En ce qui concerne cette tâche en particulier et Shary en général :

- dans le chapitre consacré à Shary page 87, nous avons étudié la définition de cette tâche à l'aide de Shary,
- dans une des sections du chapitre consacré à la modélisation de la communication page 114, nous avons étudié le lien possible avec un système de dialogue,

[16] il n'est pas encore possible de détecter que l'homme prend la bouteille alors que le bras est en mouvement

– dans cette section, nous allons maintenant montrer le lien entre les données des modules, i.e. la partie fonctionnelle de l'architecture, et la base de faits et plus particulièrement la base d'agents et les moniteurs.

Nous considérons un modèle d'agent simple, notre agent aura 5 paramètres :
– *name* : le nom de l'agent considéré
– *category* : la catégorie de l'agent considéré : homme ou robot
– *position* : la position de l'agent vis à vis du robot : inconnu ou proche, mi-distance, éloigné
– *interaction* : est-il possible pour le robot d'interagir avec l'agent ? i.e. est ce que l'agent est proche du robot et est ce qu'il regarde le robot ?
– *object owned* : objet possédé par l'agent : l'agent proche du robot lui a pris un objet

Une base de données d'agents fait partie des données a priori présentes dans la base de faits. Cette base est ensuite mise à jour en fonction de l'arrivée des données des modules.

Le schéma 21 montre la manière dont peuvent être reliées les données des modules à ce modèle d'agent. Nous voyons sur ce schéma que la définition d'un paramètre va souvent être dépendante de données de plusieurs modules, il est donc nécessaire de disposer d'un composant qui d'une part va interpréter les données (e.g. qui à partir des positions de l'agent et du robot calcule la distance entre les deux), et d'autre part les combiner (e.g. pas de detection de visages mais des jambes alors il y a quelqu'un qui ne me regarde pas devant moi).

La figure 22 relie ensuite la mise à jour de ces données au déclenchement des moniteurs. Nous voyons qu'il est plus aisé de gérer des données déjà interprétées plutôt que les données brutes des modules. De plus, cela permet une meilleure réutilisation du code, notamment si des changements interviennent au niveau des modules.

8.4 Travaux futurs

8.4.1 Shary

Nous voyons que le système permet de prendre en compte une tâche et ses possibles variations.

Après avoir porté notre travail sur la définition et l'implémentation du système d'exécution et de supervision de Shary. Nous allons maintenant porter notre étude sur la définition de modèle d'agents plus conséquents. En effet, comme nous l'avons déjà dit, c'est la nature et la pertinence des données présentes dans la base de faits en générale et la base d'agents en particulier qui va maintenant nous permettre d'étendre les capacités du système : lien avec la planification, avec le dialogue, meilleure adaptation des moniteurs et des schémas de communication, etc.

8.4.2 Démonstration

La démonstration décrite ici n'est qu'une partie d'une expérimentation développée dans le cadre du projet européen cogniron (http ://www.cogniron.org).

Le scénario consiste à détecter les éventuels besoin d'un homme assis dans la pièce où se trouve le robot. Le robot détectera par exemple que l'homme a besoin de boire. Il ira lui demander confirmation et si le besoin est avéré, le robot ramènera une bouteille à l'homme. Au sein de ce scénario, nous nous intéresserons plus précisément à la tâche consistant à donner un objet à l'homme.

L'intégration de cette démonstration se fait au fur et à mesure de l'arrivée des composants fonctionnels, mais Shary a été développé pour permettre la gestion de l'ensemble de cette tâche qui sera testé dans différents contexte.

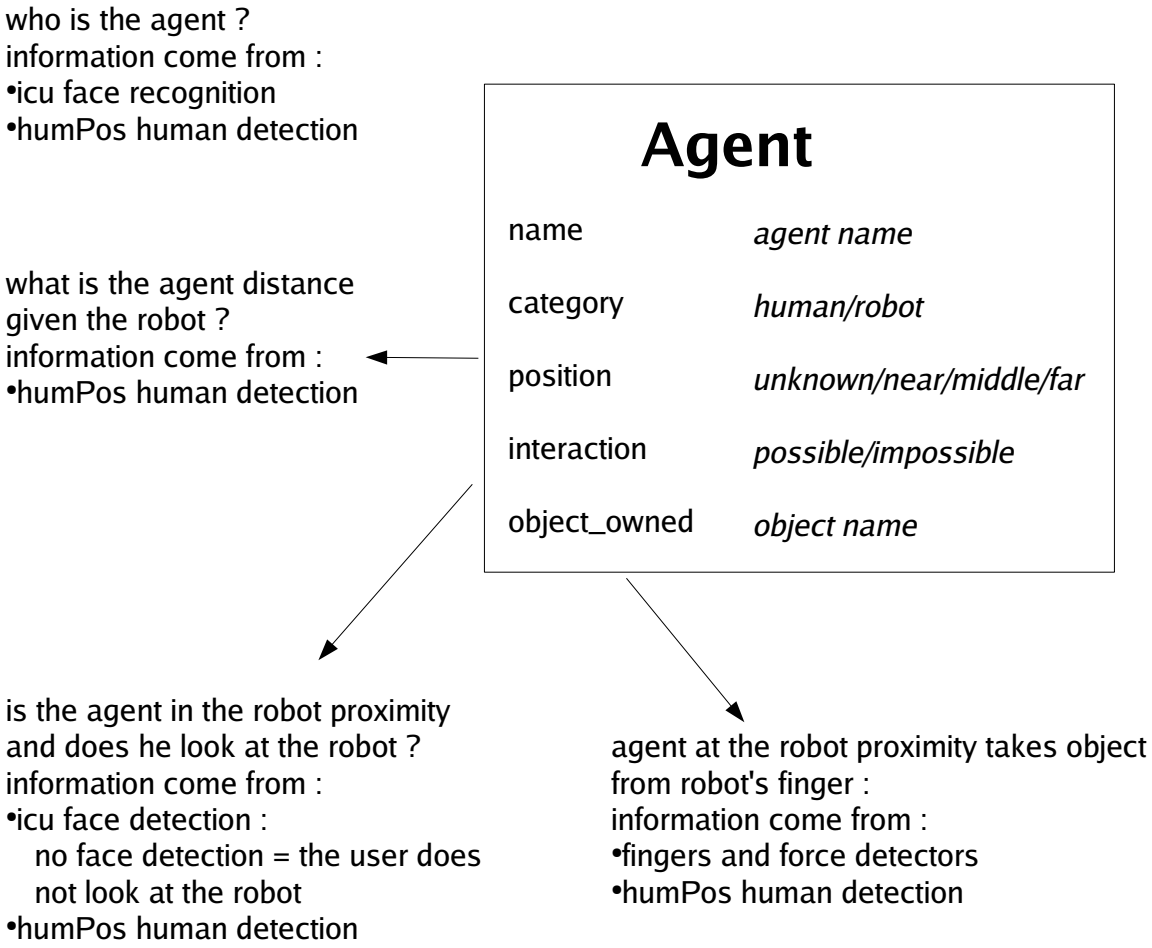


FIG. 21 – Exemple de modélisation d'un agent

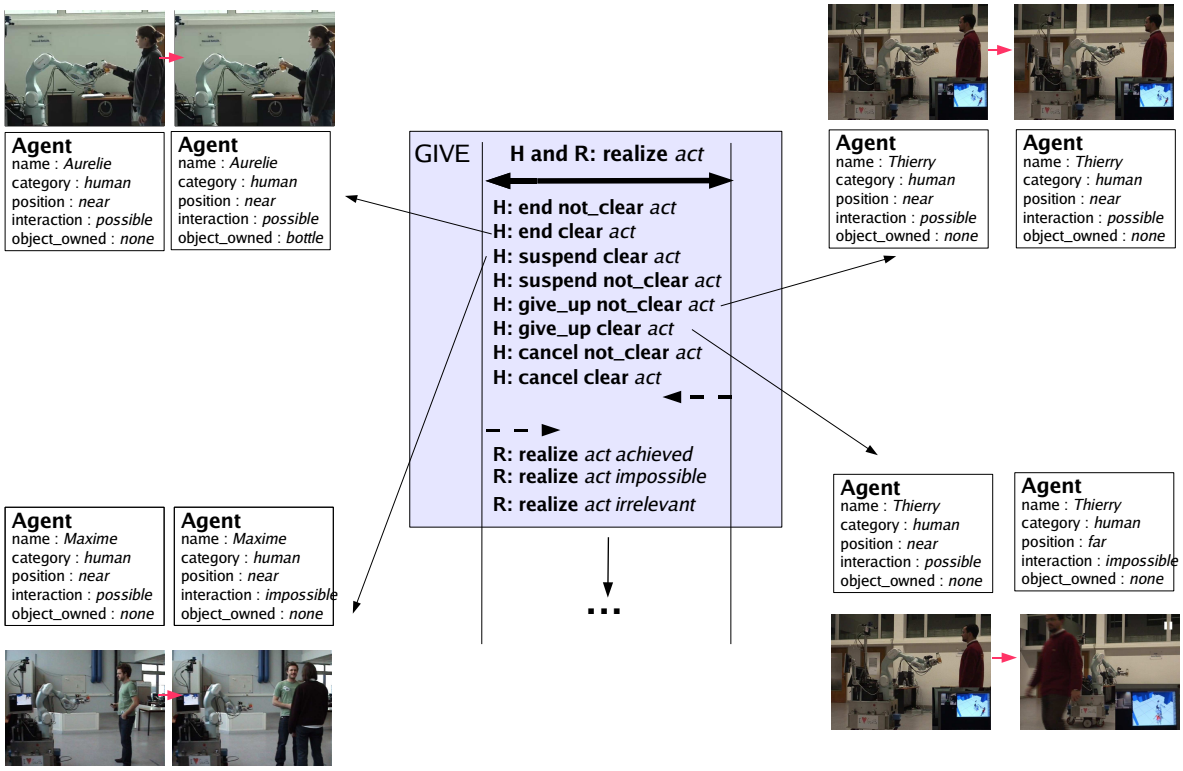


FIG. 22 – Description des changements d'états induisant le déclenchement de moniteurs. Les vidéos sont disponibles à cette adresse : http ://www.laas.fr/~mransan/jido

Conclusion Générale

Nous avons élaboré une architecture destinée à l'interaction homme-robot et permettant la réalisation de tâches collaboratives par la prise en compte de l'homme à tous les niveaux de fonctionnement et de raisonnement du robot. Cette architecture définit un cadre général regroupant des modèles d'agents, de tâches et de communications et les moyens de gestions de ceux-ci notamment par le recours à des resssources de planification d'actions et de communications.

Ce cadre découle en grande partie d'expérimentations que nous avons réalisées avec le robot Rackham à la Cité de l'Espace de Toulouse et de notre étude bibliographique.

Il définit :

- un agenda qui a la connaissance des buts et des contraintes de haut-niveau et doit gérer l'ensemble des tâches du robot,
- des ressources :
  - en terme de planification prenant explicitement en compte l'homme : HATP pour les tâches, HAMP pour le mouvement,
  - en terme de schémas de communication définissant des mécanismes de "turn-taking" lors de l'exécution de tâches,
- une base de faits regroupant d'une part les informations sur le monde et d'autre part les informations disponibles sur les collaborateurs du robot (IAA), ces informations sont rendues disponibles par l'intermédiaire des fonctions perceptives et décisionnelles du robot,
- un système de supervision et d'exécution de tâches : Shary qui constitue le noyau décisionnel du système.

Au sein de ce cadre, nous avons développé le système de supervision et d'exécution de tâches : Shary permettant la gestion d'une collaboration définie de manière adaptative au cours de la réalisation de tâches jointes.

Shary définit des schémas de communication, indépendants des tâches et des recettes exprimant le déroulement des tâches ou plutôt des actes de communication/réalisation pouvant exister au sein de ces tâches. Ce déroulement s'exprime sous la forme d'un affinement en sous-tâches, la communication inhérente à une tâche est déterminée dynamiquement lorsque la tâche doit être réalisée en fonction du contexte courant, du collaborateur considéré et de la tâche elle-même.

Nous avons étudié dans quelle mesure nous pouvions relier ces schémas de communicaiton aux notions de croyance mutuelle et de "*grounding*" issues de la littérature des théories de la coopération. Cela nous a amené à définir une notion restreinte de la croyance mutuelle, en considérant que toutes les croyances disponibles l'étaient par l'intermédiaire de la perception et des mécanismes de décision du robot. A partir de là, nous avons élaboré deux mécanismes de base de "*turn-taking*" pour un acte de communication basé sur un ensemble d'étapes. Enfin, nous avons montré comment la communication ainsi définie pouvait être reliée à un système de dialogue.

Nous avons implémenté Shary sur le robot Jido. L'implémentation et la définition même du système en font un système modulaire permettant la réutilisation aisée de tâches déjà définies

ainsi que de nouveaux schémas de communication.

Shary constitue aujourd'hui une première étape et une base pour d'autres développements.

Nous avons vu que la définition d'une base de faits plus "fine" en terme de représentation de situations, de connaissances partagées et de définition des agents pouvait permettre de renforcer les capacités du système. La prise en compte de la multiplicité des croyances, même si elle semble ardue, paraît également une piste intéressante.

L'Agenda est le composant sur lequel nous avons le moins travaillé mais son développement devrait être le pilier permettant la gestion de la prise d'initiative par le robot et une meilleure prise en compte des demandes provenant de l'homme.

Nous avons également vu que nous faisions une hypothèse forte au niveau des recettes sur le couplage entre réalisation et monitoring. Il serait intéressant d'étudier au niveau de la planification dans quelle mesure les nécessités de monitoring en terme de ressources, de contraintes sur la tâche et de contraintes spatio-temporelles influent sur la réalisation de la tâche elle-même.

Enfin, des travaux sont en cours comme nous l'avons vu sur le couplage avec un système de dialogue d'une part et avec le système de planification HATP de l'autre.

Annexes

Annexe A

Définitions

Nous rappellons ici toutes les définitions utilisées dans le manuscrit. Les références dans le texte peuvent être trouvé à l'aide de l'index page B.5.

$$
\begin{array}{rcl}
(BEL\ x\ p) & \triangleq & x\ croit\ p \\
(GOAL\ x\ p) & \triangleq & x\ a\ p\ comme\ goal \\
\Diamond p & \triangleq & p\ va\ devenir\ vrai\ finalement \\
 & \triangleq & \exists e\ (HAPPENS e; p?) \\
\Box p & \triangleq & p\ sera\ toujours\ vrai \\
 & \triangleq & \neg\Diamond\neg p \\
(HAPPENED\ a) & \triangleq & a\ vient\ juste\ de\ se\ produire \\
(HAPPENING\ a) & \triangleq & a\ est\ en\ train\ de\ se\ PRODUIRE \\
(HAPPENS\ a) & \triangleq & a\ va\ se\ produire\ ensuite \\
a; b & \triangleq & composition\ d'actions \\
a|b & \triangleq & choix\ non-dterministe \\
a||b & \triangleq & occurence\ simultan\ de\ a\ et\ de\ b \\
p? & \triangleq & test\ action \\
a* & \triangleq & rptition \\
(AGT\ x_1...x_n\ e) & \triangleq & x_1...x_n\ sont\ les\ seuls\ agents\ impliqus\ dans\ la\ squence\ d'vnements\ e \\
(DONE\ x_1...x_n\ a) & \triangleq & (HAPPENED\ a)\ \wedge\ (AGT\ x_1...x_n\ a) \\
(DOING\ x_1...x_n\ a) & \triangleq & (HAPPENING\ a)\ \wedge\ (AGT\ x_1...x_n\ a) \\
(DOES\ x_1...x_n\ a) & \triangleq & (HAPPENS\ a)\ \wedge\ (AGT\ x_1...x_n\ a) \\
(UNTIL\ p\ q) & \triangleq & \forall c\ (HAPPENS\ c; \neg q?)\ \supset\ \exists a\ (a\ \leq\ c)\ \wedge\ (HAPPENS\ a; p?) \\
 & \triangleq & Jusqu'\ ce\ que\ p\ soit\ vrai,\ q\ restera\ vrai. \\
(BMB\ x\ y\ p) & \triangleq & (BEL\ x\ p\ \wedge\ (BMB\ y\ x\ p)) \\
(MB\ x\ y\ p) & \triangleq & (BMB\ x\ y\ p)\ \wedge\ (BMB\ y\ x\ p) \\
(MK\ x\ y\ p) & \triangleq & p\ \wedge\ (MB\ x\ y\ p) \\
(MG\ x\ y\ p) & \triangleq & (MB\ x\ y\ (GOAL\ x\ \Diamond\ p)\ \wedge\ (GOAL\ y\ \Diamond\ p)) \\
(WMG\ x\ y\ p) & \triangleq & (MB\ x\ y\ (WAG\ x\ y\ \Diamond p)\ \wedge\ (WAG\ y\ x\ \Diamond\ q))
\end{array}
$$

$$
\begin{aligned}
(WAG\ x\ y\ p\ q) \triangleq\ & [\neg(BEL\ x\ p)\ \wedge\ (GOAL\ x\ \Diamond p)]\ \vee \\
& [(BEL\ x\ p)\ \wedge\ (GOAL\ x\ \Diamond(MB\ x\ y\ p))]\ \vee \\
& [(BEL\ x\ \Box\neg p)\ \wedge\ (GOAL\ x\ \Diamond(MB\ x\ y\ \Box\neg p))]\ \vee \\
& [(BEL\ x\ \neg q)\ \wedge\ (GOAL\ x\ \Diamond(MB\ x\ y\ \neg\ q))] \\
(JPG\ x\ y\ q) \triangleq\ & (MB\ x\ y\ \neg p)\ \wedge\ (MG\ x\ y\ p)\ \wedge \\
& (UNTIL\ [(MB\ x\ y\ p)\ \vee\ (MB\ x\ y\ \Box\neg p)\ \vee\ (MB\ x\ y\ \neg q)]\ (WMG\ x\ y\ p\ q)) \\
(SINCERE\ x\ y\ p) \triangleq\ & \forall e\ (GOAL\ x(HAPPENS\ e; (BEl\ y\ p)?)) \\
& \supset\ (GOAL\ x(HAPPENS\ e; (KNOW\ y\ p)?))
\end{aligned}
$$

Annexe B

Interaction entre le dialogue et la décision pour la réalisation de tâches collaboratives

Nous allons ici nous concentrer sur l'étude de l'interaction entre le dialogue et la supervision et la planification. Le but de ce travail est d'établir un mécanisme pour permettre une coordination et un turn-taking flexible au cours des phases de négotiations entre les hommes et le robot. Cela peut se produire à plusieurs niveaux de la hiérarchie de tâches. Pour cela nous allons étudier un scénario, ses possibles variations et le comportement du système au vu de ces variations.

Dans cette étude nous ne donnons pas a priori des réponses, mais des pistes à étudier et des propositions.

B.1 Actes de communication

Les actes de communication utilisé sont ceux précédemment défini : ASK_TASK, PROPOSE_PLAN, MODIFY_PLAN, GIVE_UP, CANCEL, END, REALIZE_TASK.

Le système de dialogue va être chargé de la négociation avec l'utilisateur jusqu'à l'obtention d'une réponse claire telle que défini section 5.2.2 page 111. Ainsi l'on considère que le système de supervision, lorsqu'une réponse proviendra du dialogue, aura toujours accès à une réponse claire de l'utilisateur. La supervision et le dialogue vont interagir à propos de la tâche et des tâches jointes, et les tâches participatives, mais pas les tâches individuelles autonomes.

B.2 Scenario

B.2.1 Scenario basique

Le scenario est le suivant. Un homme demande au robot de lui servir à boire. Jido attend dans une pièce. Luis arrive et demande à Jido de lui ramener un coca. Jido accepte. Il se rend à la cafeteria, trouve le coca et le ramène au bureau de Luis. Jido donne alors le coca à Luis.

B.2.2 Variation 1

Exemple 1 Jido rencontre Luis à la cafétéria. Avant qu'il prenne la canette de coca, le robot propose à Luis de la prendre lui-même. Deux possibilités : Luis accepte ou refuse.

Exemple 2 Jido rencontre Luis sur le chemin du retour (i.e. après avoir pris la canette de coca). Jido propose à Luis de lui donner la canette tout de suite. Deux possibilités : Luis accepte ou refuse.

Exemple 3 Jido va à la cafétéria mais il n'y a plus de coca, mais Jido connait une autre position possible pour la canette (et Luis n'a pas spécifié d'où devait provenir le coca).

Dans chacun de ces cas, un nouveau plan est construit au cours de l'exécution. L'information provient du système de planification et le robot propose la modification à l'utilisateur.

Nous considérons que les exemples 1 et 2 entrainent un changement dans la partie "public" du plan (comme expliqué section 4.1.1 page 87), c'est à dire que la modification s'effectue sur une partie du plan qui doit être partagé avec l'homme, dont la modification doit passer par une validation de l'homme. L'exemple 3 au contraire, porte sur une partie "privée" du plan dans le sens où Luis n'a pas expréssement indiqué d'où devait venir le coca (si cela avait été le cas, la modification aurait demandé une validation).

Le système de dialogue va recevoir un message concernant une demande de modification du plan et générera le message approprié.

Cependant, même si un nouveau plan a été calculé, nous devons nous poser la question du dérangement causé à l'homme. Doit-on considérer qu'il faut obligatoirement une validation de l'homme ou simplement lorsque l'homme est à proximité. On doit envisager de modéliser quand le robot sait que l'homme peut être dérangé.

B.2.3 Variation 2

Exemple 1 U : "Amène moi un café de la cafeteria."

Exemple 2 U : "Amène moi un café dans mon bureau."

Dans ce cas, nous allons considérer que l'utilisateur ajoute des contraintes à la tâche. Cela semble être une hypothèse raisonnable que certains objets puissent être à plusieurs positions. Si des contraintes peuvent être associées à la tâche, nous devons ajouté ces contraintes à la tâche et donc définir un langage de contraintes. Les contraintes pourront être défini de manière indépendante :

TACHE + CONTRAINTES

Les contraintes vont portées sur l'instanciation de certaines variables (e.g. le nom/la localisation d'une personne, d'un objet). L'ensemble des tâches et des contraintes qui peuvent leur être associées doit être rendu disponible. Une autre question porte sur le système de dialogue : est-il possible pour le système de dialogue d'analyser des phrases du type :

Tâche : Apporter(Café,Luis, contraintes(Café.localisation = Cafeteria))

B.2.4 Variation 3

Exemple 1 U : "Amène moi un café. Pose le sur mon bureau."

Exemple 2 U : "Amène moi un café à mon bureau. Oh, mais pose le sur mon bureau, pas sur ma chaise."

Exemple 3 U : "Amène moi un café. Donne le moi."

Exemple 4 Jido arrive dans le bureau, Luis dit "pose le là s'il te plaît".

Ici l'utilisateur impose une partie du plan. Comment le système de dialogue peut savoir que ce ne sont pas deux tâches, mais une seule et une proposition de plan (ou de modification de celui-ci) ? Cela ne peut pas être désambiguïsé si ce n'est pas lié à l'activité jointe.

Il nous faut étudier si les utilisateurs vont préférentiellement

1. donner deux tâches en séquences (e.g. : "amène moi un café et ensuite arrose la plante")
2. ou une tâche et une modification de plan (e.g. : "apporte moi un café et pose le sur mon bureau")

Si cela n'est pas possible de désambiguïsé pour le dialogue, il faut étudier un moyen de le gérer au niveau de la supervision. Le dialogue pouvant dans tous les cas renvoyer deux ASK_TASK consécutifs qui devront être géré de manière adéquat par la supervision.

Une autre possibilité serait de rendre disponible la décomposition de la tâche "apporter" pour détecter que "poser" doit être interprété comme une sous-tâche.

Un autre problème pour le dialogue est la référence au "le" (dans "pose **le** là" ou "donne **le** moi") en référence au café, ou même le "mon" (dans "**mon** bureau").

B.2.5 Variation 4

Exemple U : "si je ne suis pas dans mon bureau, pose le sur la table"

Nous avons ici une modification conditionnelle du plan. Est-il possible pour le dialogue de comprendre l'aspect conditionnel de la demande ? De plus au niveau de la planification, il semble difficile de prendre ce type d'information en compte. Nous laissons ce problème (intéressant) ouvert.

B.2.6 Variation 5

Exemple Pas de plan trouvé.

Le robot peut dire qu'il ne trouve pas de plan, mais ne sait pas dire pourquoi. En effet, les systèmes de planification actuel ne sont pas capables dans le cas général de donner des explications à l'échec de la planification.

B.2.7 Variation 6

Exemple "Apporte moi un café. Je serai dans mon bureau."

Dans ce cas, l'utilisateur donne une information qui sera vraie plus tard. Cela ne semble pas simple pour la planification de modéliser et de prendre en compte ce type de contraintes. Cependant, HATP aura la possibilité de construire des plans qui se synchronize sur des événements attendus (planification temporelle).

Pour le moment, pour que cette information soit comprise au niveau du dialogue, il faut qu'elle soit scindée en deux phrases.

B.2.8 Variation 7

Exemple "apporte moi un café... ah et apporte moi un mars aussi"

Cela pourrait être interprété comme une conjonction de buts, ou comme deux tâches qui n'ont pas forcément à être résolu en séquence.

B.2.9 Variation 8

Théoriquement, si un planificateur est appelé, cela peut aider à résoudre des actes de parole indirecte (*indirect speech acts*) comme "Oh, il fait froid ici ! c'est normal la fenêtre est ouverte."

Ainsi, le composant de dialogue peut rentrer deux informations/faits dans la base : (1) l'utilisateur pense qu'il fait froid, (2) la fenêtre est ouverte. Un planificateur pourrait déduire de cela qu'il serait plus confortable que la fenêtre soit fermée. Il pourrait alors calculer un plan pour fermer la fenêtre et envoyé cette suggestion au système de dialogue. Le dialogue pourrait alors réagir en faisant dire au robot : "Dois-je fermer la fenêtre ?"

B.2.10 Variation 9

Exemple 1 "Peux-tu réaliser ceci, avec ces paramètres ?"

Exemple 2 "Que peux-tu faire ?"

Cette variation illustre la différence entre les questions à propos des capacités générales du robot et les questions reliées à sa capacité de répondre si il est capable de réaliser une tâche dans un contexte donné.

B.3 Interaction entre le dialogue et la supervision

Nous avons défini un premier protocole d'interaction et d'échange de données entre le dialogue et la supervision. Le planificateur va intervenir ici comme une ressource appelée par le superviseur.

Les exemples montrés sont tirés des variations étudiées précédemment, plus spécialement le scénario de base et la variation 1.

Quatre étapes ont été analysées :

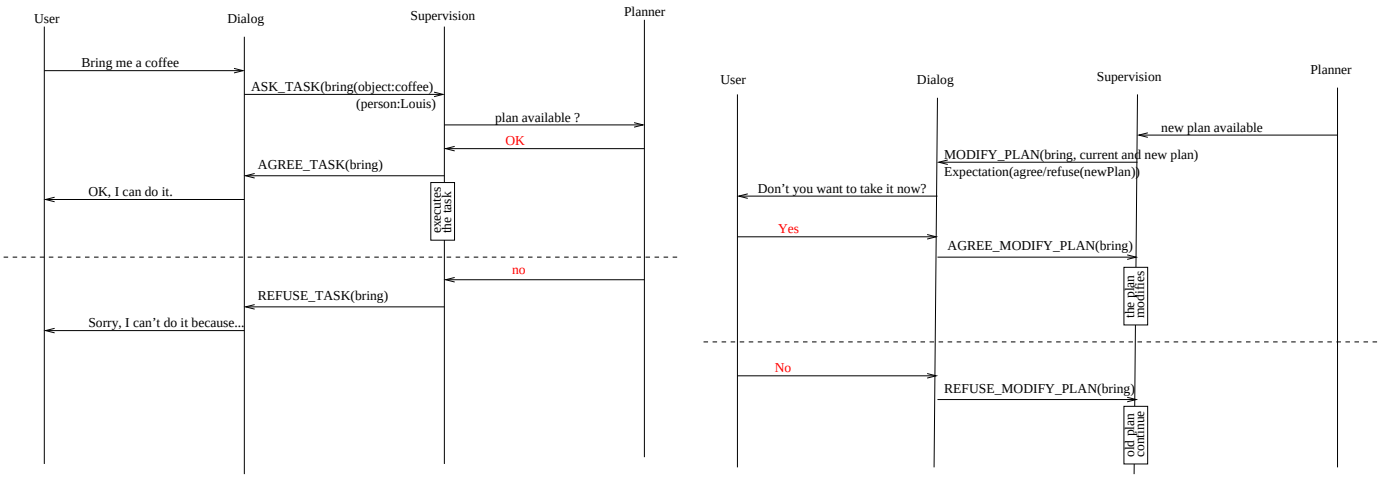
- START : le début de l'interaction (figure 1)
- TAKE : proposition faite à l'homme de prendre l'objet (figure 1)
- GIVE : échange du café (figure 2)
- TASK DONE : terminaison de l'interaction (figure 2)

B.4 Interface entre le dialogue et la supervision

Une définition préliminaire des structures qui seront échangées entre le dialogue et la supervision est proposée.

B.4.1 De la supervision vers le dialogue

```
<MSG>
<sender>Supervision</sender>
<receiver>DLG</receiver>
<commAct state = "launch/stop">
 <act-name>the communicative acts above</act>
 <act-type>agree or refuse or act</property>
 <info>
   <task id =
   "the ID of a task (specified by the user) or
```



(a) START (b) TAKE

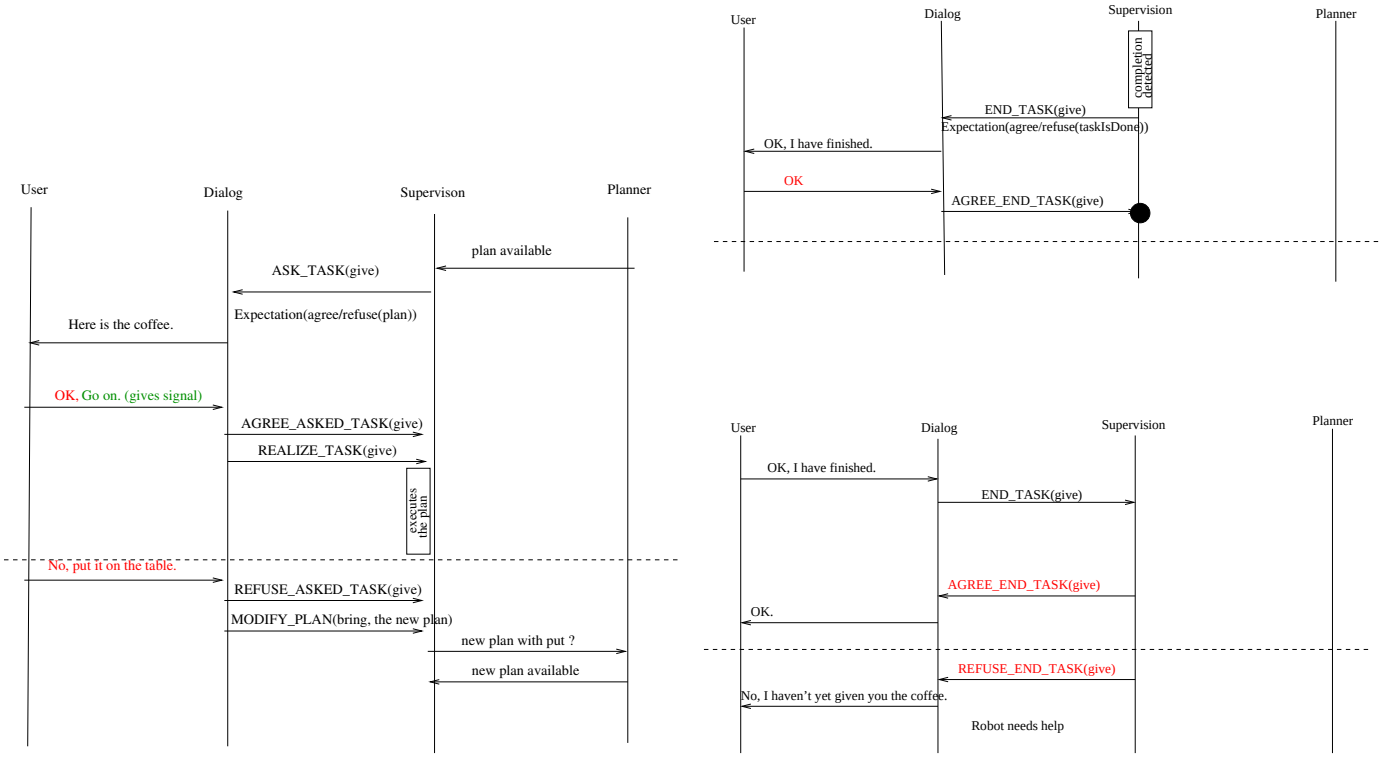
FIG. 1 – START : l'homme demande quelque chose au robot, le robot accepte ou refuse selon sa disponibilité ou sa capacité à réaliser la tâche. TAKE : le robot rencontre l'utilisateur avant d'atteindre l'endroit où il doit amener le café, il propose alors à l'homme de lui donner la café maintenant. Ceci montre une situation où c'est le robot qui va générer l'acte de communication et proposer une nouvelle manière de réaliser la tâche.

```
    a subtask (specified by the planner)">
   <person>...</person>
   <obj>...</obj>
   ...
   </task>
   <plan feature = "new or current">
    the name of the plan
   </plan>
  ...
 </info>
</commAct>
<expectation>
 <commAct> as defined above</commAct>
 ...
</expectation>
</MSG>
```

avec : `<commAct state = "launch/stop">` :
- launch : la supervision lance l'acte de communication,
- stop : la supervision stoppe cet acte de communication.

B.4.2 Du dialogue vers la supervision

```
<MSG>
<sender>DLG</sender>
<receiver>Supervision</receiver>
<commAct state = "final/negotiating">
```



(a) GIVE (b) TASK DONE

FIG. 2 – GIVE : le robot tend le café à l'utilisateur. Dans le premier cas, l'utilisateur accepte et l'échange à lieu ; Dans le deuxième cas, l'homme rejete le plan et propose une solution alternative : il peut poser le café sur la table. TASK DONE : dans le premier cas, le robot informe l'utilisateur que la tâche est terminée. Dans le deuxième cas, c'est l'utilisateur qui indique qu'il a terminé, le robot est en accord ou pas avec cette information.

```
<act>the communicative acts above</act>
<property>agree or refuse or act</property>
<info>
 <task>
 <person>...</person>
 <obj>...</obj>
 <position>the concrete position or
          gestureExpected
 </position>
 ...
 </task>
 <plan feature = "new or current">
 the name of the plan
 </plan>
 ...
</info>
</commAct>
</MSG>
```

avec : `<commAct state = "final/negotiating">` :
- final : le résultat final du dialogue,
- negotiating : le dialogue est en cours de négotiation avecx l'utilisateur, le commAct peut ne pas contenir toutes les informations

`<act-name>the communicative acts above</act-name>` : if the dialog knows the commAct, then the value of "act" will be set to. `<act-type>agree or refuse or act</act-type>` : an attribute act is the content that is refused or agreed. `<position>the concrete position or gestureExpected</position>` : "gestureExpected" est envoyé à la supervision si le système de reconnaissance indique qu'un geste est attendu, e.g. "ici" ou "là". Après avoir reçu cette demande, la supervision va enclenché la reconnaissance de geste dont le resultat sera fourni dans une base de données. Le dialogue accédera directement à cette base de données. Tant qu'il n'y a pas de résultat, le dialogue continuera à négocier.

B.5 Conclusion

Cette interface est encore en cours de développement cependant nous montrons ici que le système défini, Shary, peut s'interfacer avec le dialogue.

Bibliographie

[ACF+98] Rachid Alami, Raja Chatila, Sara Fleury, Malik Ghallab, and Félix Ingrand. An architecture for autonomy. *Internatonal Journal of robotics Research, Special Issue on Integrated Architectures for Robot Control and Programming*, 17(4), 1998.

[ACM+06a] Rachid Alami, Aurélie Clodic, Vincent Montreuil, Emrah Akin Sisbot, and Raja Chatila. Towards human-aware cognitive robotics. In *Cogrob 2006, The 5th International Cognitive Robotics Workshop (The AAAI-06 Workshop on Cognitive Robotics), Boston, USA*, July 16-17 2006.

[ACM+06b] Rachid Alami, Aurélie Clodic, Vincent Montreuil, Emrah Akin Sisbot, and Chatila Raja. Toward human-aware robot task planning. In *AAAI Spring Symposium "To boldly go where no human-robot team has gone before", Stanford, USA*, March 27-29 2006.

[BBG+04] Cynthia Breazeal, Andrew Brooks, Jesse Gray, Guy Hoffman, Cory Kidd, H. Lee, J. Lieberman, A. Lockerd, and D. Chilongo. Tutelage and collaboration for humanoid robots. *International Journal of Humanoid Robots*, 2004.

[BC03] Adrian Bangerter and Herbert H. Clark. Navigating joint projects with dialogue. *Cognitive Science*, 27 :195–225, 2003.

[BHL04] Cynthia Breazeal, Guy Hoffman, and Andrea Lockerd. Teaching and working with robots as a collaboration. *International Conference of Autonomous Agents and Multiagent Systems (AAMAS)*, 2004. Interest : 3 Quality : 3.

[BLD06] L. Brethes, F. Lerasle, and P. Danes. Particle filtering strategies for visual tracking dedicated to h/r interaction. *International Workshop on Vision Based Human-Robot Interaction*, 2006.

[BLH04] P. Brethes, L.and Menezes, F. Lerasle, and J. Hayet. Face tracking and hand gesture recognition for human robot interaction. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1901–1906, 2004.

[Bre03] Cynthia Breazeal. Towards sociable robots. *Robotics and Autonomous Systems*, (42(3-4)) :167–175, 2003.

[CA05] A. Clodic and R. Alami. Specification of an architecture for a cognitive robot. *Cogniron Deliverable D6.1.1, LAAS Internal Report*, 2005.

[CB91] Herbert H. Clark and Susan E. Brennan. *Perspectives on socially shared cognition*, chapter Grounding in communication, pages 127–149. APA Books, 1991.

[CFA+05] Aurélie Clodic, Sara Fleury, Rachid Alami, Matthieu Herrb, and Raja Chatila. Supervision and interaction : Analysis from an autonomous tour-guide robot deployment. *In the 12th International Conference on Advanced Robotics*, 2005.

[CFA+06] Aurélie Clodic, Sara Fleury, Rachid Alami, Raja Chatila, Gérard Bailly, Ludovic Brèthes, Maxime Cottret, Patrick Danès, Xavier Dollat, Frédéric Elisei, Isabelle Ferrané, Matthieu Herrb, Guillaume Infantes, Lemaire Christian, Frédéric Lerasle, Jérôme Manhes, Patrick Marcoul, Paulo Menezes, and Vincent Montreuil. Rackham : An interactive robot-guide. In *IEEE International Workshop on Robots and Human Interactive Communication (ROMAN), Hatfield, UK*, september 6-8 2006.

[CK04] Herbert H. Clark and Meredyth A. Krych. Speaking while monitoring addressees for understanding. *Journal of Memory and Language*, 50 :62–81, 2004.

[CL90] Philip R. Cohen and Hector J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(2-3) :213–361, 1990.

[CL91] Philip R. Cohen and Hector J. Levesque. Teamwork. *Nous*, 25(4) :487–512, 1991.

[Cla96] Herbert H. Clark. *Using Language*. Cambridge University Press, 1996.

[CLS98] Philip R. Cohen, Hector J. Levesque, and Ira Smith. On team formation. *Contemporary Action Theory*, 1998.

[CMAC05] Aurélie Clodic, Vincent Montreuil, Rachid Alami, and Raja Chatila. A decisional framework for autonomous robots interacting with humans. *IEEE International Workshop on Robot and Human Interactive Communication (RO-MAN, Nashville, USA)*, August 13-15 2005.

[Dom07a] Peter Ford Dominey. Progress in programming the hrp-2 humanoid using spoken language. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2007.

[Dom07b] Peter Ford Dominey. Sharing intentional plans for imitation and cooperation : Integrating clues from child developments and neurophysiology into robotics. In *Proceedings of the AISB 2007 Workshop on Imitation*, 2007.

[FHC97] Sara Fleury, Matthieu Herrb, and Raja Chatila. Genom : a tool for the specification and the implementation of operating modules in a distributed robot architecture. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Grenoble, France*, 1997.

[FHM05] S. Fleury, M. Herrb, and A. Mallet. *GenoM : Generator of Modules for Robots*. http ://softs.laas.fr/openrobots/tools/genom.php, 2005.

[FKHB06] Terrence W Fong, Clayton Kunz, Laura Hiatt, and Magda Bugajska. The human-robot interaction operating system. In *2006 Human-Robot Interaction Conference*. ACM, March 2006.

[FNK+05] Terrence W. Fong, Illah Nourbaksh, Clayton Kunz, Lorenzo Flückiger, John Schreiner, Robert Ambrose, Robert Burridge, Reid Simmons, Laura M. Hiatt, Alan Schultz, J. Gregory Trafton, Magda Bugajska, and Jean Scholtz. The peer-to-peer human-robot interaction project. *AIAA Space 2005*, September 2005.

[FSM05] David Feil-Seifer and Maja J. Mataric. A multi-modal approach to selective interaction in assistive domains. *IEEE International Workshop on Robots and Human Interactive Communication*, pages 416–421, 2005.

[FSS+06] Terrence W Fong, Jean Scholtz, Julie Shah, Lorenzo Flueckiger, Clayton Kunz, David Lees, John Schreiner, Michael Siegel, Laura Hiatt, Illah Nourbakhsh, Reid Simmons, Robert Ambrose, Robert Burridge, Brian Antonishek, Magda Bugajska,

Alan Schultz, and J. Gregory Trafton. A preliminary study of peer-to-peer human-robot interaction. In *International Conference on Systems, Man, and Cybernetics*. IEEE, October 2006.

[FTB01] Terrence Fong, Charles Thorpe, and Charles Baur. Collaboration, dialogue, and human-robot interaction. *Proceedings of the 10th International Symposium of Robotics Research*, 2001.

[GBLS07] T. Germa, L. Brethes, F. Lerasle, and T. Simon. Data fusion and eigenface based tracking dedicated to a tour-guide robot. *5th International Conference on Computer Vision Systems (ICVS'2007)*, 2007.

[GK96] Barbara J. Grosz and Sarit Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86 :269–358, 1996.

[GK99] Barbara Grosz and Sarit Kraus. The evolution of sharedplans. *Foundations and Theories of Rational Agencies*, 1999.

[GS86] Barbara Grosz and Candace Sidner. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3) :175–204, 1986.

[HAS06] I. Herrera Aguilar and D. Sidobre. Soft-motion and visual control for service robots. *5th International Symposium on Robotics and Automation (ISRA'2006)*, 2006.

[HB07] Guy Hoffman and Cynthia Breazeal. Effects of anticipatory action on human-robot teamwork efficiency, fluency, and perception of team. In *HRI '07 : Proceeding of the ACM/IEEE international conference on Human-robot interaction*, pages 1–8, New York, NY, USA, 2007. ACM Press.

[ICAR96] F. Ingrand, R. Chatila, R. Alami, and F. Robert. Prs : a high level supervision and control language for autonomous mobile robots. *IEEE International Conference on Robotics and Automation*, 1996.

[Ing05] F. Ingrand. *OpenPRS : an open source version of PRS (Procedural Reasoning Systems)*. http ://softs.laas.fr/openrobots/tools/openprs.php, 2005.

[IP02] F. Ingrand and F. Py. Online execution control checking for autonomous systems. *7th International Conference on Intelligent Autonomous Systems (IAS-7)*, 2002.

[KFBW04] Gary Klein, Paul J. Feltovich, Jeffrey M. Bradshaw, and David D. Woods. *Organizational simulation*, chapter Common Ground and coordination in Joint Activity. W.R. Rouse and K.B. Boff, 2004.

[KHC02] Sanjeev Kumar, Marcus J. Huber, and Philip R. Cohen. Representing and executing protocols as joint actions. In *Proceedings of the First International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS)*, 2002.

[KHCM02] Sanjeev Kumar, Marcus J. Huber, Philip R. Cohen, and David R. McGee. Toward a formalism for conversation protocols using joint intention theory. *Computational Intelligence*, 18(2), 2002.

[LCN90] Hector J. Levesque, Philip R. Cohen, and J. H. T. Nunes. On acting together. In *Proceedings of the Annual Meeting of the American Association for Artificial Intelligence (AAAI)*, 1990.

[Loc98] Karen E. Lochbaum. A collaborative planning model of intentional structure. *Computational Linguistics*, 1998.

[MCA07] V. Montreuil, A. Clodic, and R. Alami. Planning human centered robot activities. In *Submitted to IEEE Int. Conf. on Systems, Man and Cybernetics (SMC)*, 2007.

[MCAR07] V. Montreuil, A. Clodic, R. Alami, and M. Ransan. Task planning with social rules for interactive robots. In *Submitted to IEEE International Conference on Automated Planning and Scheduling (ICAPS)*, 2007.

[MOM04] Javier Minguez, Javier Osuna, and Luis Montano. A divide and conquer strategy based on situations to achieve reactive collision avoidance in troublesome scenarios. In *IEEE International Conf. on Robotics and Automation (ICRA), New Orleans, USA*, 2004.

[MTM03] R. T. Maheswaran, M. P. Tambe, Varakantham, and K. Myers. Adjustable autonomy challenges in personal assistant agents : A position paper. *Proceedings of Autonomy'03*, 2003.

[NF05] Illah R. Nourbakhsh and Terrence Fong. Human-robot teams on the moon : Challenges and plans (extended abstract). *International Conference on Robotics and Automation (ICRA), Workshop on Fielding Multi-Robot Systems*, 2005.

[RS97] Charles Rich and Candace L. Sidner. Collagen : When agents collaborate with people. *Proceedings of the first international conference on Autonomous Agents*, 1997.

[RS98] Charles Rich and Candace L. Sidner. Collagen : A collaboration manager for a collaborative interface agent. *User Modeling and User Assisted Interaction*, 7(3-4), 1998.

[RSL01] Charles Rich, Candace L. Sidner, and Neal Lesh. Collagen : Applying collaborative discourse theory to human-computer interaction. *Artificial Intelligence Magazine, Special Issue on Intelligent User Interfaces*, 2001.

[SA98] R. Simmons and D. Apfelbaum. A task description language for robot control. *Proc. Conference on Intelligent Robots and Systems*, 1998.

[SAS+05] E. A. Sisbot, R. Alami, T. Simeon, K. Dautenhahn, M. Walters, S. Woods, K. L. Koay, and C. Nehaniv. Navigation in presence of humans. *IEEE-RAS International Conference on Humanoid Robots, Humanoids2005, Tsukuba, Japan*, 2005.

[SCMU+06] Emrah Akin Sisbot, Aurélie Clodic, Luis Felipe Marin Urias, Matthias Fontmarty, Ludovic Brethes, and Rachid Alami. Implementing a human-aware robot system. In *IEEE International Workshop on Robots and Human Interactive Communication (ROMAN), Hatfield, UK*, September 6-8 2006.

[Sid94] Candice L. Sidner. An artificial discourse language for collaborative negotiation. In Barbara Hayes-Roth and Richard Korf, editors, *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 814–819, Menlo Park, California, 1994. AAAI Press.

[SLK+05] Candace L. Sidner, Christopher Lee, Cory Kidd, Neal Lesh, and Charles Rich. Explorations in engagement for humans and robots. *Artificial Intelligence*, 166(1-2) :140–164, 2005.

[SPS+04] Paul Scerri, David Pynadath, Nathan Schurr, Alessandro Farinelli, Sudeep Gandhe, and Milind Tambe. Team oriented programming and proxy agents : The next generation. In *In Proceedings of 1st international workshop on Programming Multiagent Systems*, 2004.

[SUAS06] E. Akin Sisbot, Luis F. Marin Urias, Rachid Alami, and Thierry Siméon. A mobile robot that performs human acceptable motion. *Proc in (IEEE/RSJ) International Conference on Intelligent Robots and Systems*, 2006.

[Tam97] Milind Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7 :83–124, 1997.

[TBP+06] Milind Tambe, Emma Bowring, Jonathan Pearce, Pradeep Varakantham, Paul Scerri, and David Pynadath. Electric elves : What went wrong and why ? *Proceedings of the AAAI Spring Symposium on "What Went Wrong and Why"*, 2006.

[TCB+05] J. Gregory Trafton, Nicholas L. Cassimatis, Magdalena D. Bugasjska, Derek P. Prock, Farilee E. Mintz, and Alan C. Schultz. Enabling effective human-robot interaction using perspective-taking in robots. *IEEE Transactions on Systems, Man and Cybernetics*, 35(4) :460–470, 2005.