Compréhension dynamique du contexte pour l'aide à l'opérateur en robotique

Mohamed Walid Ben Ghezala

DOCTORAT EN CO-ACCREDITATION
TELECOM SUDPARIS ET L'UNIVERSITE EVRY VAL D'ESSONNE

Spécialité: Informatique

Ecole doctorale: Sciences et Ingénierie

Présentée par

Mohamed Walid Ben Ghezala

Pour obtenir le grade de
DOCTEUR DE TELECOM SUDPARIS

Compréhension Dynamique du Contexte pour l'Aide à l'Opérateur en Robotique

Soutenue le 21 Juillet 2015

devant le jury composé de :

Rapporteurs :

Pr. Chantal Reynaud	Professeur, LRI, Paris 11, France
Pr. Rachid Alami	Directeur de Recherche du LAAS/CNRS, France

Examinateurs :

Pr. Bernadette Dorizzi	Professeur, TELECOM SudParis, France
Pr. Etienne Colle	Professeur, Université d'Evry, France
M. Stéphane Zieba	Docteur, AKEO PLUS, France

Directeurs de thèse :

Pr. Amel Bouzeghoub	Professeur, TÉLÉCOM SudParis, France
Dr. Christophe Leroux	Docteur, LRI, CEA-LIST, France (Encadrant)

Thèse numéro : 2015TELE0006

RÉSUMÉ

Les technologies de l'informatique et de la robotique sont en perpétuelle évolution. S'appuyant sur cette évolution technologique, les systèmes d'aide à l'opérateur restent un domaine de recherche d'actualité. Le principal défi des systèmes de la future génération est d'être "intelligents", sensibles au contexte dans un environnement complexe et imprévisible. Cette thèse entre dans ce cadre et traite de la compréhension dynamique du contexte par un robot évoluant dans un tel environnement. En particulier, elle s'intéresse à la question suivante : Comment rendre un robot capable de réagir face aux situations de blocage, imprévues dans son plan d'action initial, pour accomplir l'objectif fixé par l'opérateur ? Dans la littérature, ce problème a été soulevé et résolu en partie en programmant, dans le système robotique, certaines fonctions, telles que la génération des plans d'action, en adoptant un raisonnement géométrique et aussi symbolique. La procédure généralement adoptée se déroule en trois phases : le choix d'un objectif à atteindre à partir de l'état initial, la planification et la supervision de l'exécution. En cas de problème d'exécution détecté au niveau de la supervision, il y a une reconstruction de l'état courant et un nouvel appel au planificateur. Certes, ces systèmes ont permis de noter une avancée certaine en rendant le robot plus autonome. Toutefois on constate que dans ces systèmes robotiques autonomes, d'une part les connaissances sont fortement couplées dans le code, et, d'autre part, la planification reste du niveau de la tâche et le planificateur peut ne pas trouver de plan lorsqu'il considère que l'état initial n'est pas valide ou qu'il n'a pas les opérateurs nécessaires pour trouver un plan d'action. Cette thèse propose un système supportant une approche complète et générique, qui assure à un robot la capacité d'être conscient de la situation de blocage dans laquelle il se trouve, de la comprendre et ainsi y faire face. Cette approche, nommée Robot Situation AWareness (RSAW) se propose de permettre à un système robotique de produire un plan, non seulement en planifiant, mais aussi en appliquant un raisonnement lui permettant de profiter de son expérience passée. Les principales contributions dans RSAW portent sur la conception d'un cadre sémantique intégrant la capacité de compréhension, fondé sur une représentation des connaissances générique et donnant la possibilité d'appliquer des techniques de raisonnement empruntées aux sciences cognitives. La structure d'ontologie est adoptée pour représenter les connaissances relatives aux situations de blocage. L'approche IDM est utilisée pour concevoir la représentation des connaissances de façon abstraite et selon différents points de vue (méta modèle, modèles), et pour profiter de la notion de transformation de modèles pour concevoir les raisonnements. Pour la compréhension des situations de blocage, le raisonnement conçu est fondé sur deux mécanismes. Le premier mécanisme est un raisonnement déductif pour élaborer l'interprétation des situations de blocage. Le deuxième mécanisme est un raisonnement par analogie permettant de profiter de l'expérience passée du robot afin de générer un nouveau plan d'action. L'intégration de RSAW dans un système robotique a également été étudiée, conçue et mise en œuvre dans un système à couches. Ce système d'expérimentation est le robot SAM (Smart Autonomous Majordomo) doté du système AVISO et développé par le CEA. Les expérimentations menées ont permis de tester le raisonnement déductif dans la résolution des situations de blocage et d'affirmer le besoin d'avoir recours au raisonnement par analogie. Une autre vague d'expérimentation a eu lieu pour prouver l'efficacité de nos choix. Les résultats des travaux menés et expérimentés dans cette thèse sont concluants et prometteurs.

ABSTRACT

Computer technology and robotics are in perpetual evolution. Based on this technological evolution, the operator support systems remain a topical domain of research. The main challenge for the next generation of systems is to be "intelligent", aware of the context in a complex and unpredictable environment. This thesis is into this framework and addresses the dynamic understanding of the context by a robot evolving in such an environment. In particular, the work is interested in the question : How to make a robot able to react to blocked situations unplanned in its initial action plan to achieve the goal set by the operator ? In literature, this issue was raised and resolved in part by programming in robotic system, some functions, such as generating action plans, adopting a geometric and also symbolic reasoning. The procedure generally adopted is in three phases : the choice of a goal from the initial state, planning and supervision of execution. If execution problem detected in supervision, there is a reconstruction of the current state and a new call to the planner. These systems have noted some progress while making it more autonomous robot. However we note that in these autonomous robotic systems knowledge are strongly coupled in the code. Also the planning stays in the level of the task and the planner can not find an action plan when it considers that the initial state is invalid or has not the required operators to find an action plan. This thesis proposes a system supporting a complete and generic approach that ensures a robot the ability to be aware of the blocking situation in which it is found, to understand and deal with it. This approach, called Robot Situation Awareness (RSAW) allows a robotic system to produce a plan, not only in planning but also by applying reasoning allowing it to benefit from past experience. The main contributions in RSAW relate to design a semantic framework integrating the understanding capacity, based on a generic representation of knowledge and making it possible to apply reasoning techniques borrowed from cognitive science. The ontology structure is adopted to represent knowledge about the blocked situation. The MDE approach is used to design an abstract knowledge representation according to different points of view (meta model, models) and to benefit of the concept of model transformation to design reasoning. To understand the blocked situation, the designed reasoning is based on two mechanisms. The first mechanism is a deductive reasoning to interpret the blocked situations. The second mechanism is an analogical reasoning to benefit from the past experience of the robot for generating a new action plan. Integrating RSAW in a robotic system has also been studied, designed and implemented in a layer system. This experimental system is the robot SAM (Smart Autonomous Majordomo) with the AVISO system developed by CEA. The conducted experiments allowed testing of the deductive reasoning in resolving a blocked situation and confirmed the need to resort to analogical reasoning. Another wave of experimentation has taken place to prove the effectiveness of our choices. The results of the work done and experienced in this thesis are successful and promising.

REMERCIEMENTS

Ces travaux de thèse ont été menés dans le cadre du projet Digiteo Roboteo pour la mise en oeuvre d'une plateforme robotique visant le développement d'applications dans les domaines d'assistance robotique, de manipulation et des véhicules autonomes.

Je tiens à exprimer ma profonde gratitude à ma directrice de thèse Pr. Amel Bouzeghoub et mon encadrant Dr. Christophe Leroux. Je les remercie pour leur disponibilité et leurs nombreux conseils.

J'adresse toute ma reconnaissance aux membres du jury de soutenance de thèse pour leurs observations enrichissantes et leurs conseils. Je remercie tout spécialement Pr. Chantal Reynaud, Pr. Bernadette Dorizzi, pour avoir accepté d'être dans mon comité de thèse, ainsi que Pr. Rachid Alami pour ses remarques très constructives.

Je remercie M. Yann Perrot et Pr. Bruno Defude pour m'avoir accueilli, respectivement, au Laboratoire de Robotique Interactive (LRI) du CEA-LIST et au Département Informatique de Télécom SudParis, et m'avoir donné l'opportunité d'effectuer une thèse en lien avec les domaines qui m'ont toujours intéressé, à savoir la robotique et l'informatique.

Je souhaite remercier mes collègues du LRI dont l'aide et l'expérience m'ont été précieuses pour la compréhension et la prise en main des systèmes robotiques, en particulier Olivier Lebec, Stéphane Zieba, Julie Dumora et Niccolo' Tosi. Je remercie Ali Ben Messaoud pour la collaboration à l'occasion de son travail de fin d'études. Je remercie Elodie Rénividaud, Dominique Schoen, Martine Guerrand ainsi que Marie Rivasseau pour leur disponibilité, leur soutien et leurs précieux aides et conseils concernant toutes les questions administratives et informatiques.

Je souhaite également remercier les professeurs et les collègues de Télécom SudParis tout particulièrement Pr. Samir Tata pour sa présence, sa gentillesse et ses encouragements ainsi que Dr. Alda Gancarski pour ses relectures constructives et pour sa gentillesse. Mes remerciements s'adressent à tous mes amis, en particulier Fethi Belghaouti, Mourad Amziane, Nour Assli, Sami Yangui et Rami Sellami.

Je remercie mon père Mohamed pour m'avoir inculqué les principes nécessaires à mon éducation, ma mère Henda pour tout l'amour qu'elle nous porte et pour sa précieuse présence. Je remercie chaleureusement ma soeur Foufa, son mari Majdi et son petit ange Mayouta.
A ma femme Amira, pour ses encouragements, son soutien et sa patience, merci.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

INTRODUCTION

Sommaire

1.1 Cadre de travail

Avec la mondialisation, le secteur des services ne cesse de se développer. Aujourd'hui, nous entamons une nouvelle révolution cherchant à optimiser les outils de production en vue de permettre aux industriels de produire plus rapidement, à meilleur cout, tout en étant plus écologique [18][12]. Cette révolution a donné naissance au concept de la ville intelligente (smart city) [224] et l'industrie 4.0 [80], basées sur les nouvelles technologies et l'innovation. Parmi les technologies émergentes et utiles à la mise en œuvre de la génération future de l'industrie et des villes, nous pouvons citer les plus prometteuses, à savoir, les robots de services intelligents [261], les capteurs intelligents [155], le Big Data [102], le cloud computing [259][229][171], etc. De nos jours, plusieurs travaux de recherche portent sur la robotique intelligente et à doter les robots de services de techniques d'intelligence artificielle afin d'augmenter leur autonomie dans la réalisation des désirs de leurs opérateurs.
Avec les avancées technologiques constatées ces dernières années, les robots commencent à communiquer entre eux [230], les interfaces homme-robot deviennent naturelles et intuitives comme la reconnaissance vocale ou gestuelle [238] et les capteurs facilitent la localisation et les échanges d'information sans contact. Aussi, les robots ont franchi le seuil de la maturité et ont trouvé un réel usage auprès de nombreux opérateurs comme, par exemple, les systèmes robotiques dans les chaines de production et les cellules de travail, dans les systèmes de télé-opération ou encore dans les systèmes d'aide aux personnes à mobilité réduite. Ces développements conduisent à des robots perfectionnés, capables de réaliser des tâches de plus en plus sophistiquées et maitrisées par leurs opérateurs. Seulement, la difficulté de la maitrise d'un robot par un opérateur augmente avec la complexité du robot, c'est ainsi que la tendance est de réduire l'intervention de l'opérateur en augmentant le degré d'autonomie du

robot qui évolue dans un environnement dynamique et imprévisible. S'appuyant sur cette évolution technologique, les systèmes d'aide à l'opérateur demeurent un domaine de recherche d'actualité. Le principal objectif de ces systèmes est d'être sensible au contexte défini par l'environnement, l'opérateur et le robot. Cela signifie que le robot devrait tenter d'accomplir le désir de l'opérateur en s'adaptant aux différents changements intervenant dans le contexte au cours du temps.

Les technologies et services pour l'autonomie constituent un enjeu économique porteur. Dans certains pays, l'innovation portée par la recherche académique est fortement encouragée, donnant naissance à un tissu industriel de PME très dynamiques qui réinvestissent une part importante de leur chiffre d'affaires en Recherche, Développement et Innovation (R&D&I). Ces recherches sont entreprises dans les domaines spécifiques de la cognition, l'interaction homme-robot [154], la navigation et la perception [242]. Plus encore, des recherches très poussées combinent l'ensemble de ces disciplines pour œuvrer à l'amélioration de l'autonomie des robots matérialisée par la compréhension des objets et leur manipulation, la planification réactive etc.

Nos travaux de thèse se situent dans le cadre de la compréhension du contexte où un robot, évoluant dans un environnement complexe et imprévisible, peut se trouver bloqué dans ses actions. Dans la littérature, les travaux s'intéressant à cette question de recherche considèrent tous les cas de figure pouvant induire à un blocage d'une manière prédéfinie ou stéréotypée. Généralement, face à une situation imprévue, le robot se bloque et attend une intervention de son opérateur pour terminer sa mission.

Notre thèse aborde cette problématique des situations de blocage. L'objectif est de rendre un robot capable d'affronter les blocages rencontrés afin qu'il puisse accomplir les tâches le désirées de l'opérateur.

1.2 Problématique et contribution

Plusieurs équipes de recherche visent à construire des robots intelligents capables d'évoluer dans des environnements variés, connus ou non, et de réaliser des tâches de plus en plus complexes de la manière la plus autonome possible. De tels robots intelligents seront capables d'acquérir et de retenir des connaissances, d'apprendre ou de comprendre grâce à leur expérience et d'utiliser la faculté de raisonnement pour résoudre des problèmes et répondre d'une manière appropriée.
Notre recherche, menée dans le cadre de cette thèse, vise à offrir une approche, générique et facilement intégrable dans un système robotique, permettant d'augmenter l'autonomie d'un robot pour satisfaire le désir d'un opérateur. Cette approche vise à rendre le robot capable de comprendre les situations de blocage pouvant survenir et déclencher des comportements adéquats pour résoudre le blocage et atteindre l'ob-

jectif fixé par l'opérateur.
De cette problématique découlent trois verrous scientifiques qui sont :

1. Comment offrir à un système robotique la capacité de compréhension du contexte ?
2. Quelle architecture d'intégration et quelle représentation des connaissances adopter pour répondre au critère de généricité ?
3. Quels raisonnements appliquer pour comprendre et aider à la prise de décision en profitant de l'expérience passée ?

Pour répondre à ces questions, nous adoptons le principe consistant à rendre un robot conscient de la situation, principe fondé sur un raisonnement opérant sur différentes connaissances hétérogènes. Pour ce faire, nous avons conçu une approche, dénommée RSAW, générique et facilement intégrable dans un système robotique, proposant (I) d'intégrer les fonctions rendant le robot capable de comprendre la situation de blocage dans laquelle il se trouve, (II) un cadre sémantique proposant une représentation générique des connaissances et (III) un raisonnement pour comprendre les situations en vue d'aider dans la prise de décision.

Nos principales contributions dans cette thèse se résument ainsi :

I Une approche, RSAW, pour l'intégration de la capacité de compréhension : Face aux changements imprévisibles de l'environnement et à la diversité des tâches à accomplir par un robot, les fonctions le rendant capable de comprendre deviennent indispensables pour assurer son autonomie. A cet effet, nous proposons l'approche RSAW intégrant dans un même cadre ces fonctions, permettant ainsi à un robot d'affronter les blocages afférents à l'imprévisibilité de l'environnement. Cette approche a été conçue, implémentée et intégrée dans un système robotique réel.

II Un cadre sémantique proposant une représentation générique des connaissances dans RSAW : Pour favoriser la généricité de notre approche et la facilité de son intégration dans un système robotique, des choix relatifs à la représentation des connaissances dans RSAW ont été pris. La structure d'ontologie est adoptée pour représenter les situations du fait qu'elles offrent dans un même cadre une puissante représentation des connaissances ainsi que la possibilité de raisonnement. L'organisation des ontologies suit l'approche Ingénierie Dirigée par les Modèles (IDM). Cette méthodologie permet, d'une part, de concevoir la représentation des connaissances de façon abstraite et selon différents points de vue (meta modèle, modèles), et d'autre part, de profiter de la notion de transformation de modèles pour concevoir les raisonnements.

III Un raisonnement pour l'interprétation des situations et la préparation de la prise de décision : Le raisonnement dans RSAW est le noyau de la compréhension des situations de blocage puisque l'efficacité de notre approche en dépend. En effet, ce raisonnement participe à l'interprétation des situations de blocage ainsi qu'à

la préparation de la prise de décision. Il a pour objectif, d'une part, d'uniformiser la syntaxe utilisée dans la construction des situations à partir de données hétérogènes et d'autre part, de vérifier l'existence d'une situation de blocage semblable dans l'expérience passée du robot pour s'en inspirer.

1.3 Organisation du mémoire

Ce manuscrit de thèse est composé, en plus de cette introduction générale, de trois parties principales, d'une conclusion générale et d'une partie annexe :
La première partie, intitulée "Systèmes d'aide à l'opérateur en robotique", présente un état de l'art sur la robotique et l'autonomie des systèmes robotiques. Elle dresse, dans le chapitre 2, un panorama sur l'évolution des robots, les concepts de base de la robotique et de la compréhension du contexte dans les systèmes robotiques, les architectures de contrôle ainsi que les plateformes logicielles existantes dans la littérature. Dans le chapitre 3, destiné à présenter les différents travaux autour de nos verrous scientifiques identifiés plus haut, une focalisation sur les systèmes robotiques intelligents est faite notamment autour des architectures intégrant la capacité de compréhension ainsi que la représentation des connaissances et le raisonnement adoptés dans les systèmes intégrant l'"intelligence". Nous concluons ce chapitre par une étude comparative des ces travaux en vue de positionner notre approche.

La deuxième partie, intitulée "Compréhension dynamique du contexte dans un système robotique", présente nos contributions élaborées dans cette thèse. Dans cette partie, trois chapitres décrivent nos choix et notre proposition.
Le chapitre 4 traite de notre approche «RSAW" (pour Robot Situation AWareness) intégrant la capacité de compréhension ainsi que son intégration dans un système robotique.
Le chapitre 5 motive nos choix de représentation des connaissances, à savoir la structure, le formalisme de description et l'organisation des connaissances. L'élaboration de la représentation des connaissances du contexte dans RSAW est par la suite décrite.
Le chapitre 6 traite du raisonnement appliqué dans RSAW pour la compréhension du contexte. Après une motivation des choix des mécanismes de raisonnement répondant à nos besoins, l'interprétation des données perçues par les capteurs du robot ainsi que la préparation de la prise de décision sont détaillées.

La troisième partie, intitulée "Etude expérimentale de l'aide à l'opérateur à partir du module logiciel de compréhension dynamique du contexte", présente la mise en œuvre d'un système supportant RSAW ainsi qu'une étude expérimentale menée pour montrer la diminution du taux d'intervention de l'opérateur grâce à notre approche de compréhension dynamique du contexte. Cette partie est composée de deux chapitres. Le premier chapitre 7 présente la conception et les développements effectués durant cette thèse pour la mise en œuvre de RSAW. Les exigences d'intégration

dans un système robotique sont également présentées et décrites. Le deuxième chapitre 8 présente, quant à lui, la préparation d'un assistant robotique, dénommé Smart Autonomous Majordomo (SAM) sur lequel des évaluations techniques et cliniques au sein de deux centres de réadaptation ont été réalisées. L'implémentation de RSAW dans la plateforme logicielle de SAM est ensuite décrite, suivie des différentes expérimentations et améliorations faite pour aboutir à des résultats concluants.

La conclusion générale rappelle les contributions de la thèse et dresse certaines perspectives pour ce travail de recherche.

La dernière partie comporte quatre annexes fournissant de plus amples informations sur des concepts clés utilisés dans ce travail.

Première partie

SYSTÈMES D'AIDE À L'OPÉRATEUR EN ROBOTIQUE

Chapitre 2

SYSTÈMES ROBOTIQUES

Sommaire

2.1 Introduction

Depuis les années cinquante, la robotique évolue grâce à l'essor de l'informatique et aux progrès de la recherche en intelligence artificielle, c'est ainsi que des robots contrôlés par des ordinateurs et même par des équipements mobiles continuent à voir le jour. Aujourd'hui, les chercheurs dans différentes disciplines concourent pour profiter de ces évolutions technologiques et développer l'autonomie des robots pour devenir capables non seulement de produire un comportement intelligent face aux événements, mais aussi d'avoir une réelle conscience du monde réel et une compréhension de leurs propres actions.
Ce chapitre traite des systèmes robotiques en dressant leur évolution, en précisant les principaux concepts de ces systèmes ainsi que leurs architectures et les plateformes généralement adoptées.

2.2 Robotique : Historique et évolution

La robotique est la science qui se concentre et se préoccupe des robots. Il s'agit d'un domaine de recherche vaste et pluridisciplinaire qui intègre la mécanique, l'automatique, l'électronique, l'informatique et l'intelligence artificielle, les neurosciences et les sciences humaines et sociales. Les robots ont fait un incontestable progrès au cours des dernières années. Les améliorations dans le stockage d'énergie, la disponibilité de processeurs peu coûteux et hautement intégrés aux applications embarquées, ont largement contribué à doter les robots mobiles de fonctionnalités avancées telles qu'une vision à haute résolution, un logiciel de contrôle sophistiqué, etc...
En 1961, les robots industriels ont vu le jour à travers le robot "Unimate" [195]. Ce robot installé dans un entrepôt de General Motors, a fait ses preuves et l'utilisation des robots s'est par la suite généralisée sur les chaînes de montage.
En 1966 dans un laboratoire de l'université de Caroline du Sud, le robot Phoney Pony a été conçu. Ce système robotique est une table composée de quatre pieds dont les mouvements sont contrôlés par ordinateur.
En 1973, au Japon et plus précisément à l'université de Waseda [126], les robots humanoïdes capables de marcher sont apparus avec Wabot. Ce robot peut parler, peut évaluer les distances et la direction d'objets en mouvement et les attraper.
Dans les années 80, les robots n'interagissaient pas avec l'humain : ils le remplaçaient pour des tâches simples et répétitives, à l'image des robots des chaînes de montage dans les usines. Dans les années 90, ils commencent à assister les humains dans certaines tâches, par exemple en s'invitant dans les salles d'opération, des robots prolongent le bras du chirurgien. Depuis, des robots capables de faire des tâches de plus en plus sophistiquées ont vu le jour, tels que le robot compagnon Aïbo ou encore l'androïde Asimov [168] ayant la capacité de reconnaître les visages, de se comporter face à des escaliers, d'écouter la parole humaine (comprendre) et de faire l'analyse de son environnement. Les robots sont devenus très répandus dans le quotidien des

êtres humains sous la forme de jouets, d'outils de nettoyage (tondeuse, aspirateur), d'explorateurs d'endroits dangereux ou inaccessibles pour l'humain.
Aujourd'hui, sur le marché, les applications commerciales de la robotique ne montrent qu'un faible niveau d'autonomie : la plupart des robots sont dirigés soit à distance ou effectuent des tâches nécessitant un contrôle réactif trivial. Pourtant, la recherche en robotique a beaucoup progressé. C'est ainsi que les robots autonomes peuvent cartographier des environnements inconnus, détecter des objets et des personnes, et exécuter des tâches d'assistance de manière autonome.

2.3 Robot

Dans cette section, nous dressons des généralités sur les robots. En particulier, nous exposons ses composantes et ses différentes catégories.

2.3.1 Définitions

Le mot robot trouve son origine dans les langues slaves où le terme "robota" signifie travail forcé. Dans la littérature, plusieurs définitions de ce que peut être un robot sont données. En général, un robot est considéré comme une "machine programmable qui imite des actions d'une créature intelligente" [51].
D'après cette référence, nous retenons qu'un robot est un agent physique ayant pour objectif de réaliser des tâches dans l'environnement dans lequel il évolue afin de répondre au désir de son opérateur.
Généralement, les agents/robots évoluent dans des environnements dynamiques et incertains, pouvant être difficilement accessibles. Toutefois, les robots n'ont qu'une capacité limitée pour percevoir l'environnement dans lequel ils évoluent car ils n'en ont qu'une vision locale, laquelle peuvent s'ajouter des erreurs et imprécisions sur leurs capteurs. La partie de l'environnement qu'un robot peut percevoir grâce à ses capteurs constitue l'espace de perception.

2.3.2 Composantes d'un robot

En général, un robot agit conformément à une boucle de décision (Fig. 2.1). Il est capable d'extraire de l'information à partir de son environnement et d'utiliser ses connaissances pour décider comment agir. Ceci suppose qu'en plus de sa structure mécanique ou squelette, le robot est équipé de capteurs, d'effecteurs et d'un contrôleur. Les principales composantes d'un robot sont donc :

- Les capteurs : Les robots sont équipés de capteurs leur permettant de percevoir l'environnement dans lequel ils évoluent : caméras, sonars, détecteur de lumière, boussole, GPS, détecteur de chaleur... Les capteurs peuvent être plus ou moins précis : ils ont une portée, une certaine incertitude des mesures, etc...

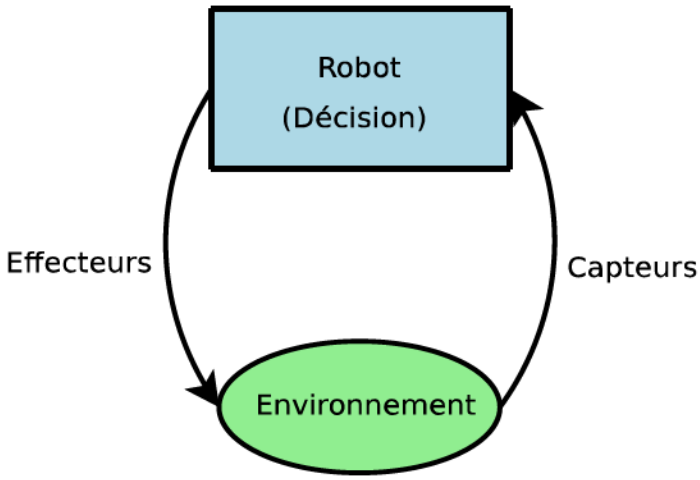


FIGURE 2.1 – Boucle de décision

- Les effecteurs : Les robots sont équipés d'effecteurs leur permettant d'agir dans l'environnement : roue, bras, jambes, pinces... Les effecteurs peuvent également être plus ou moins précis notamment en ce qui concerne l'exactitude des déplacements, les déviations dues à l'environnement, les événements externes imprévus, etc.
- La partie de commande ou cerveau : C'est cette partie qui va permettre au robot d'analyser les données provenant des capteurs et d'envoyer les ordres aux effecteurs. La partie commande est matérialisée physiquement par les logiciels informatiques dotant le robot d'une certaine intelligence dite artificielle.

De nombreuses autres composantes peuvent être présentes, selon le type de robot considéré. De telles composantes peuvent concerner les périphériques de stockage (des programmes, des informations sur l'environnement, des connaissances,...), les interfaces de communication (écran, wifi...), les unités d'alimentation (batteries, panneaux solaires,...) etc.

2.3.3 Types de robots

Il apparaît de ce qui précède que la principale raison d'être d'un robot est d'effectuer des tâches répétitives et/ou précises, dans des environnements de travail plus ou moins dangereux pour l'Homme. En fait, différents types de robots sont recensés dans notre vie quotidienne. Dans ce manuscrit, nous traitons la classification des types de robots par deux points de vue complémentaires. Le premier étant la classification par type d'application et le deuxième étant la classification par type de fonctionnalité.

2.3.3.1 Classification par type d'application

Étant donnée la diversification des domaines où l'homme fait appel aux services des robots et la diversité des tâches pouvant être rendues par ces robots, nous proposons de classer les robots selon le type d'applications où ils peuvent être utilisés. Nous dénotons ainsi les types suivants que nous classons par type d'application. Cette classification est décrite ci-dessous.

a Robots industriels

Les robots industriels sont des robots utilisés dans un environnement de fabrication industrielle. Habituellement, ces robots sont représentés par des bras articulés spécifiquement développés pour des applications telles que le soudage, la manutention, le Co-Working, la peinture et autres. Ce type de robots incluent également certains véhicules guidés automatisés [38].

b Robots domestiques

Les robots domestiques sont utilisés à la maison. Ce type de robots comprend de nombreux robots très différents, tels que robots nettoyeurs de parquet, robots nettoyeurs de piscine, balayeuses, nettoyeurs de gouttières et autres robots qui peuvent faire différentes tâches. En outre, certains robots de surveillance et de télé présence peuvent être considérés comme des robots domestiques s'ils sont utilisés dans un tel environnement [202].

c Robots pour la santé

Les robots pour la santé sont les robots utilisés en médecine et les institutions médicales. Ce type de robots comprend les robots de chirurgie (télé-opération ou autres), les robots de réhabilitation, les exosquelettes (aide à la marche pour personnes handicapées), les robots d'assistance aux personnes dépendantes qui sont généralement des robots de service à la personne [129][170].

d Robots militaires

Les robots militaires sont les robots utilisés dans les applications militaires. Ce type de robots rassemble les robots démineurs, les robots de transport, les exosquelettes (aide pour soulevé des charges), les drones de reconnaissance. Souvent, ces robots sont créés initialement à des fins militaires mais ils sont utilisés dans l'application de loi, dans les recherches et le sauvetage de personnes et dans d'autres domaines connexes [232].

e Robots de divertissement

Ce sont des robots utilisés pour le divertissement des personnes. C'est une catégorie très vaste. Cette catégorie englobe les robots jouets tels que "Robosapien" ou le "Running Alarm Clock", mais aussi les bras de robot articulés comme les simulateurs de mouvement [140].

f Robots explorateurs

Ce type de robots est utilisé dans les milieux dangereux ou inaccessibles pour l'être humain. Cette catégorie comprend les robots utilisés dans la Station Spatiale Internationale (SSI), le "Canadarm≫" utilisé dans les navettes spatiales, ainsi que martian rovers (Mars Exploration Rover) [123], les robots sous-marins tels que "Nemo" [204]. Les robots sont donc utilisés dans les usines mais également dans les laboratoires pharmaceutiques ou encore dans les hôpitaux, etc. Ils sont généralement présents dans différents domaines d'application et effectuent diverses tâches telles que la manipulation d'objets ou l'exploration [100].

2.3.3.2 Classification par catégorie

Pour effectuer les tâches pour lesquelles ils ont été créés, les robots doivent répondre à certaines exigences telles que la manipulation, le déplacement et la prise de décision. Les catégories de robots connues dans la littérature est la suivante :

- Robots volants
- Robots sous-marins
- Robots humanoïdes
- Robots manipulateurs
- Robots mobiles à roues

a Robots volants

Les robots volants sont appelés le plus souvent des drônes ou des UAV (pour Unmanned Aerial Vehicle). Ces robots volants, sans pilote, évoluent dans les environnements d'une façon télécommandée ou autonome. Ces robots sont destinés le plus souvent à des missions de surveillance, de combat, de renseignement, d'exploration, etc.. Dès leurs apparition, ces robots ont un usage militaire au profit des forces armées ou de sécurité (police, douane, etc.) d'un État. Après cela, l'utilisation de ces robots a été répandue au quotidien dans des applications de divertissement et de santé [162].

FIGURE 2.2 – Exemple de drône utilisé dans une application militaire

b Robots sous-marins

Les robots sous-marins sont le plus souvent appelés ROV (Remotely Operated Vehicle) ou AUV (Autonomous Underwater Vehicles). Ces robots évoluent sous l'eau. Ils sont utilisés généralement dans des applications militaires, dans des aspects de déminage. Leurs principales activités, durant ces dernières années, sont la surveillance, la climatologie, l'observation et l'océanographie [161].

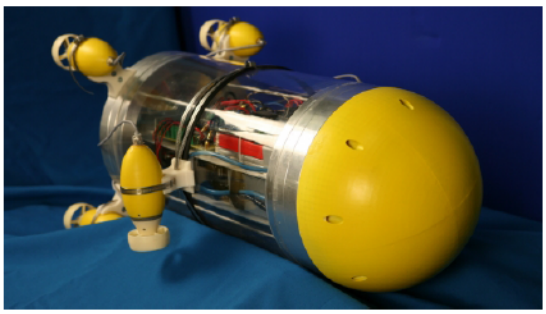


FIGURE 2.3 – Exemple d'un robot sous-marin utilisé dans une application militaire

c Robots humanoïdes

Un robot humanoïde est appelé souvent androïde. Cette catégorie de robots a une apparence proche de celle d'un corps humain. Ils ont généralement un torse avec une tête, deux jambes et deux bras. Dans la littérature, il existe des modèles ne représentant qu'une partie de la nomenclature de l'humain, par exemple jusqu'à la taille. Certains robots humanoïdes peuvent avoir un « visage » ressemblant à celui de l'humain comportant des yeux et une bouche comme le robot HRP-4C [125]. Ces robots peuvent être utilisés dans plusieurs domaines d'application notamment ceux de l'exploration et du divertissement.

FIGURE 2.4 – Exemple de robots humanoïdes

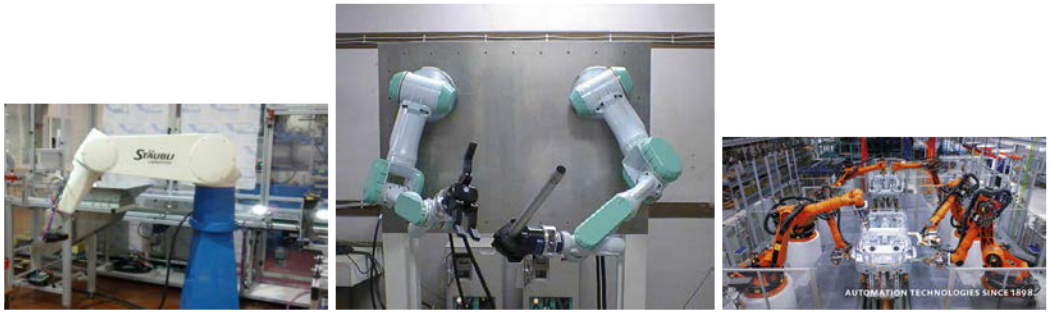
d Robots manipulateurs

Les robots manipulateurs [87] sont ancrés physiquement à leur place de travail et généralement mis en place pour réaliser une tâche précise ou répétitive (Fig. 2.5). De complexité variable, les robots manipulateurs peuvent être :

- Des automates : dans ce cas, le robot exécute indéfiniment une même série d'actions, sans aucune perception de son environnement [223].
- Réactifs : dans ce cas, le robot exécute une action selon l'état actuel de l'environnement dans lequel il évolue (le mapping état/action est fixé initialement)[23].
- Cognitifs : dans ce cas, le robot analyse son environnement et décide de la meilleure action à effectuer [6].

En général, cette catégorie caractérise les types de robots suivants :

- Robots industriels tels que dans les chaînes de montage, la manipulation de produits chimiques,...
- Robots d'assistance tels que pour l'assistance médicale, aux personnes âgées,...



(a) Stäubli Arm (b) Dual-Arm Robot Teleoperation in Space (DARTS) (c) KUKA Arm

FIGURE 2.5 – Exemple de robots manipulateurs

e Robots mobiles

Les robots mobiles sont capables de se déplacer dans un environnement [209]. Ils peuvent être équipés de manipulateurs, suivant leur utilisation. Cette catégorie, comprend les robots explorateurs, les robots de services, les robots de divertissement et ceux reconfigurables.

1. Robots explorateurs : Les robots explorateurs sont destinés à explorer des environnements où l'homme ne peut pas se rendre (Fig 2.6). Comme décrit plus haut, ces robots peuvent être utilisés pour le déminage de terrains ou pour explorer de nouvelles planètes, des zones radioactives, des épaves ou décombres (suite à un tremblement de terre, par exemple), etc...

(a) Mars Rover (b) Mule pour l'armée américaine (c) Mini Andros pour le déminage

FIGURE 2.6 – Robots explorateurs

2. Robots de services : Les robots de services sont destinés à aider l'homme pour certaines tâches, en général difficiles ou répétitives (Fig 2.7). On trouve ainsi des robots agricoles, des robots de transport de marchandises, des robots ménagers, des robots d'assistance aux personnes (personnes âgées ou ayant un handicap), des robots de divertissement, des robots guide de musée, etc...
3. Robots reconfigurables : Dans cette catégorie, chaque robot est considéré comme une petite entité. Les systèmes de robots reconfigurables (Fig 2.8) sont composés d'un grand nombre de ces entités, ces dernières pouvant s'auto-organiser pour réaliser des tâches complexes (Exemple : le projet MAAM [98]).

Le but des robots reconfigurables est d'arriver à obtenir des robots composés de plusieurs unités qui s'auto-organisent au moyen de reconfigurations dynamiques (matérielles ou logicielles) afin de coopérer pour répondre aux besoins de leur opérateur [178]. Aussi, la reconfiguration de ses unités permet une adaptation du robot à son environnement qui peut varier selon les tâches qui lui sont attribuées.

Tous les robots de cette classification ont en commun le besoin d'être munis d'un système informatique leur permettant de comprendre le contexte dans lequel ils évoluent pour répondre aux attentes de l'opérateur. Dans ce qui suit, nous abordons la question de la compréhension pour l'aide à l'opérateur en robotique.

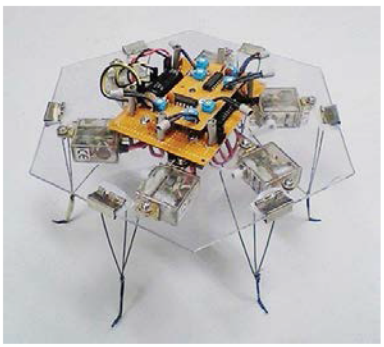
(a) Les robots de Honda

(b) Les robots de Sony

(c) Les véhicules autonomes

FIGURE 2.7 – Exemples de robots de services

(a) Les robots araignées



(b) Les robots moléculaires

FIGURE 2.8 – Exemples de robots reconfigurables

2.4 Compréhension du Contexte pour l'Aide à l'Opérateur en Robotique

Alors que les robots ont d'abord été utilisés dans des tâches répétitives où tous les besoins des humains sont connus à priori, ils sont aujourd'hui impliqués dans des tâches et des activités de plus en plus complexes et moins structurées, dues aux

changements, parfois inattendus, de l'environnement. Cette fonctionnalité est classée dans l'intelligence du robot ou encore dans sa capacité d'autonomie comportementale. Malgré les travaux considérables menés depuis plusieurs années, ce problème essentiel reste encore d'actualité. Sa solution fait appel à trois exigences : (1) l'existence d'un système de perception des informations sur le contexte dans lequel se trouve le robot (informations pertinentes en rapport avec ce que doit faire le robot), (2) la connaissance préalable de cet environnement qui s'enrichit par apprentissage et enfin (3) la reconnaissance impliquant en général une comparaison entre les informations perçues et celles qu'il a déjà mémorisées. Cette dernière phase permet de conclure par une prise de décision d'action de déplacement adéquate à la progression vers le but recherché (qui doit donc être lui aussi connu ou identifié).

Dans cette section, nous présentons, dans un premier temps, la définition de l'autonomie des robots, ses complexités et les comportements que le robot doit avoir. Dans un deuxième temps, nous exposons les fonctions nécessaire à la compréhension du contexte pour augmenter l'autonomie d'un robot et donc aider l'opérateur en robotique.

2.4.1 Autonomie des robots

Au fil des années, les robots ont évolué pour effectuer, d'une manière la plus efficace possible, les tâches pour lesquelles ils sont conçus afin de répondre aux attentes de leurs opérateurs et satisfaire leurs besoins. C'est ainsi qu'il est possible de trouver des robots pouvant effectuer des tâches automatiques, mais certains sont aussi dotés d'une certaine intelligence [263]. L'autonomie d'un robot s'exprime par sa capacité, via son programme informatique, à appréhender l'environnement physique à partir des données reçues de ses différents capteurs. On parle alors de système robotique. Aujourd'hui, on cherche à réaliser des systèmes robotiques autonomes, capables de réagir seuls à l'environnement, leur octroyant une certaine capacité d'adaptation à un environnement inconnu. De nos jours, on note différents niveaux d'autonomie chez les robots. Ces derniers peuvent être :

- Des robots télécommandés qui, sans aucune autonomie, reçoivent les commandes fournies par l'utilisateur.
- Des robots semi-autonomes, où l'utilisateur intervient en cas de panne ou de situations non-prévues dans le programme informatique du robot. Dans ce cas, l'autonomie du robot vis à vis des obstacles qu'il est possible de rencontrer, peut être assurée lorsque l'inconnu reste relativement prévisible.
- Des robots totalement autonomes, où l'utilisateur n'intervient jamais. Dans ce cas, les facultés d'adaptation du robot exigent de bonnes qualités de perception dans le but d'interagir avec leur environnement.

2.4.1.1 Complexité de l'environnement

Comme le souligne Raja Chatila dans [43], l'autonomie est mise en œuvre lorsqu'un équilibre entre trois complexités est obtenu. Il s'agit de la complexité des mécanismes décisionnels, celle de l'environnement et celle de la tâche.
La complexité de l'environnement provient de sa dynamicité due à son changement en temps réel. Durant ces changements, le robot peut se trouver dans des situations de blocage. Ce type de situations sont les situations dans lesquelles le robot se trouve et qui bloque l'accomplissement des attentes de l'opérateur. Ces changements en temps réel ne sont, donc, pas résultants de l'exécution de la tâche par le robot et donc indépendant des décisions prises par le robot. De ce fait, la prise de décision du robot doit être en temps réel en ne disposant que de peu de temps pour comprendre, raisonner et calculer des possibilités de décision. La compréhension dynamique et automatique de l'environnement devient alors vitale.

2.4.1.2 Comportement du robot

Dans la littérature, généralement, la compréhension de l'environnement se fait à base de règles en liant la perception et l'action sur la base des changements prévus au préalable. Dans ce cas, nous parlons de réflexe (ou comportement réactif) qui permet au robot de réagir directement aux changements de l'environnement à travers des boucles sensori-motrices définissant des stratégies primitives. Le réflexe est utilisé en cas de situation connue élaborée par les stratégies primitives.
Dès lors que l'environnement devient beaucoup plus complexe, le robot ne trouve plus aucune règle qui soit associée au contexte.
En cas de situation de blocage, le robot doit planifier son comportement en prenant en compte cette situation afin de la résoudre et accomplir les attentes de l'opérateur. Le robot doit, donc, être doté de capacités de compréhension pour prendre les bonnes décisions. Nous parlons, alors, de comportement délibératif (ou comportement planifié).
Pour améliorer le comportement d'un robot face à l'accomplissement des tâches, une complémentarité entre le réflexe et le comportement planifié est nécessaire. En effet, pour obtenir un robot autonome en ayant des comportements optimaux, il est judicieux voire obligatoire de combiner ou associer les stratégies primitives aboutissant au réflexe avec des comportements planifiés, octroyant ainsi un comportement hybride au robot. L'un des challenges explicités dans le référentiel des projets de recherche européens CORDIS [1] et notamment sa partie relative aux systèmes cognitifs et robotiques, est de concevoir et développer des robots, entièrement autonomes ou en collaboration avec des opérateurs, capables de comprendre aussi bien leur environnement complexe et dynamique que les attentes de leurs opérateurs. Ces robots doivent élaborer des comportements robustes et raisonnables à des situations imprévues. La

1. CORDIS : "COmmunity Research and Development Information Service" : http ://cordis.europa.eu/fp7/ict/robotics/projects/areas-projects_en.html

capacité de compréhension est donc vitale pour les robots autonomes. Dans la section suivante nous présentons un tour de la littérature des travaux de recherches dans la compréhension des systèmes robotiques.

2.4.2 Compréhension du contexte

La capacité de compréhension est une capacité large et riche. Elle contient les fonctions telles que la reconnaissance, le raisonnement et interprétation, la planification et l'adaptation.

2.4.2.1 Reconnaissance (Recognition)

La reconnaissance est liée directement à la perception. La reconnaissance est la traduction de données issues des capteurs de perception en une information utile pour le système cognitif. La reconnaissance est effectuée dans trois niveaux :

- Niveau du signal : Analyse des signaux dans les boucles de contrôle ;
- Niveau de l'état : Analyse des caractéristiques de l'environnement, du robot et de leur lien avec les faits et les relations caractérisant l'état du monde ;
- Niveau de l'historique : des séquences ou trajectoires, des événements, des actions et des situations pertinentes pour la mission du robot.

La reconnaissance est habituellement utilisée dans les techniques de contrôle du robot. Les approches d'asservissement visuel pour le suivi ou la manipulation d'objets [251] ou encore la navigation jusqu'à sa destination offrent un excellent exemple de techniques matures qui peuvent être considérées comme étroitement intégrées dans les fonctions de base d'un robot [134]. De même pour les techniques de localisation et de cartographie simultanées. Ce domaine représente un des sujets les plus actifs et l'un des plus avancés en robotique. Ce qui est prouvé par la consécration de nombreuses publications dans ce domaine [67]. La reconnaissance utilisée en robotique est empruntée à d'autres domaines tels que les domaines de traitement du signal avec la reconnaissance des formes et l'analyse d'images, les domaines de l'analyse de l'émotion avec la reconnaissance de voix, les domaines de la psychologie avec les systèmes de reconnaissance de geste. Cela offre un historique et une évolution riches pour cette fonction. Cependant, l'intégration de ces techniques dans les exigences de l'autonomie reste difficile à mettre en œuvre. Le problème d'ancrage (Symbole Grounding) fournit une excellente illustration de la complexité de l'intégration des méthodes de reconnaissance de formes pour un robot autonome. Ce problème est dû à la création et au maintien d'une correspondance entre les symboles et les données des capteurs qui se réfèrent au même objet physique. Plusieurs fonctions de compréhension raisonnent sur les objets par attributs symboliques. Il est essentiel que la description symbolique et les données de perception soient en accord avec les objets auxquels ils font référence.

2.4.2.2 Raisonnement et Interprétation *(Reasoning)*

Le raisonnement sur l'objectif concerne la capacité d'un robot autonome de raisonner, formuler, sélectionner et gérer pour atteindre ses buts/objectifs. Le raisonnement diffère du cadre dans lequel le robot affiche les objectifs à atteindre, et de la décomposition des objectifs en sous-objectifs. Le raisonnement est la façon avec laquelle un robot décide, d'une manière dynamique et autonome, des tâches à faire pour atteindre ses objectifs. Ce sujet n'est pas nouveau, plusieurs chercheurs dans différents domaines mènent leur recherche sur le raisonnement. Les domaines d'études sont par exemple, les sciences cognitives, la planification automatique, l'Intelligence Artificielle ou encore la robotique.
Dans plusieurs implémentations, cette fonction est intégrée dans les fonctions de planification ou d'adaptation. Cette fonction a clairement des similitudes avec la fonction de supervision. Le raisonnement n'est pas apparenté à la planification, car il ne produit pas vraiment un plan, mais il établit de nouveaux objectifs et gére un existant, qui sont ensuite transmis au module de planification. Comme pour la fonction supervision, cette fonction vérifie en permanence les événements ou les situations inattendues. Ceux-ci sont analysés pour évaluer les objectifs actuels et, si nécessaire, établir de nouveaux objectifs. Certains systèmes ont un composant dédié à exercer cette fonction de haut niveau. Par exemple, l'Objectif Guidée par Autonomie ("Goal Directed by Autonomy "GDA") modélise et applique un raisonnement sur plusieurs objectifs et parfois des objectifs contradictoires d'un robot peuvent être considérés. Le raisonnement a été déployé dans un certain nombre d'expériences réelles, notamment dans l'expérience de l'agent Deep Space 1 (DS1) New Millennium [180] et dans le cadre du Continuous Planning and Execution Framework (CPEF) [181]. Pourtant, dans l'ensemble, cette fonction n'est pas souvent développée. Néanmoins, il est nécessaire pour les systèmes complexes et nécessitant une grande gestion des objectifs, à long terme, de prendre en compte la dynamique de l'environnement en déclenchant de nouveaux objectifs. En fournissant à un agent la capacité de raisonner sur les objectifs, ses mesures augmentent la performance des tâches. Les recherches sur le raisonnement ne sont pas à leurs débuts mais aucune recherche mature n'existe encore. Cette thématique semble avoir un bel avenir car la communauté de planification automatisée a expressément reconnu que le raisonnement a un rôle de premier plan parmi les agents intelligents qui agissent sur leurs propres plans, et il recueille une attention croissante des roboticiens et des chercheurs dans les systèmes cognitifs [107].

2.4.2.3 Planification *(Planing)*

La planification est une discipline appartenant à l'Intelligence Artificielle (IA). Cette discipline tend à concevoir des systèmes ayant la capacité de générer automatiquement, en s'appuyant sur un formalisme, des plans d'action permettant à un système d'accomplir un but, imposé par l'opérateur, à partir d'une base de connaissances contenant les actions possibles. Cette base de connaissances englobe

les préconditions (ce qui doit être vrai avant d'appliquer l'action) et les effets (ce qui arrive après). Ce résultat est sous la forme d'un système intégré de décisions appelé plan-solution. Ce plan d'action est principalement destiné à contrôler l'action d'un ou plusieurs agents exécuteurs (systèmes robotiques ou humains). Ces agents agissent dans un même monde pour atteindre le but tout en prenant des décisions adéquates aux situations successives rencontrées.
L'exécution du plan est la réalisation des actions par les robots. Elle vise à réaliser la suite d'actions du plan. Lorsque l'exécution est conforme à la réalité, cela permet (en l'absence d'événements imprévus et en présence d'une modélisation pertinente du monde) de faire évoluer le monde du robot de l'état initial à l'état final (le but).
Les problèmes de la discipline de planification sont des problèmes :

- De robustesse pour la prise en compte des états du monde partiellement connus ou d'actions non déterministes
- De complexité algorithmique pour la génération automatique des plans
- De contrôle d'exécution, de réaction face aux changements imprévus de l'environnement et de l'adaptation des plans déjà produits

Les premiers systèmes de planification existent depuis plus de quarante ans avec le système STanford Research Institute Problem Solver (STRIPS) [85], développé en 1971. Ce système a délimité le cadre classique de la planification. Ce cadre consiste en la prise en compte de l'aspect statique de l'environnement, l'observabilité totale, l'omniprésence de l'agent et les actions atomiques et déterministes. Et depuis, ce domaine a connu plusieurs évolutions se matérialisant en :

- Une extension des langages de représentation de domaine et de problème (STRIPS, ADL, PDDL, PDDL 2.1, PDDL+ [159]). Plusieurs benchmarks ont été mis au point
- Une formalisation de l'algorithmique de la planification constituée par les espaces d'états, les espaces de plans, les critères de vérité, GRAPHPLAN [255], les codages SAT [152], la planification CSP [22], la recherche locale
- Une augmentation des performances des algorithmes (complexité, complétude, décidabilité, optimalité) et sur de nombreux domaines de planification
- Une étude et exploitation des caractéristiques des problèmes et des domaines de planification pour améliorer la recherche de solutions (hiérarchisation, symétries, landmarks...)

Depuis 1987 avec le système TWEAK [45] de Chapman jusqu'en 1995 avec les formalisations de Kambhampati et Srivastava [234], l'approche fondée sur les stratégies de raffinements dans les espaces de plans partiels était la principale approche de recherche. Les autres approches existantes étaient marginalisées. En 1995, la conception du planificateur GRAPHPLAN a bouleversé cet ordre. Les idées de sa conception ont été les précurseurs à augmenter les performances des algorithmes de manière si importante que l'on peut maintenant commencer à envisager des applications réelles. Grâce à ce bouleversement, la planification par recherche heuristique dans les espaces d'états s'est élancée et a fait ses preuves en s'avérant être très performante avec

l'apparition de planificateurs comme HSP, HSPr [92], ALTALT [185], YAHSP [128]. Ces planificateurs utilisent des heuristiques inspirées de GRAPHPLAN et permettent actuellement de résoudre des problèmes non résolus avec les planificateurs d'il y a seulement quelques années. D'autres méthodes algorithmiques sont maintenant en concurrence comme la recherche heuristique dans les espaces d'états, dans les espaces de plans partiels, planification par satisfaction de bases de clauses (planification SAT), GRAPHPLAN, planification par satisfaction de contraintes pondérées (CSP), ou recherche locale. Les planificateurs issus de ces algorithmes sont évalués en menant des études comparatives régulières dans International Planning Competition (IPC) de la conférence ICAPS, et anciennement AIPS.
Les algorithmes récents de recherche heuristique dans les espaces d'états augmentent la performance, au niveau temporel, de résolution des problèmes compliqués et hors de portée de l'humain. Les plans d'actions que ces algorithmes génèrent peuvent comporter des milliers d'actions.
Les progrès remarquables élaborés ces dernières décennies ont permis la description dans le langage PDDL des domaines complexes proches des problèmes réels comprenant plus d'une dizaine d'opérateurs en prenant en compte des aspects temporels et de gestion des ressources (transport, satellites, gestion d'aéroports...). C'est pour cela que nous nous sommes basés sur ce langage au niveau de la planification dans ce travail de thèse.

2.4.2.4 Adaptation *(Acting)*

Contrairement à la planification (génération de plan d'action) définie comme une fonction prédictive en ligne, découplée de la complexité de la plate-forme d'exécution, l'adaptation est plus difficile à définir comme étant une fonction de compréhension.
La tâche de ce module n'est pas uniquement de déclencher les actions prévue dans le plan d'action, elle consiste aussi dans la gestion des capteurs bruyants et des modèles imparfaits. Ce module utilise des modèles de l'environnement non-déterministes, partiellement observables et dynamiques, traités par des commandes en boucle fermée. Pour intégrer ces exigences avec celles des modèles de planification, plusieurs formes de hiérarchisation sont explorées.
La génération de plan d'action se concentre sur les préconditions-effets et actions. Par contre, l'adaptation est raffinée d'une manière opportuniste en compétences "Skills" et une commande "command" de plus bas niveau. Ce mécanisme de raffinement peut également utiliser des techniques de planification, mais avec un espace d'état distinct et d'espace d'action différent de celui du planificateur.
La compétence "The Skill" dans laquelle une action est raffinée peut changer pendant l'exécution de l'action. Par exemple, plusieurs compétences de navigation peuvent offrir différentes possibilités de contrôle de localisation ou de mouvement adaptées aux différentes caractéristiques de l'environnement. Une action telle que moveTo(salon)

peut être raffinée en une séquence de différentes compétences de navigation par exemple. Ce schéma de hiérarchisation peut s'appuyer sur la représentation des connaissances, par exemple, les opérateurs de STRIPS combinés avec PRS [115] ou avec les procédures de RAP [86]. Dans certains cas, une représentation unique est utilisée pour la planification et l'action, par exemple le langage Golog [103]. D'autres approches utilisent une représentation unique vue à différents niveaux d'abstraction et raffiné de manière appropriée, comme dans les MDPs hiérarchiques [173]. Diverses techniques de calcul peuvent être utilisées pour concevoir un système d'action délibéré.
Dans la littérature, il existe cinq approches qui tiennent compte de la dynamique de l'environnement, à savoir, les approches fondées sur les procédures, les automates, la logique, l'analyse stochastique et le problème de contrainte de satisfaction CSP (Constraint Satisfaction Problem). Ces approches permettent aux robots d'agir, l'une des fonctions de délibération qui améliore et contrôle la réalisation du plan d'action.
Les approches fondées procédures, telles que le Système de Raisonnement Procédural (PRS) [115], Réactive Active Paquet (RAP), et XFRM [10], permettent au robot d'atteindre les objectifs. Cependant, elles n'ont pas été appliquées dans un contexte général. Ces approches supposent qu'il existe une procédure pour tous les objectifs que le système de surveillance doit traiter, et que les changements de l'environnement sont prévisibles. Le planificateur génère des plans d'action lorsque l'utilisateur fournit de nouvelles données. Cette approche ne résout que les situations de blocage prédéfinies.
Les approches utilisant les automates [19] et les approches basées sur la logique [103] ne réparent pas le plan d'action de départ. Lorsque le robot, muni d'un système comportant l'une de ces approches, se trouve dans une situation de blocage, il tente de trouver le bon raffinement de l'action dans une séquence de compétences afin de se débloquer. Cependant, ce robot est incapable de remplir sa mission quand il est confronté à une situation imprévisible et ne peut pas trouver un autre moyen pour atteindre sa destination.
Les approches basées sur l'analyse stochastique [205] observent les événements extérieurs de l'environnement afin de prédire par des méthodes non-déterministes les actions nécessaires pour atteindre l'objectif. Le plan d'action ne peut être réparé en ligne. Ces approches permettent aux robots d'agir en général, mais leur réaction pourrait être fatale à un robot s'il y a une erreur de calcul.
Les approches CSP (problème de satisfaction de contrainte) [94], comme IxTeT [137], RMPL, IDEA, T-REX, sont les approches les plus efficaces en termes de raffinement, d'instanciation, de gestion des temps et de réparation de plan. Ces approches nécessitent une connaissance approfondie préalable du contexte dans lequel le robot évolue. Dans toutes ces approches l'opérateur a un rôle crucial, qui est de fournir toutes les informations sur l'environnement et faire les interprétations.

2.5 Systèmes informatiques des robots

Les robots sont des structures mécatroniques qui interagissent avec l'environnement dans lequel ils évoluent. Un système informatique pour les robots, est un ensemble de logiciels embarqués œuvrant à son pilotage et à l'accomplissement de ses tâches, depuis la lecture des données capteurs jusqu'à la commande des actionneurs en passant par tous les calculs intermédiaires et connexes. Ces robots sont pilotés par des logiciels informatiques sous des contraintes temps réel. Avec ces logiciels, les robots acquièrent une autonomie de planification, de décision et d'accomplissement des tâches complexes. Ils acquièrent aussi la capacité d'interagir d'une manière non prévue ou d'une manière réactive face aux circonstances pour accomplir les différentes missions.
Les robots de nouvelle génération (robots de service ou d'intervention) couvrent des enjeux considérables. Dans tous les domaines, ces robots d'intervention (explorateurs, d'assistance, de défense, etc.) ont atteint un niveau de complexité considérable provenant notamment de l'intégration de multiples fonctionnalités : perception, planification, navigation, commande, autonomie du comportement, etc. Les travaux de recherche sur ces différentes thématiques ont certes permis d'accroître les performances et d'envisager la réalisation de missions toujours plus difficiles. Cependant, ces avancées restent locales au niveau des équipes qui travaillent indépendamment les unes des autres [2]. En plus des diverses fonctionnalités dotant les robots, tant pour leur autonomie opérationnelle que décisionnelle, une architecture de contrôle efficace et évolutive devrait être prévue pour ces nouveaux systèmes robotiques. En effet, les remarquables avancées technologiques, en termes de puissance de calcul, embarquées sur les robots et de missions plus délicates devant être exécutées avec plus d'autonomie et parfois aussi en coordination avec d'autres robots autonomes, ont impliqué une complexité de plus en plus croissante de l'architecture logicielle.

2.5.1 Architectures des systèmes informatiques des robots

L'autonomie suppose que l'architecture du système informatique intégré dans un robot prévoit la survenue des situations, puis la ou les réactions appropriées à celles-ci. Une bonne architecture pour un robot autonome doit prendre en compte la diversité des situations pouvant se présenter dans l'environnement réel. Les programmes utilisés dans les systèmes informatiques des robots autonomes les plus avancés font appels à l'intelligence artificielle ainsi qu'à des algorithmes fournissant au robot autonome une certaine forme d'apprentissage.
En plus des différents algorithmes nécessaires pour qu'un robot soit autonome tel que la perception, la planification, le contrôle, l'apprentissage et autre, c'est l'assemblage et le contrôle de ces algorithmes qui donne à un robot une réelle autonomie d'actions. Cet assemblage est souvent défini comme "l'architecture de contrôle" d'un robot.

2. http ://www.gdr-robotique.org

Dans la suite, nous présentons des différentes techniques de supervision ainsi que les architectures de contrôle des robots.

2.5.1.1 Supervision

La supervision est une fonction essentielle dans le système d'un robot. La supervision est considérée comme un module intermédiaire qui contrôle en ligne le comportement du robot tout en évitant les états incompatibles du système. La supervision s'assure du bon fonctionnement du robot et du contrôle de son comportement. Elle est, donc, chargée (1) de détecter la différence entre les prévisions (ce qui est planifié) et les observations, (2) de classifier ces différences, et (3) de récupérer dans le cas où la différence est importante ce qui induit à une anomalie. La fonction de supervision est distincte de la fonction de contrôle des actions mais, elles peuvent être mises en œuvre dans un même système, par exemple, dans le système Planex [77]. La fonction de diagnostic et de récupération ont un rôle crucial dans plusieurs applications. Ces fonctions sont mises en œuvre dans les FDIR (Fault Detection, Isolation and Recovery) [141] qui sont des systèmes complets de supervision. En effet, la supervision peut être spécifique, elle peut être concentrée sur le robot lui-même ou sur l'environnement.
Un tour de la littérature a été mené pour la fonction de supervision en robotique dans [203]. Dans cet article, trois approches ont été distinguées : les approches analytiques, les approches basées sur les données et les approches fondées sur les connaissances.

a Approches analytiques

L'approche analytique utilise un modèle mathématique fondé sur des principes de base de la physique (la dynamique des fluides), de la mécanique ou de l'électricité. Cette approche repose sur la notion de redondance analytique [243]. Cela signifie qu'un calcul de différence entre la mesure et l'estimation fournie par un modèle est élaboré. Cette différence résultante, appelée résidu, indique la présence ou non d'un défaut dans le système.
Cette approche est basée sur deux principales étapes, à savoir la génération résiduelle et la prise de décision. La génération résiduelle est un algorithme qui traite une entrée mesurable u(s) et une sortie du système y(s) afin de générer le signal résiduel r(s), où s désigne l'état du système donné dans l'espace de Laplace.
La prise de décision consiste dans la comparaison du résidu avec la probabilité d'apparition des états de panne et une règle de décision est utilisée pour déterminer si une erreur (panne) s'est produite. Ce processus de décision peut être basé sur, par exemple, un test de seuil simple, sur les valeurs instantanées des moyennes de la valeur résiduelle en mouvement, ou il peut s'agir de méthodes de la théorie statistique de la décision. Pour obtenir des informations d'erreur, le processus décisionnel doit également inclure l'isolation des pannes. Un seul signal résiduel est suffisant

pour détecter la présence d'un défaut, mais plusieurs signaux résiduels sont souvent nécessaires pour l'isolement de défaut.
Dans la littérature, trois approches différentes de génération de résidus peuvent être identifiées [88] qui sont l'estimation des paramètres, des relations de parité et des observateurs d'états. L'estimation des paramètres avec les approches fondées observateurs sont les deux méthodes les plus fréquemment utilisées pour la détection des erreurs, en particulier pour la détection et traitement des erreurs. Ces deux méthodes sont utilisées dans près de 70% de tous les travaux utilisant cette approche [117]. Ceci dit, cette méthode est très sensible aux bruits dans les signaux.

b Approche dirigée par les données

Contrairement à l'approche analytique, l'approche dirigée par les données ne repose pas sur des modèles mathématiques. Au lieu de cela, l'information utilisée pour la détection des erreurs et des pannes est dérivée directement à partir des données en entrée du système.
La prise de décision est souvent basée sur des théories statistiques. Les systèmes industriels modernes avec leur instrumentation lourde produisent une quantité exceptionnellement élevée de données. La force de cette approche est d'extraire à partir d'un grand volume de données l'information importante. Cela est fait par le calcul de mesures statistiques. Le principal inconvénient de cette approche est que le rendement est fortement tributaire de la quantité et la qualité des données d'entrée. Cette approche a été adoptée dans certains travaux de la robotique dans le domaine de la supervision des robots industriels [203].
L'application de la théorie statistique dans la supervision repose sur l'hypothèse que les caractéristiques des variations de données sont relativement inchangées, sauf si un défaut se produit dans le système. Cela implique que les propriétés des variations de données, telles que la moyenne et la variance, sont reproductibles pour les mêmes conditions de fonctionnement, bien que les valeurs réelles des données ne soient pas très prévisibles. L'approche dirigée par les données peut être divisée en deux groupes se référant au nombre de variables mesurées par le superviseur. Dans la supervision basée sur la statistique uni-variée, une seule variable est mesurée à la fois tandis qu'en multivariée, plusieurs variables différentes sont mesurées et combinées.

Supervision basée sur la statistique uni-variée Cette approche est utilisée pour déterminer les seuils de certaines variables d'observation. Ces variables d'observation sont des données d'entrée obtenues directement à partir des capteurs. Ce type de supervision est mis en œuvre dans Shewhart [172] afin de réduire le nombre de fausses alarmes et de détections manquées. Les seuils sont déterminés à partir des observations des taux des fausses alarmes et des taux de détection manquées. Compte tenu des valeurs de seuil, la théorie de l'hypothèse statistique est appliquée pour prédire la fausse alarme et les détections manquées sur la base des statistiques

des données dans un ensemble d'apprentissage. Un des premiers exemples de détection des pannes en utilisant la valeur limite de vérification sur un robot autonome est le travail sur "Thinking Cap" [222]. "Thinking Cap" est une architecture de contrôle de robot dans laquelle, il y a eu une intégration du système de détection de pannes qui est basé sur la logique floue. Dans ce système, une variable, qui est une composition floue de plusieurs variables, est comparée à un seuil donné, afin de détecter les plans défectueux. Le travail décrit dans [194] est un prolongement, du travail décrit précédemment, par la comparaison de la variable floue à plusieurs limites, afin d'obtenir la grandeur de l'erreur. Plus récemment, une autre approche uni variée de contrôle de valeur limite a été appliquée à un robot mobile [127]. Ici, la variable d'entrée est le courant du moteur. La valeur limite est dérivée à partir de données empiriques lorsque le robot mobile fonctionne dans des conditions normales.

Supervision basée sur la statistique multivariée Cette approche considère, par rapport à la statistique uni-variée, la corrélation entre les variables. Cela donne un outil beaucoup plus puissant à la fois pour la détection d'erreurs et des pannes. La qualité et la quantité des données ont une grande influence sur la performance des méthodes statistiques dirigées sur les données.

Le principale inconvénient de cette approche est qu'un traitement sur les données est effectué sans arrêt pour trouver l'erreur ou la panne. Ce qui est coûteux en terme de temps de calcul et de mémoire.

c Approche fondée sur les connaissances

L'approche fondée sur les connaissances offre la possibilité de combiner l'approche analytique et celle basée sur les données dans un système de supervision. Les approches fondées sur la connaissance peuvent être divisées en trois méthodes : l'analyse causale, les systèmes experts, et les réseaux de neurones artificiels.

Analyse causale C'est une méthode d'analyse basée sur la modélisation causale des relations "symptômes" "pannes". Cette méthode est principalement utilisée dans l'isolation des pannes. Elle est réalisée à l'aide du graphe "Signed Directed Graph" (SDG) [44]. Le SDG est une carte montrant la relation entre les variables du système. Cela reflète le comportement de l'équipement en cause ainsi que la topologie du système général.
Les nœuds représentent les variables du système, des capteurs, des pannes possibles du système, ou des pannes de composants. Quand un SDG est utilisé pour l'isolation des pannes, des seuils minimaux et maximaux pour chaque variable doivent d'abord être définis. Cette méthode ne traite pas la résolution des pannes qui est le but majeur de la supervision.

Systèmes experts Ils sont utilisés pour imiter le raisonnement des opérateurs lors de l'isolation des défauts. De nombreuses applications de détection des pannes dans les domaines de l'ingénierie ont fait usage de systèmes experts [118]. Les connaissances dans le système expert sont souvent formulées en termes de règles SI-ALORS, qui peuvent être trouvées à partir des principes de base du système ou à partir d'une description structurelle du système pour détecter et récupérer les pannes. La sortie des systèmes experts est en binaire soit vrai soit faux. Cela rend le système sensible aux incertitudes.

Réseaux de Neurones Artificiels (RNA) Ils ont été motivés par l'étude du cerveau humain, qui est constitué de millions de neurones interconnectés [201]. Les RNA sont plus utiles quand il est difficile, voire impossible, de créer des modèles mathématiques du système étudié. Plusieurs recherches ont appliqué RNA pour la détection et la récupération des pannes.
Dans les travaux présentés dans [262], les chercheurs notent qu'une combinaison de plusieurs réseaux de neurones peut faire mieux qu'un seul réseau. Dans les travaux [109] une application a été élaboré dans le cadre des RNA dans une architecture de contrôle de robot pour l'extraction de faits à partir de l'évolution du robot dans son environnement. Ce réseau neuronal récurrent est utilisé pour l'apprentissage. Cette méthode est mise en évidence sur les données d'un simulateur de robot mobile [109]. Le principal inconvénient est que leur efficacité est très corrélée avec la quantité et la qualité des données d'entrée.

2.5.1.2 Architectures de contrôle

Une architecture de contrôle est en charge de la configuration, de l'ordonnancement, du déclenchement et du contrôle de l'exécution des différents algorithmes. Une telle architecture doit être conçue pour doter le robot d'(i) une capacité de réaliser une grande variété de tâches complexes et de haut niveau, sans être configuré manuellement, et (ii) une capacité à affronter un ensemble d'événements inconnus à priori, dans un environnement imprévisible. Ces deux capacités sont les caractéristiques primordiales pour un robot afin qu'il soit autonome.
Comme vu dans la section 2.4, il existe trois genres d'autonomie en robotique : l'autonomie se basant sur des comportements réactifs, l'autonomie se basant sur des comportements délibératifs et l'autonomie se basant sur des comportements hybrides. Les architectures des systèmes informatiques des robots se sont basées sur cette distinction. On retrouve le comportement réactif ou réflexe dans l'architecture de contrôle réactive. Ce type d'architecture offre des boucles sensori-motrices effectuant les stratégies primitives. Les capacités de compréhension offrent un comportement délibératif. Ce comportement est issu des robots ayant une architecture de contrôle délibérative. Pour augmenter l'autonomie d'un robot, on utilise l'architecture hybride pour faire collaborer le réflexe avec les fonctions de compréhension.

Toutes les architectures, ou du moins les plus évoluées, embarquent des composants réactifs et des composants décisionnels/délibératifs.
Dans la suite, nous décrivons les différentes architectures informatiques des robots.

a Architecture réactive : *Don't think (re) act*

La manière la plus simple de concevoir un système informatique pour un robot est de le doter d'un ensemble de règles stimulus-réponse (perception-action comme le montre la Fig. 2.9). Cela correspond à un comportement réactif comme décrit dans la section 2.4. Un robot se basant sur une telle approche est nommé un système réactif

FIGURE 2.9 – Principe de l'architecture réactive

ou robot réactif. La réactivité se juge alors par la capacité du robot à associer les actions qu'il peut exécuter en réaction à ce qu'il perçoit. Ce type d'architecture est dû à Brooks du Massachusetts Institute of Technology (MIT) avec l'architecture de subsumption [28]. L'architecture de Brooks (Fig. 2.10) telle que définie dans [28] est

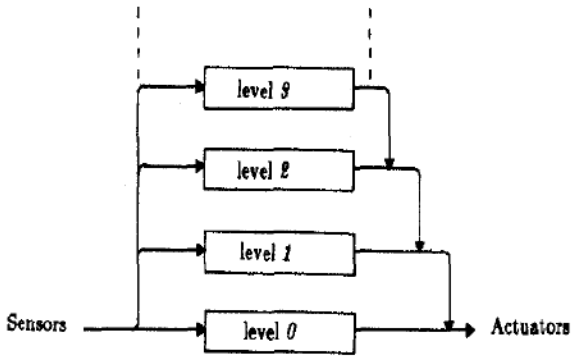


FIGURE 2.10 – Architecture de brooks [28]

composée de modules simples "comportements" et d'un module nommé "arbitration" ou module arbitrage.
Chaque module de "comportement" se limite à une seule fonctionnalité de contrôle du robot. Ces modules sont indépendants et exécutés parallèlement à une fréquence précise. A chaque itération, chacun des modules "comportements" calcule les com-

mandes à faire et les envoie au module d'arbitrage. Ce module fusionne alors l'ensemble de ces commandes et effectue le calcul des commandes finales qui contrôlent le robot. Les robots réactifs ne sont pas capables d'effectuer des tâches structurées, cela est dû à l'absence de processus délibératif.
En effet, les tâches sophistiquées requièrent la capacité de prédiction des effets de leurs actions afin de déclencher les comportements adéquats pour la finalisation de leurs activités. Ces comportements doivent être planifiés et requièrent des capacités d'analyse, de compréhension et de raisonnement [150].

b Architecture deliberative : *Think Hard, Act Later*

L'architecture délibérative a été imposée afin de résoudre le problème de prise en compte de la complexité de l'environnement. Cette architecture permet à un robot d'intégrer les capacités d'analyse, de compréhension, de raisonnement et de prise de décision. Un robot se basant sur une telle approche est nommé un système délibératif ou robot délibératif.
Un système délibératif possède un modèle du monde ainsi qu'un modèle d'actions. Le modèle du monde permet aux systèmes robotiques la prédiction de l'évolution de l'environnement. Le modèle des actions, quant à lui, permet de prédire l'effet des actions du robot sur l'environnement. Ces deux modèles agissent sur des représentations internes de l'environnement. Le principe de cette approche est décrit dans la Fig. 2.11. Il se résume en la perception, la compréhension et l'action. Un système délibératif est capable de percevoir son environnement en mettant à jour le modèle de cet environnement pour pouvoir prédire son évolution. Ce système planifie ses actions dans la perspective de satisfaire ses objectifs en agissant conformément aux plans d'action du robot. Un système délibératif manipule des structures de données de façon dynamique

FIGURE 2.11 – Principe de l'architecture délibérative

dans la mémoire. Cette propriété est la force de cette approche. Cela peut avoir de mauvaises conséquences, car si cette architecture permet de résoudre des problèmes formalisés, elle est coûteuse en temps et mémoire. Nous pouvons obtenir facilement un système nécessitant des heures de réflexion ou encore un système n'ayant pas assez de mémoire de stockage pour mémoriser les connaissances [91][90].

c Architecture hybride : *Think and act independently*

Les inconvénients des deux architectures précédentes justifient l'apparition et l'émergence de l'architecture hybride. En effet, les systèmes réactifs agissent uniquement en suivant des stratégies primitives préprogrammées et les systèmes délibératifs sont coûteux en temps et mémoire. L'architecture hybride combine les avantages des architectures délibératives et réactives et pallie leurs inconvénients. Elle est décomposée en deux niveaux, comme cela est schématisé dans la Fig. 2.12 : niveau réactif ("Perception", "Action") et niveau délibératif en passant par la compréhension. Le choix est fait au moyen du contrôle d'exécution. L'exemple d'une architecture hybride Fig.

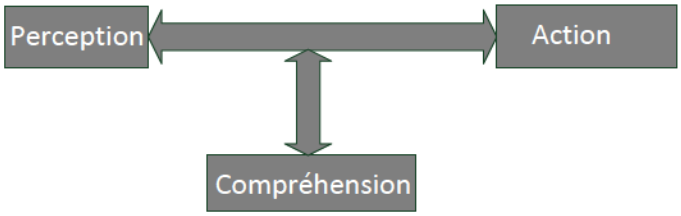


FIGURE 2.12 – Principe de l'architecture hybride

2.13 illustre trois couches. La couche décisionnelle s'occupe des capacités délibératives du robot, la couche de contrôle d'exécution (comme présenté dans 2.5.1.1) vérifie les requêtes envoyées aux modules fonctionnels et l'utilisation des ressources et la couche fonctionnelle qui s'occupe de la perception du contexte. L'utilité de cette architecture n'est pas uniquement de pallier les inconvénients des autres architectures, elle permet aussi de faire l'ordonnancement des tâches entre plusieurs robots afin d'atteindre un but/objectif commun issu de la collaboration entre ces robots [4][250].

2.5.2 Plateformes d'intégration logicielles existantes

Des plateformes informatiques sont nées du besoin de faire l'intégration de tous ces composants logiciels, tel que décrit précédemment, dans un robot. Cela est illustré dans la Fig. 2.14 extraite de [32]. Cette figure montre les différents composants à intégrer dans un système robotique. Des laboratoires ainsi que des industriels se sont intéressés à l'élaboration et la mise en place de plateformes d'intégration. Un des objectifs de la communauté robotique cherche la standardisation des plateformes afin de favoriser l'interopérabilité. Dans la suite, nous présentons les plateformes les plus utilisées et qui ont fait leur preuve dans plusieurs systèmes robotiques.

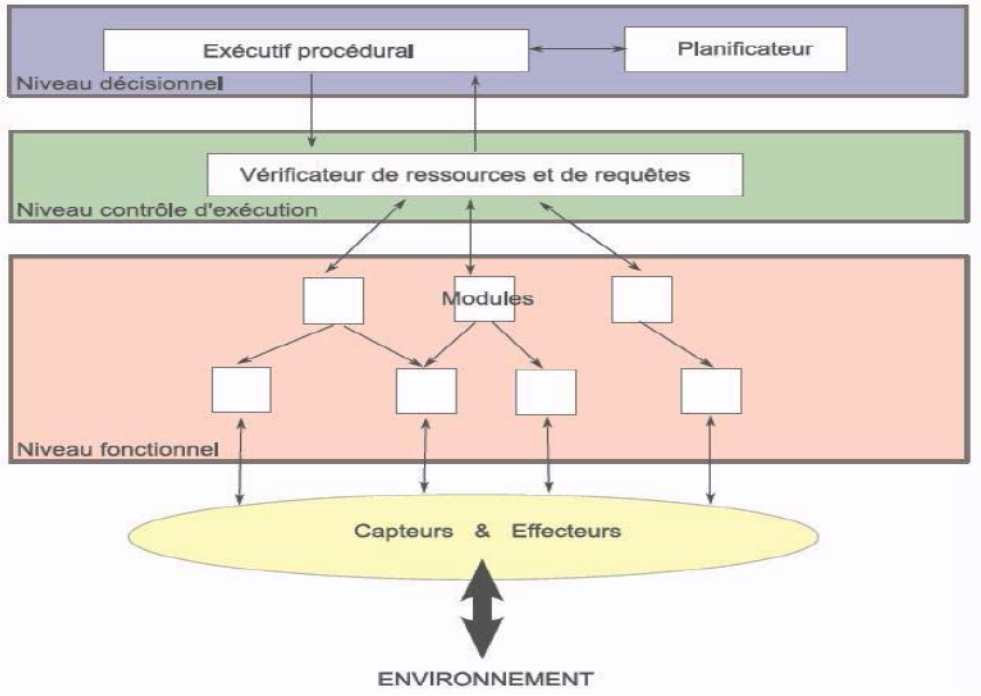


FIGURE 2.13 – Exemple d'une architecture hybride

2.5.2.1 ROS : Robot Operating System

ROS [210] est une plateforme Open Source. C'est un ensemble de logiciels développés par Willow Garage. Elle est basée sur le middleware ROSComm et sur le simulateur Gazebo. ROS contient des pilotes pour les capteurs et les actionneurs. Elle possède aussi des piles applicatives telles que la navigation (e.g MoveIt), manipulation (e.g. GraspIt), etc... ROS s'appuie aussi sur des bibliothèques fonctionnelles élaborées par des start-ups. ROS est une plateforme qui peut tourner sur plusieurs systèmes d'exploitations (Linux, Windows, Android, arduino,...). La communication avec ROS peut être faite avec C++, Java, Python ou autres. Cette communication utilise le service RPC (Remote Procedure Call) qui gère les liaisons entre clients et serveur sur des ordinateurs distincts. Il existe une version industrielle de cette plateforme.

2.5.2.2 OROCOS : Open RObot COntrol Software

OROCOS [30] est une plateforme open source. Le développement de OROCOS a été financé par la Communauté Économique Européenne (CEE).
Au départ, l'équipe était composée de quatre laboratoires de recherche : l'Université de Leuven, le KTH en Suède, le LAAS en France et l'université de l'ULM en Allemagne.

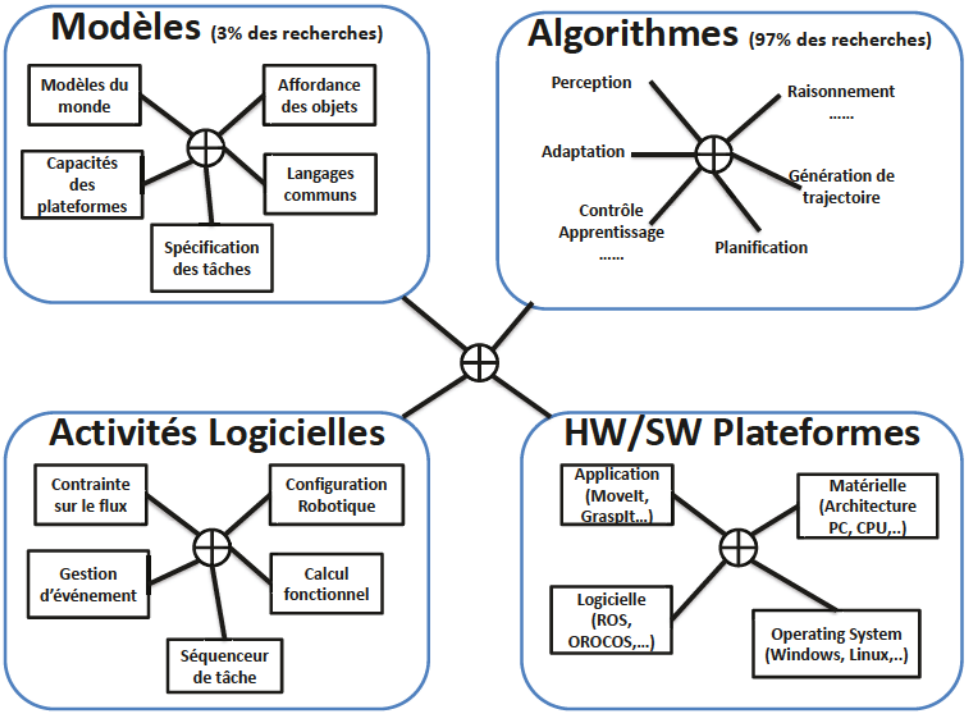


FIGURE 2.14 – Intégration d'une application robotique [32]

Le développement a été dirigé par le professeur Herman Bruyninckx. OROCOS a été créée comme un standard qui fournit des outils de développement de contrôleurs adaptés à la recherche académique et à l'industrie. OROCOS est maintenant parfaitement mis à disponibilité sous licence LGPL.

2.5.2.3 YARP : Yet Another Robot Platform

YARP [165] est une plateforme open-source sous licence LGPL. Elle est écrite en C++ pour interconnecter les capteurs, les transformateurs et les actionneurs du robot. Plus précisément, YARP permet de construire d'une manière peer-to-peer un système de commande de robot selon les besoins. Ceci est fait comme une collection de programmes communiquant dont le type de connexion peut être TCP, UDP, multicast, XML/RPC. Yarp est compatible avec Windows, Linux et Mac OSX.

2.5.2.4 MRPT : Mobile Robot Plateform Toolkit

MRPT [49] est une multiplateforme open source sous la licence LGPL écrite en C++ qui vise à aider les chercheurs en robotique pour la conception et la mise en œuvre d'algorithmes liés à Simultaneous Localization And Mapping (SLAM), la vision par ordinateur et la planification des mouvements (l'évitement d'obstacles).

2.5.2.5 Urbi : Universal Real-Time Behavior Interface

Urbi [7] est une plate-forme logicielle open-source pour contrôler des robots. Cette plate-forme comprend une bibliothèque C++ de composants appelés "UObject". Elle est disponible en API standard pour décrire les moteurs, les capteurs et les algorithmes. Cette plateforme offre l'utilisation du langage de script d'orchestration "urbiscript" pour intégrer l'ensemble des composants et décrire les comportements que le robot peut faire, comme python ou LUA sauf qu'elle utilise la programmation événementielle pour rendre le travail plus facile. Le but d'Urbi est d'aider à la construction des robots compatibles et à la simplification des programmes et des comportements des robots. La gamme d'applications potentielles de Urbi va au-delà de la robotique, car il a été utilisé avec succès dans les systèmes complexes génériques et dans la collaboration entre plusieurs agents.

2.6 Synthèse

Dans ce premier chapitre de l'état de l'art et après un bref historique présentant l'évolution observée en robotique, nous avons donné un aperçu sur les différentes catégories des robots et leur utilisation dans les applications réelles. Cette étude a montré que les robots sont de nos jours impliqués dans des tâches et des activités de plus en plus complexes et moins structurées, notamment lorsqu'ils sont livrés à eux-mêmes dans un environnement changeant et imprévisible. Leur capacité d'autonomie comportementale exige de disposer, dans leur système informatique et en plus des fonctions réactives de bas niveau et de supervision, de fonctions délibératives opérant sur la connaissance préalablement connue et pouvant être enrichie de l'environnement. Les chercheurs et les ingénieurs en robotique pensent qu'il est nécessaire de disposer d'un système intégré (hardware/Software) permettant d'intégrer, dans un même cadre, les activités logicielles, les algorithmes et les modèles rendant un robot capable d'évoluer dans un environnement dynamique en agissant face à des imprévus tout en maintenant son objectif, ce qui améliore son autonomie [32].
Aujourd'hui, plusieurs équipes travaillent sur l'amélioration de l'autonomie des robots. Les premiers travaux, qui se sont intéressés à la détection et la résolution des collisions dans un environnement dynamique, se basent généralement sur la modélisation mathématique et ne permettent pas au robot de faire de l'actualisation de plans d'actions [149][35]. Or, au cours de son déplacement, le robot ne rencontre pas, sur son chemin, que des obstacles à contourner, mais aussi des problèmes à résoudre. Dans

ce cadre, certains travaux de recherche, parmi les plus récents, sont tournés vers les applications émergentes de la robotique et des systèmes intelligents dans différents domaines, notamment ceux liés aux neurosciences [253] et aux sciences cognitives [11]. Parmi les recherches menées, nous pouvons citer celles qui portent sur la modélisation, l'analyse et la conception de systèmes dynamiques et de systèmes de perception [265][5][164]. Elles donnent lieu à des travaux fondamentaux associés le plus souvent à des développements expérimentaux. Cette pratique vise à garantir la pertinence des propositions formulées. Elle permet aussi d'avancer simultanément dans les connaissances relatives aux problèmes scientifiques traités et à leurs applications. Les capacités d'interactivité et d'autonomie constituent des dimensions essentielles dans les systèmes robotiques. Elles sont à la base des développements des systèmes innovants dans des domaines délicats tels que les interventions chirurgicales, la suppléance fonctionnelle, la rééducation, la manipulation d'objets et les aides pour diverses formes de handicap.

Ces nouveaux systèmes reposent sur des architectures complexes qui intègrent des systèmes mécaniques et électroniques ainsi que les logiciels associés, mais aussi des contenus adaptés à la réalisation des fonctions pour lesquelles ils sont destinés. Ces systèmes que l'on veut « intelligents » devraient disposer de grandes capacités cognitives, notamment s'ils agissent dans un environnement naturel, donc dynamique, afin de ne pas provoquer de désastre.

C'est dans ce cadre que se situent nos travaux de recherche menés dans cette thèse. Ils visent l'amélioration de l'autonomie d'un robot en adoptant une approche fondée sur les connaissances et présentant une architecture hybride facilement intégrable dans un système robotique. Cette approche favorise la capacité de compréhension au moyen d'un raisonnement intelligent. Dans le chapitre suivant, l'intérêt est porté à ces systèmes robotiques qualifiés d'intelligents.

Chapitre 3

SYSTÈMES ROBOTIQUES INTELLIGENTS

Sommaire

3.1 Introduction

Les prochaines générations des systèmes robotiques offriront des systèmes avec plus d'intelligence [53]. Cela ouvrira des opportunités et des possibilités d'applications robotiques dans plusieurs secteurs. Ces systèmes doivent être faciles à utiliser et à comprendre de façon dynamique le contexte. Pour favoriser cela, ces systèmes robotiques doivent être capables de reconnaître, d'analyser, de raisonner et de décider. Ces capacités devraient les aider à s'adapter à des situations changeantes ainsi qu'à des besoins et des préférences variables de l'opérateur et à optimiser les tâches en les mettant en œuvre d'une manière intelligente. C'est dans ce cadre que nos travaux de recherche s'inscrivent.
D'après les principales organisations de recherche et développement telles que l'agence du Département de la Défense des Etats Unis, DARPA [1], chargée de la recherche et développement des nouvelles technologies destinées à un usage militaire, l'association japonaise JRA [2] ou encore le référentiel de l'information sur les projets de recherche et développement européens CORDIS [3], les projets de recherches dans les systèmes cognitifs et robotiques sont classés dans différentes catégories selon les domaines de recherche et les domaines d'application.
Les principaux domaines de recherche de cette catégorisation sont la perception, la compréhension et l'action. Les domaines d'application se subdivisent en applications : aériennes, sous-marines, industrielles et fabrications, de services professionnels et domestiques, médicales et de réadaptation ainsi que de suivi et de supervision. Comme vu dans la section 2.4, les comportements planifiés se réfèrent à des actions résolues, choisies ou envisagées, menées en vue d'atteindre certains objectifs. De nos jours, généralement les applications des systèmes cognitifs et robotiques sont limitées à une tâche unique et évoluent dans un environnement parfaitement connu, ne nécessitant pas de capacités de compréhension. La compréhension englobe les capacités critiques pour qu'un robot puisse faire face aux changements de l'environnement en agissant d'une manière autonome. Les limites de la capacité de compréhension du robot dépendent des modélisations et des architectures spécifiques.

1. Defense Advanced Research Projects Agence
2. Japaneese Robotic Association
3. COmmunity Research and Development Information Service

Les fonctions, offrant la capacité de compréhension distinguées dans 2.4, sont la reconnaissance, le raisonnement et interprétation, la planification et l'adaptation. Ceci laisse entendre que le robot doit mémoriser des connaissances qui lui permettent de faire un certain raisonnement pour décider de l'action à entreprendre.
Ce chapitre dresse un panorama des architectures intégrant les fonctions de compréhension puis les modèles de représentation des connaissances et les types de raisonnement utilisés dans les systèmes cognitifs et robotiques.

3.2 Architectures d'intégration pour la compréhension

Les systèmes informatiques déployés sur les plateformes robotiques comprennent, entre autres les logiciels, le module d'exécution et de contrôle qui assure la configuration, l'assemblage et l'intégration des fonctionnalités du système. Ce module est déterminant pour l'autonomie du robot puisqu'il intégre la compréhension. Son architecture dépend donc du comportement qu'aura le robot. Elle sera réactive si le robot est réactif, délibérative si le robot est délibératif et hybride si le robot est autonome avec des capacités de contrôle d'exécution et de compréhension lui permettant d'effectuer des tâches avec succès dans des environnements complexes et dynamiques. Dans cette section, nous présentons les architectures de la littérature, intégrant la capacité de compréhension pour les robots comportant des mécanismes de gestion d'objectif de l'opérateur.
Les travaux qui n'ont pas été inclus dans l'étude sont représentatifs d'approches similaires.

3.2.1 MBA : Motivated Behavior Architecture

L'architecture MBA, développée par l'Université de Sherbrooke [54], offre différentes capacités au robot, à savoir, la perception, les algorithmes de navigation, les algorithmes de planification ainsi que les algorithmes de raisonnement. Le but de cette distinction est de simplifier la programmation des comportements du robot dans l'accomplissement de diverses tâches.
L'architecture MBA est composée de trois principaux modules comme l'indique la Fig. 3.1 :

1. Behavior-Producing Module (BPM) définit les capteurs de perception ainsi que les conditions influençant le contrôle des effecteurs. Cet élément est le générateur de comportement réactif. L'utilisation effective d'un BPM est déterminée par un module d'arbitrage appelé **"BPM Arbitration"** qui est basé sur la priorité, la fusion de données et la sélection de l'action. L'activation du BPM est faite selon un module appelé **"BPM Selection"**. La correspondance entre les tâches et BPM est faite par les paramètres (p) et les résultats du comportement (res) sont gérés par un module appelé SNOW.

2. Motivations module (ou "Sources de motivation") est le module qui recommande l'utilisation ou l'inhibition des tâches à accomplir par le robot pour atteindre son objectif. Ce module contient trois principales classes de motivation : instinctive, rationnelle ou émotionnelle. Les motivations instinctives permettent le fonctionnement de base du robot en utilisant des règles d'inférence simples. Les motivations rationnelles sont plus liées à des processus cognitifs, comme la navigation et la planification. Les motivations émotionnelles surveillent les situations conflictuelles ou de transition entre les tâches. Ces motivations suivent l'évolution des engagements du robot établis avec d'autres agents, humains ou des robots de son environnement. Ce module s'occupe aussi de la supervision (définition dans la sous section 2.5.1.1). Un mécanisme d'arbitrage est utilisé pour coordonner les comportements du robot.
3. Dynamic Task Workspace module(DTW) organise les tâches dans une structure arborescente en fonction de leurs interdépendances et leur niveau d'abstraction. Par exemple, "ramener un objet de la cuisine" se décompose en plusieurs tâches primitives comme attraper l'objet ou éviter l'obstacle. Le DTW échange des informations de manière asynchrone avec le BPM sur la façon d'activer, configurer et surveiller le robot. Des tâches peuvent être ajoutées, modifiées en soumettant des requêtes (m), les requêtes (q) et souscrire les événements (e) liés à la tâche. Le DTW émet également des recommandations de comportement (rec) concernant les tâches : ces recommandations peuvent être positives, négatives ou neutres, sur l'opportunité de permettre au robot d'accomplir des tâches spécifiques.

Cette architecture a été mise en place sur le robot Spartacus [166]. Ce robot été la star de la conférence AAAI'05 [54]. L'utilisation d'un arbre pour représenter les tâches implique la possibilité d'obtenir des redondances dans les différents niveaux d'abstraction dans l'arbre, ce qui n'est pas optimal. En plus des difficultés d'anticipation et de réorganisation des stratégies comportementales reconnues par les concepteurs de MBA, nous constatons un manque d'information concernant des capacités de raisonnement et d'apprentissage afin de gérer les changements dynamiques dans des contextes de la vie réelle.

3.2.2 DIARC : A Distributed Integrated Affect Reflection Cognition Architecture

D'après les concepteurs de DIARC [226], cette architecture développée à l'université de Notre Dame (Indiana) offre plusieurs fonctionnalités. Elle est basée sur des connaissances complètes qui peuvent être utilisées dans les Interactions Homme-Robots sans aucune modification structurelle. Les connaissances sur les tâches et les comportements appropriés pour accomplir le désir de l'opérateur peuvent être représentées à la fois par des connaissances déclaratives et procédurales, exprimées sous la forme de scripts contenant des informations sur des séquences d'actions,

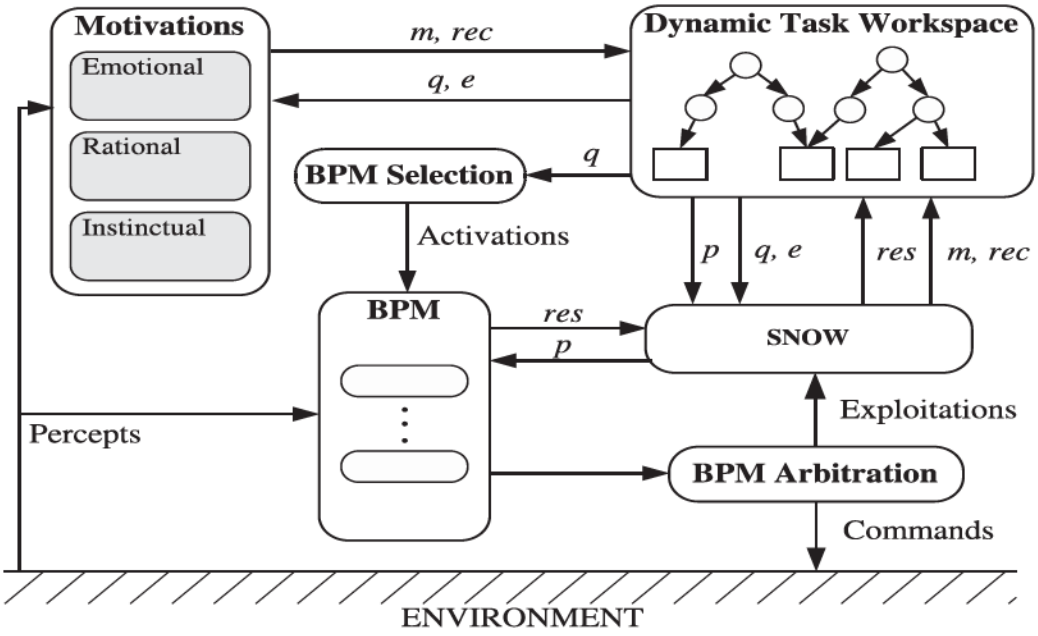


FIGURE 3.1 – Architecture de MBA [9]

des événements et des objectifs associés à leurs résultats. En outre, DIARC est construite sur l'infrastructure multi-agent (Agent Development Environment "ADE") qui considère que le robot est un agent dans un Système Multi-Agents (SMA), permettant ainsi la distribution de composants architecturaux sur plusieurs agents en offrant un appui pour la détection automatique des pannes (Supervision) et pour une récupération ultérieure de l'erreur. L'utilité de DIARC pour l'Interaction Homme-Robot a été évaluée dans une variété d'expériences, à la fois qualitativement et quantitativement, avec des scénarios dans un environnement réel (c'est à dire, des expériences avec des humains et des interactions naturelles à travers les compétitions de robots AAAI) [227]. La Fig 3.2 représente une vue de l'architecture de DIARC mise en œuvre dans ADE. Le niveau supérieur montre des parties de la décomposition fonctionnelle de DIARC en termes de perception sensorielle, d'action, et d'effecteurs. Le niveau du milieu montre la correspondance entre les composants fonctionnels de haut niveau et leur mise en œuvre dans le cadre ADE. A ce niveau, nous trouvons des composants informatiques autonomes qui fonctionnent en parallèle et communiquent via plusieurs types de liens de communication. Le niveau le plus bas représente le matériel dans le système. Bien que cette architecture offre à un robot les fonctions de reconnaissance et de gestion d'objectifs, elle ne le lui permet pas d'interpréter et de raisonner afin de réagir d'une manière intelligente face à la complexité d'un environnement réel.

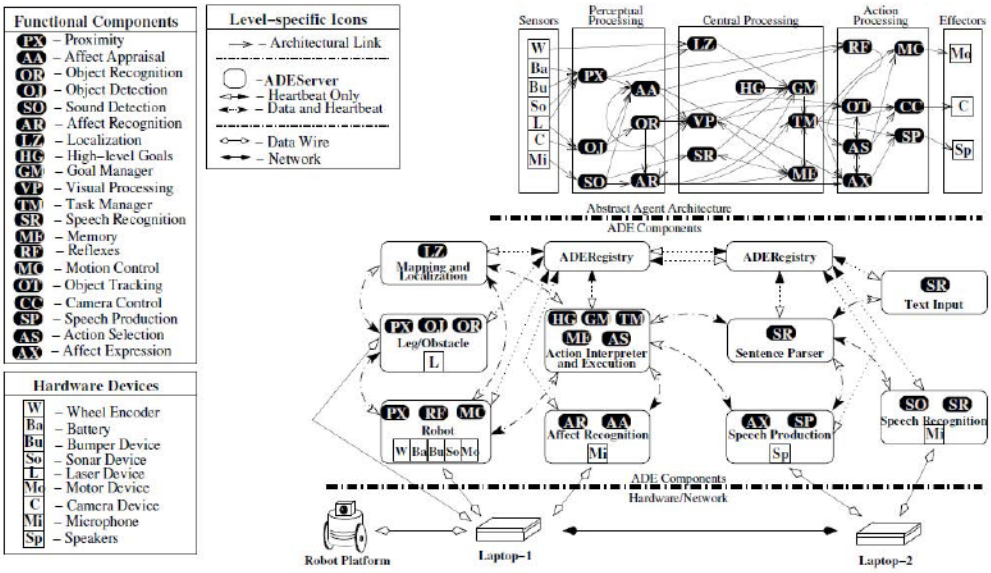


FIGURE 3.2 – Architecture de DIARC [226]

3.2.3 MADbot : A Motivated And Goal Directed Robot

Le projet de MADbot, conduit à l'université of Strathclyde, concerne le développement d'un robot autonome, capable de générer ses propres objectifs comme cela est effectué dans [50] et de superviser son propre comportement dans l'exécution des tâches.
Le contrôle de l'exécution se fait par rapport à des modèles stochastiques sur les effets des actions que le robot peut exécuter. A tout moment, durant l'exécution de son plan, le robot peut estimer son état le plus probable sur la base du modèle approprié et les observations actuelles du système. C'est à dire que s'il estime que l'état le plus probable est un état de panne, il planifie pour atteindre les objectifs en suspens. Le comportement du système peut changer afin de gérer des objectifs générés en réponse à des situations observées. Ceux-ci se produisent lorsque :

- Le plan en cours d'exécution quitte les ressources non utilisées, ce qui les rend disponibles pour les activités précédemment imprévues.
- Les changements dans l'environnement durant l'exécution causent des changements de valeurs des caractéristiques du système pour modifier la valeur, de sorte que certains dépassent les seuils acceptables.
- Le plan d'actions subit une panne, nécessitant une reconsidération des priorités de ses objectifs en suspens.

La Fig. 3.3 représente l'architecture de MADbot. Dans cette figure, les rectangles représentent les composants de l'architecture. Les ovales représentent des types de données transmises entre les différents composants dans les directions indiquées par les flèches. La ligne en pointillés et ovales indique une instruction "Halt" communiquée

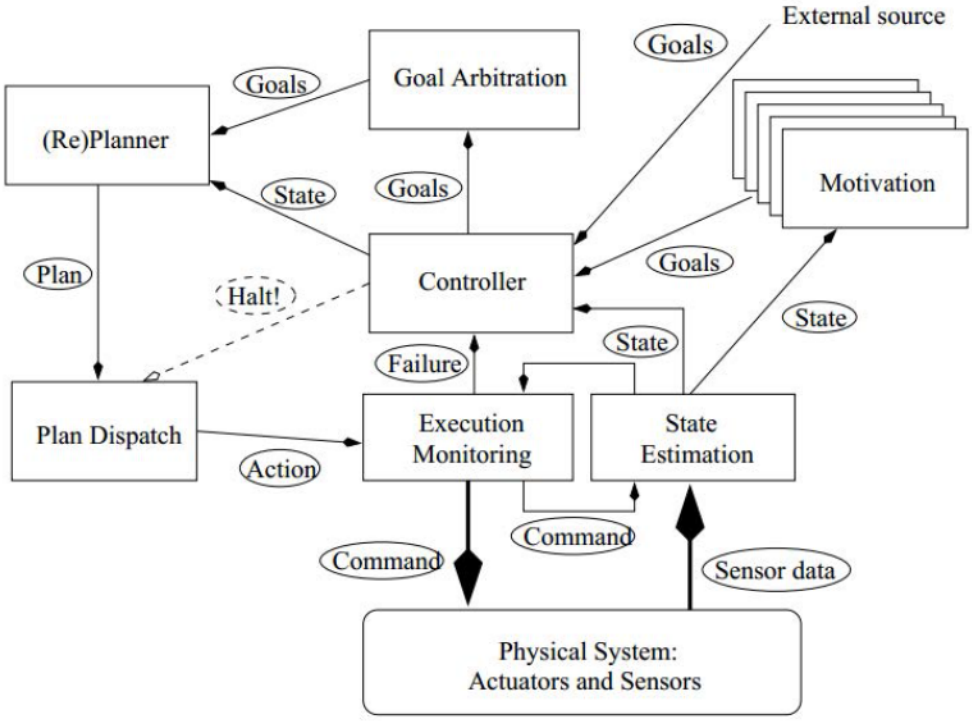


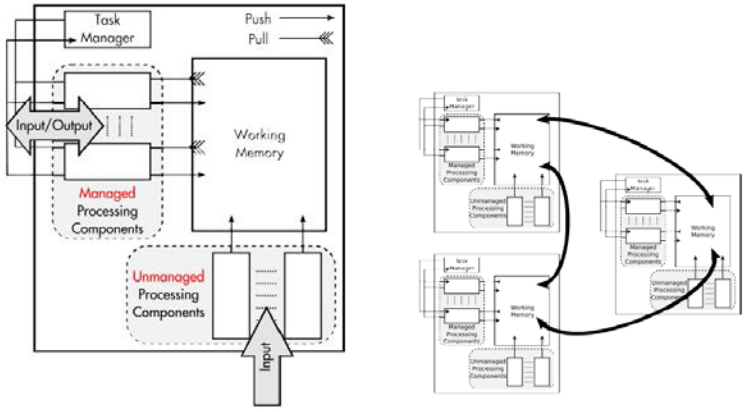
FIGURE 3.3 – Architecture de MADbot [50]

par le contrôleur au répartiteur de tâches. Cette architecture permet au robot une prise en compte de l'objectif de l'opérateur lui permettant de générer un comportement délibératif. Les plans générés pour un objectif prennent en compte au préalable toutes les situations possibles, sans pour autant qu'il n'existe un comportement réactif (en cours d'exécution). Cela engendre deux problèmes : Manque de réactivité et Manque de contrôle d'exécution. Si le robot rencontre une situation de blocage, le système ne pourra rien faire sauf si l'opérateur adapte son objectif et dès lors, il prendra en compte ce blocage. Cette architecture ne permet donc pas au robot de raisonner et de reconnaître son environnement au cours de son évolution.

3.2.4 CAS : CoSy Architecture Schema

CoSy Architecture Schéma (CAS) [108], développé à l'université de Birmingham, représente la conception d'une architecture pivot construite à partir d'un ensemble de Sous-Architectures (SA : "SubArchitecture"), où une SA modélise des traitements appelés composants liés au fonctionnement du système. A chaque composant est associé une fonctionnalité du robot (planification, reconnaissance d'objets,..). Dans cette architecture, il n'existe pas de communication directe entre les composants qui ne

peuvent échanger les données qu'à travers la mémoire de travail partagée présente dans chaque SA. De plus, chaque SA dispose d'un gestionnaire de tâches spécifiant l'activation des composants. Ce composant peut être actif ou non. L'architecture d'une SA ainsi que les liasions entre les différents SA sont illustrées dans la Fig. 3.4. Dans PlayMate/Explorer CoSy Sub-Architecture Schema (PECAS) [107], les SA re-



(a) L'architecture d'une SA dans CoSy

(b) Les liaisons entre les différentes SA

FIGURE 3.4 – Architecture CoSy

groupent les composants par fonction (par exemple, la vision, la communication, ou la navigation). Toutes ces SA sont actives en parallèle, combinant généralement des comportements réactifs et délibératifs, et le tout fonctionnant sur une SA spécifique pour les représentations (qui est nécessaire pour le traitement robuste et efficace de la tâche). Ces représentations disparates sont unifiées, ou liées, par une SA de liaison, qui effectue l'abstraction et fusionne de l'information de l'autre SA [119]. PECAS permet d'utiliser les multiples possibilités offertes par les SA d'un système afin d'effectuer de nombreux objectifs fixés par les opérateurs. Afin de munir les robots d'une architecture générique et extensible pour accomplir les objectifs, le calcul et la coordination du comportement du système sont traités comme un problème de planification. L'utilisation de cette architecture permet d'augmenter l'autonomie du robot. Le robot peut adapter, d'une manière autonome, ses plans d'action durant l'évolution des situations en utilisant la planification continue et est donc bien adaptée aux environnements dynamiques. S'appuyer sur la planification automatisée signifie que les tâches du robot doivent être posées comme objectifs pour un planificateur et le comportement, pour atteindre ces objectifs, doit être codé comme des actions que le planificateur peut traiter.

Bien que cette architecture permette au robot de faire face aux situations qu'il peut rencontrer, nous remarquons un silence à propos de la capacité de raisonnement et d'apprentissage de nouvelles situations d'un robot muni de cette architecture afin qu'il

adapte son mouvement sans pour autant procéder à la planification automatique.

3.2.5 Synthèse

Les travaux de recherches présentant les architectures explicitées dans cette section n'intégrent pas des fonctions rendant les robots capable de comprendre. Chacune des architectures de l'état de l'art dans le domaine des sciences cognitives et robotiques présentées dans cette section s'occupe uniquement d'un ensemble de fonctions mais n'intègre pas les cinq fonctions distinguées dans la section compréhension du contexte (cf. section 2.4). Notre contribution dans ce travail de thèse a ainsi pour objectif d'intégrer de l'intelligence à un robot pour le rendre capable de comprendre les attentes de l'opérateur et de déclencher les événements appropriés aux différentes situations dans lesquelles il se trouve pour aider et satisfaire l'opérateur. Durant nos recherches, nous avons conçu et mis en place un système logiciel capable de rendre un robot capable de faire face, d'une manière optimale, à des situations de blocage qu'il peut rencontrer au fur et à mesure qu'il évolue dans son environnement.
De ce fait, **notre première contribution** consiste en la proposition d'une architecture basée sur l'approche de Situation Awareness pour rendre un robot conscient de la situation afin de satisfaire l'opérateur et faire face à l'aspect dynamique de l'environnement. Ce qui signifie que le robot devra être capable de :

- Détecter des situations de blocage,
- Interpréter et raisonner sur les situations
- Prendre les décisions adéquates pour atteindre son objectif.

3.3 Représentation des connaissances dans les systèmes robotiques

Les systèmes robotiques déployés dans les différents environnements complexes sont confrontés aux problèmes de la représentation et de raisonnement sur les connaissances du domaine incomplètes acquises, à partir des données capteurs. Une bonne représentation des connaissances minimisera l'intervention de l'opérateur car elle permet de pallier l'absence et/ou le bruitage des données capteurs [21]. Le robot doit donc comprendre la situation dans laquelle il se trouve en assurant une compréhension symbolique de l'environnement dans lequel le système évolue [260].

3.3.1 Types de représentations des connaissances

En tant que branche de l'intelligence artificielle symbolique, la représentation des connaissances vise à la conception de systèmes informatiques capable d'interpréter l'état du monde. Plusieurs types de représentation des connaissances existent dont les plus utilisés qui sont les règles, les frames, les réseaux sémantiques, la logique et les

ontologies.
L'utilisation des règles est aisée et compréhensible par l'homme. Les règles permettent la représentation de connaissance dynamique. Une régle est généralement éxprimée sous la forme :

SI Premise (s) ALORS conséquence (s).

Ce type de représentation exige l'expérience d'un expert. Les règles sont dynamiques et peuvent être archivées et mises à jour si nécessaire. Dans un système de raisonnement automatisé, il est facile d'appliquer cette approche dans la construction d'une modélisation d'un robot. **Les Frames** [169] supposent que la connaissance humaine n'est pas compliquée mais s'articule autour des unités d'information. Tous les scénarios de la vie quotidienne peuvent être représentés en tant que Frame. Un Frame est une structure de données comprenant à la fois des informations déclaratives et procédurales. Il représente une situation typique et comprend des attributs (fentes) pour les objets. Chaque attribut a un aspect unique (facette) de la description des concepts qu'il représente.
Les réseaux sémantiques sont des modèles psychologiques de Quillian et Raphaël [212], ils sont des outils qui simulent les performances de la mémoire. Il s'agit d'un modèle qui montre 1) comment l'information peut être représentée dans la mémoire et 2) comment on peut accéder à ces informations. Un réseau sémantique est composé de nœuds dont les interrelations sont établies par des pointeurs étiquetés. Les noeuds sont les différents types d'informations dans la mémoire. Chaque noeud peut être associé à des propositions et des états qui caractérisent les propriétés s'appliquant aux noeuds du réseau. L'étiquette apposée sur le pointeur indique le type de relation entre deux noeuds. Il n'existe aucune norme dans les relations, mais il y a des relations communes :

X Y instance ; X is-a Y ; X has-Part Y.

Les réseaux sémantiques utilisent les métadonnées pour représenter les informations.
La logique fait partie d'une famille de langages de représentation des connaissances qui peuvent être utilisés pour représenter la connaissance terminologique d'un domaine d'application dans un cadre formel et structuré. Ce type de représentation des connaissances a été développé comme une extension des frames et des réseaux sémantiques. **Une ontologie** est une spécification explicite d'une conceptualisation. Le terme est emprunté à la philosophie. Pour les systèmes d'Intelligence Artificielle, ce qui "existe" est ce qui peut être représenté. Lorsque la connaissance d'un domaine est représentée dans un modèle déclaratif, l'ensemble des objets qui peut être représenté est appelé l'univers de l'application. Cet ensemble d'objets et les relations entre eux, se retrouvent dans le vocabulaire de représentation avec lequel un programme représente la connaissance. Ce type de représentation des connaissances offre une grande expressivité. Une ontologie représente un ensemble de concepts structurés qui sont organisés dans un graphe dont la relation peut être d'ordre sémantique et/ou de la composition

et de l'héritage. Une ontologie offre la possibilité d'avoir un vocabulaire commun pour décrire un domaine ainsi que les classes et les relations de typages primitifs. Le plus important est que l'ontologie possède des capacités d'inférence.
Les différents types de représentations des connaissances décrits ci-dessus ont été une contribution à la mise en place de l'intelligence dans les systèmes robotiques. Toutefois, certaines lacunes sont relevées, en particulier avec les types basés sur les règles, les Frames, les réseaux sémantiques et la logique. Ces lacunes sont en fait liées à la lenteur, la complexité croissante des systèmes quand on considère la classification et les liens de causalité existants dans la représentation du contexte et de l'environnement réel.
Avec la nécessité d'une plus grande interactivité entre l'opérateur et le robot, l'ontologie est présentée comme une approche qui pourrait aider à remédier aux inconvénient des différents types de représentation des connaissances. En outre, en utilisant les ontologies cela permet de gagner en interopérabilité en fournissant un accès commun à l'information et une compréhension commune des concepts. Elles permettent aussi la réutilisation des sources de connaissances. Les techniques basées sur les ontologies offrent dans un même cadre une puissante représentation des connaissances du domaine ainsi qu'une capacité de raisonnement. Nous présentons dans cette section les projets de recherches en robotique ayant une représentation de connaissances sous forme d'ontologies.
L'ontologie est une thématique bien élaborée dans le domaine de l'intelligence artificielle. Il existe de nombreuses ontologies couvrant les connaissances encyclopédiques dont la plus importante étant Cyc [84]. Ces ontologies sont peu ou pas utiles pour le contrôle de robots autonomes [240]. Les robots ont des exigences très spécifiques qui ne sont généralement pas considérées par ces bases de connaissances. Néanmoins, il existe dans la littérature des systèmes basés sur les ontologies adaptées aux applications robotiques. Dans la suite, nous présentons deux types de systèmes robotiques basés sur les ontologies. Le système propriétaire OUR-K (Ontology-Based Unified Robot) [144] adopte une plateforme construite sur son propre système ontologique, qui est original. Ensuite, nous présentons un ensemble de systèmes basés sur une structure ontologique open source, nommée OpenCyc, représentant en fait l'encyclopédie Cyc.

3.3.2 OUR-K : Ontology-Based Unified Robot

Nous présentons dans cette sous section, l'historique de l'évolution du système OUR-K ainsi que sa description.

3.3.2.1 Historique

Cette plateforme a été proposée en 2006 par Wonil Hwang du Department of Industrial Engineering, KAIST, Daejeon, Korea [114]. Au départ, elle se nommait Multi-layered Context Ontology Framework (MLCOF).
MLCOF est une plateforme pour la compréhension, la modélisation du contexte et le

raisonnement pour la reconnaissance d'objets. MLCOF est constituée de six couches (image, 1D géométrie, 2D géométrie, 3D géométrie, objet, espace). MLCOF modélise l'information de contexte de bas niveau (images) et de plus haut niveau (sémantique de l'objet de l'espace). En 2007, l'équipe de Suh du "College of Information and Communications", Hanyang University, Seoul, Korea a amélioré la technologie MLCOF avec OMRKF (Ontology-based Multi-layered Robot Knowledge Framework) qui propose l'implémentation de l'intelligence du robot qui sera utilisé dans son environnement. OMRKF est constitué de 4 couches de connaissances, des axiomes et deux types de règles. Les 4 couches sont la perception, le modèle, l'activité et le contexte. Chaque couche est organisée dans une hiérarchie de 3 niveaux de connaissance et 3 couches d'ontologies. Le successeur d'OMRKF est la plateforme OUR-K (Ontology-Based Unified Robot Knowledge) qui a été publiée en 2011 par la même équipe [236].

3.3.2.2 Description du système

Le système OUR-K (Ontology-Based Unified Robot) est constitués, de deux parties : La description des connaissances et l'association des connaissances.
La description des connaissances définit les données et l'environnement du robot dans cinq classes de connaissances (Fig. 3.5), et intègre toutes les connaissances du robot : depuis les connaissances de bas niveau (perception et action) jusqu'aux connaissances de haut niveau (ou du monde réel) dont les connaissances sur les objets, l'espace et le contexte. Chacune de ces classes présente deux à trois niveaux de connaissances où chaque niveau inclut trois couches ontologiques représentant les connaissances génériques (la méta ontologie), le domaine des connaissances (le schéma d'ontologie) et les instances des connaissances (l'instance d'ontologie). La Fig. 3.5 montre cette hiérarchie. Ainsi, les classes représentant les connaissances sont décrites comme suit :
La classe ***Object*** contient trois niveaux de connaissances :

- Niveau incluant les parties fonctionnelles et de perception. Un objet peut être décomposé en parties selon ses fonctions. Exemple : une tasse est composée d'un corps et d'une ance et chaque élément de cette composition a une fonction : la première est la contenance d'un liquide et la seconde est pour la prise en main. Cette distinction est faite pour inclure les objets selon leur fonction (*PartObject(O)*)
- Niveau incluant le nom et la fonctionnalité de l'objet (*Object(O)*)
- Niveau incluant les objets qui sont liés tels que (tasse et soustasse) (*CorespondObject(O)*)

La classe ***Space*** contient aussi trois niveaux :

- Niveau contenant la carte métrique (*MetricMap(M)*)
- Niveau contenant la carte topologique (*TopologicalMap(M)*)
- Niveau contenant la carte sémantique (*SemanticMap(M)*)

La classe ***Features*** contient deux niveaux :

- Niveau contenant les caractéristiques de perception (*FeatureConcept(P)*)

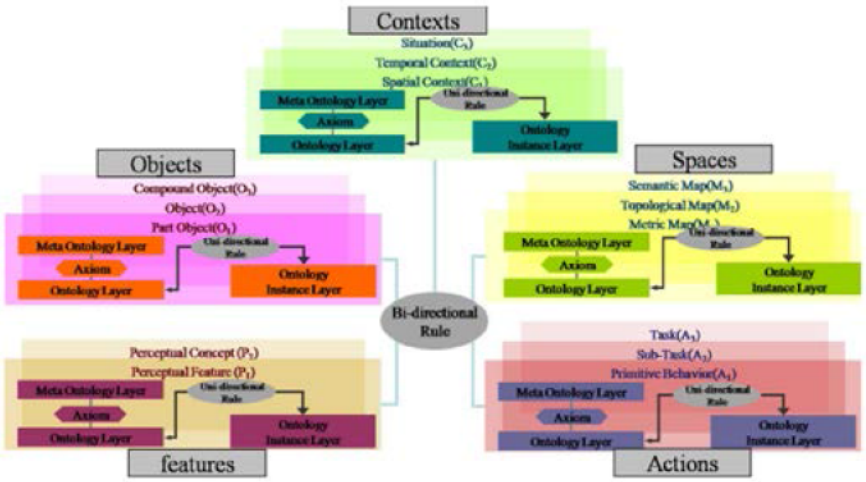


FIGURE 3.5 – Description des connaissances dans OUR-K [144]

- Niveau contenant les concepts de perception (*PerceptualConcept(P)*)

La classe ***Contexts*** est composée de trois niveaux :

- Niveau contenant les relations spatiales telles que (in, on, left et right). Elles se déduisent des instances et des valeurs de propriétés contenues dans les classes ***Object*** et ***Spaces*** (*SpatialConcept(C)*).
- Niveau contenant les relations temporelles comme before, after, met-by, overlap et meet. Ces relations se déduisent en utilisant les instances du modèle et le niveau de contexte spatial (*TemporalContext(C)*).
- Niveau contenant les informations de la situation. Dans ce niveau est représentée la situation (la liste des objets de la scène et les relations entre les objets) pour déterminer le stratagème de navigation (*Situation(C)*).

La classe ***Actions*** est composée de trois niveaux de connaissances :

- Niveau de comportement primitif tel que les fonctions atomiques du robot (*PrimitiveBehavior(A)*)
- Niveau de sous-tâche qui contient les sous-tâches telles que gotoSpace, localization et regognizeObject (*Sub-Task(A)*).
- Niveau de tâche qui décrit le but avec ses effets et ses préconditions symboliques telles que delivery, navigation, findObject et generateContext (*Task(A)*).

L'association des connaissances est un module dans lequel sont créées et décrites les relations existantes entre les classes du module de description des connaissances. Pour former les relations, cette technologie utilise :

- Les inférences bayésiennes qui déduisent des variables aléatoires inconnues pour les classes de connaissances avec des conditions d'indépendance.
- Les inférences logiques qui spécifient et déduisent les relations à partir des pro-

priétés ontologiques des classes de connaissances.
- L'algorithme heuristique qui cherche l'information à propos des classes de connaissances avec des connaissances connues au préalable.

Avec cette méthode le robot peut reconnaître les objets, comprendre le contexte dans lequel il se trouve et éviter les obstacles.

Inconvénient :
Le calcul du chemin suivi (Navigation) par le robot muni de la technologie OUR-K, se fait au préalable avant que le robot ne se mette en service. Ce système OUR-K permet au robot de savoir, par la carte métrique et topologique de la classe Espace, les nœuds par lesquels il doit passer pour atteindre son objectif. Cependant, le robot ne peut se rendre compte du changement pouvant intervenir sur l'environnement, tel que l'apparition soudaine d'un obstacle inattendu sur son chemin, sans l'intervention de l'opérateur.

3.3.3 OpenCYC : Open source Computer Technology Corporation

Cette sous section présente l'encyclopédie OpenCyc ainsi que les travaux de recherche en robotique qui ont développé leurs ontologies en se basant sur cette encyclopédie.

3.3.3.1 Présentation

Le projet CYC[4] lancé par Douglas Lenat est le plus grand projet existant visant à capturer ses connaissances et à construire des raisonnements logiques à partir de celles-ci. Depuis vingt ans, une dizaine de volontaires passent leur temps à entrer de nouvelles données dans CYC. CYC contient actuellement plus de deux millions cinq cent mille faits et règles sur la vie de tous les jours, relatif à près de deux cent mille concepts différents, et l'équipe de CYC en ajoute chaque jour des dizaines de nouveaux. *C'est le plus grand « système expert », et également le plus grand « réseau sémantique » jamais construit*[5].
OpenCYC[6] a été publié en juillet 2009 et incluait la totalité de l'ontologie de CYC, soit plusieurs centaines de milliers de termes ainsi que des millions d'assertions reliant les termes les un aux autres, même s'il s'agit principalement d'assertions taxonomiques plutôt que de règles complexes disponibles dans CYC. La base de connaissances d'OpenCYC 1.0 contient 47 000 concepts et 306 000 informations, et peut être explorée sur le site Web d'OpenCYC.
Dans le domaine de la compréhension du contexte en robotique, on distingue deux systèmes qui dérivent directement d'OpenCYC. Le premier est fondé sur l'interaction Homme Machine et le deuxième est fondé sur la description logique fournie par les

4. http ://www.opencyc.org/doc/tut/
5. http ://sboisse.free.fr/technique/info/cyc.php
6. http ://www.cyc.com/opencyc/

mécanismes de spécification et les représentations centrées sur les actions. Dans ce qui suit nous présentons les deux systèmes issus d'OpenCyc. La plateforme développée au LAAS, ORO (Open RobotOntology) et celle développée au TUM [240] en Allemagne qui est nommée KnowRob.

3.3.3.2 ORO : Open Robot Ontology

Le premier système qui a vu ses débuts avec la technologie nommée ORO (Open-Robot Ontology) a été publié en 2010 par Lemaignan et Ros du CNRS - LAAS, Toulouse, France. La technologie ORO est une base de connaissances dans laquelle les connaissances du robot sont enregistrées, gérées et exposées. La plateforme ORO est open-source. Elle est conçue comme un service visant à maintenir un stockage cohérent des faits (représentés techniquement comme des triples RDF). ORO comprend une ontologie de compréhension de connaissances pour la classification et le raisonnement. ORO gère, en même temps, plusieurs modèles d'agents représentant les objets que le robot rencontre. ORO connecte avec cohérence deux éléments de connaissances pour créer des blocs d'informations du monde afin d'interpréter/de comprendre un contexte.
La Fig. 3.6 de [138] montre les trois couches de la plateforme ORO. La première couche Front-end est l'interface du robot avec le monde. La deuxième couche Modules contient les opérations, telles que le stockage et la récupération des connaissances, et des plugins nécessaires à l'Interaction Homme Machine. Cette couche est basée sur des ontologies parallèles qui sont stockées sur la troisième couche qui est le Back-ends. L'usage expérimental d'ORO mis en oeuvre par l'apprentissage des connaissances à travers l'Interface Homme Machine est implémenté sur trois différents robots BERT, Rosie et Jido. Sur le robot BERT, la tâche d'acquisition des connaissances, "Point and Learn", se fait comme suit : l'opérateur montre au robot l'objet qu'il désire lui faire apprendre. Le robot demande le nom et le type de l'objet. Le module de perception identifie et localise l'objet puis le module de détection met à jour ORO avec la liste d'objets connus et leurs états (en mouvement ou pas) et la relation de l'objet avec les autres). L'Interface Homme Machine est basée sur CLSU Toolkit qui se charge de la reconnaissance du dialogue, la synthèse du dialogue et le traitement du langage naturel basique.
Concrètement, par consultation d'ORO pour manipuler l'objet, l'interface récupère l'identifiant de l'objet que l'opérateur désire manipuler et demande à l'opérateur en cas de non reconnaissance de l'objet son nom et son type. Sur le robot Rosie, le scénario "Odd One Out" qui est une extension de "Point and Learn" est mis en place. ORO est utilisé dans l'Interaction Homme Machine et dans le raisonnement. Le robot prend un objet inconnu de la table et demande à l'opérateur le nom et le type de cet objet. L'opérateur donne des informations à propos de cet objet jusqu'à ce le robot le reconnaisse. Le scénario "Spy Game" est mis en place sur le robot Jido [138]. Le but de ce scénario est de découvrir l'objet ou le concept auquel appartient l'objet

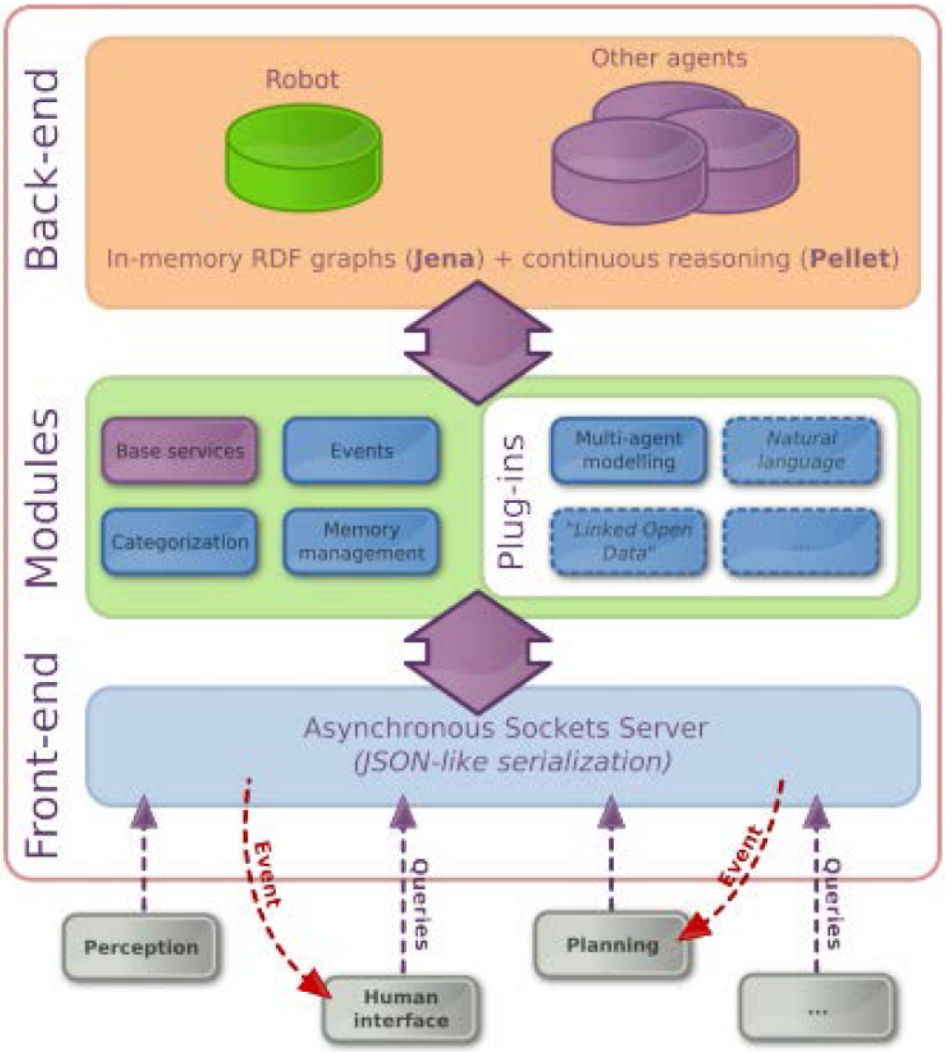


FIGURE 3.6 – Architecture de la plateforme ORO [138]

et ceci par la demande d'une réponse à des questions telles que : est-il vert ? Est-ce que c'est une machine ? Est ce qu'il est à droite ? etc... ORO est mis en place pour l'apprentissage des objets, plus généralement pour l'acquisition des connaissances.

Inconvénient :
Dans ORO, l'opérateur décrit vocalement les caractéristiques des objets inconnus du système pour les caractériser selon ce qui est dans l'ontologie. Les objets sont classés par type uniquement (nom : coca ; type canette), l'opérateur donne toutes

les informations, le système ne fait pas d'analogie (de classement automatique des informations). L'ontologie risque de se déstructurer avec l'introduction de nouveaux objets car la classification ne s'appuie pas sur la typologie déjà définie. ORO ne prend pas en compte le comportement spécifique associé aux objets : la méthode se veut générale et n'est pas spécialement adaptée à la robotique de manipulation ; on saisit les objets toujours de la même manière.

3.3.3.3 KnowRob : A knowledge processing infrastructure for cognition-enabled robots

Dans ce paragraphe, nous allons décrire le deuxième système dérivé d'OpenCYC et qui est basé sur la description logique fournie par les mécanismes de spécification et les représentations centrées sur les actions. Pour suivre la chronologie des développements qui ont abouti à cette technologie, nous allons commencer par KnowRob pour aboutir à K-CoPMan. KnowRob dérive de ce grand réseau sémantique OpenCYC. Il s'agit d'une technique de représentation de connaissances de premier ordre (haut niveau) ayant pour but principal l'acquisition automatique des concepts (du monde) par observation et par expérience et de fournir une réponse rapide. D'autres sources de connaissances sont intégrés dans KnowRob pour former un système complet pour un robot afin d'être autonome et subvenir aux besoins des opérateurs. La Fig. 3.7 montre la disposition de ces technologies pour former un système intelligent pour un robot. La Fig. 3.8 montre la représentation des connaissances de KnowRob. La représentation des connaissances dans KnowRob est faite par une ontologie dans laquelle nous pouvons distinguer les quatre classes suivantes :

- Action
- Event
- SpatialThing : Cette classe contient les endroits accessibles par le robot.
- Computable : joue le rôle d'interface entre le système et ses observations. Elle intègre les observations dans la représentation des connaissances.

La dernière classe "Computable" est une classe très importante dans le système. Cette classe crée des instances ou des relations entre les instances (comme montré par des flèches dans la Fig. 3.8), tout dépend des propriétés de l'objectif à atteindre. La définition des propriétés calculables spécifie la commande de lecture de l'objet ou plus généralement du sujet depuis un triplet OWL. Avec l'intégration des observations dans le système, les propriétés calculables sont utilisées pour calculer de nouvelles relations entre les connaissances déjà existantes comme les relations entre les objets (on, in, below) pour déterminer la position exacte des objets.

3.3.3.4 K-CoPMan : Knowledge for COgnitive Perception for Manipulation

Après le développement de KnowRob, une extension de cette technologie a été réalisée par la même équipe en 2011. Cette extension s'appelle K-CoPMan [192].

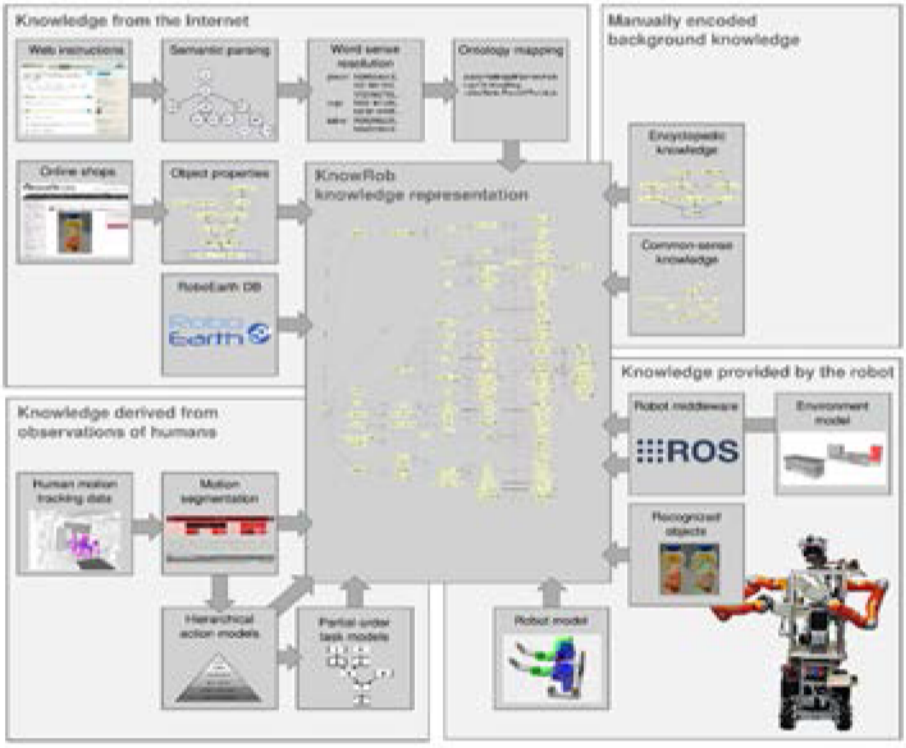


FIGURE 3.7 – Connaissances intégrées à KnowRob [240]

Cette technologie permet de saisir et de manipuler les objets. La tâche principale d'un système robotisé basé sur la saisie d'objets nécessite l'utilisation des modules de perception liés à la représentation des connaissances, afin de fournir des solutions optimales. Le système K-CoPMan utilise l'information sémantique dans la cartographie et la planification pour la préhension. Toutefois, ce système n'a pas la connaissance conceptuelle qui peut aider le module de perception à l'identification des objets dans le champ de vision pour la tâche de manipulation. Le système dépend entièrement des modèles d'objets en 3D afin d'effectuer la saisie. Cela restreint l'évolutivité, l'extensibilité, la convivialité et la polyvalence du système. KnowRob et K-CoPMan sont intégrés dans la plateforme ROS (cf. section 2.5.2.1).
Pour attraper l'objet, le robot est capable de le reconnaître. Par la combinaison de la carte d'environnement avec les connaissances de l'ontologie du système, le robot peut connaître l'objet par ses fonctionnalités. Les connaissances requises pour reconnaître les actions qu'on peut faire sur les objets se trouvent dans OpenCYC ou dans la base de données OMICS (Open Mind Indoor CommonSense). Le robot connaît l'état courant de l'objet (e.g. dans le placard il y a une tasse). La position de la tasse dans le placard est fournie par un tag Radio-Identification déposé à l'intérieur du placard.

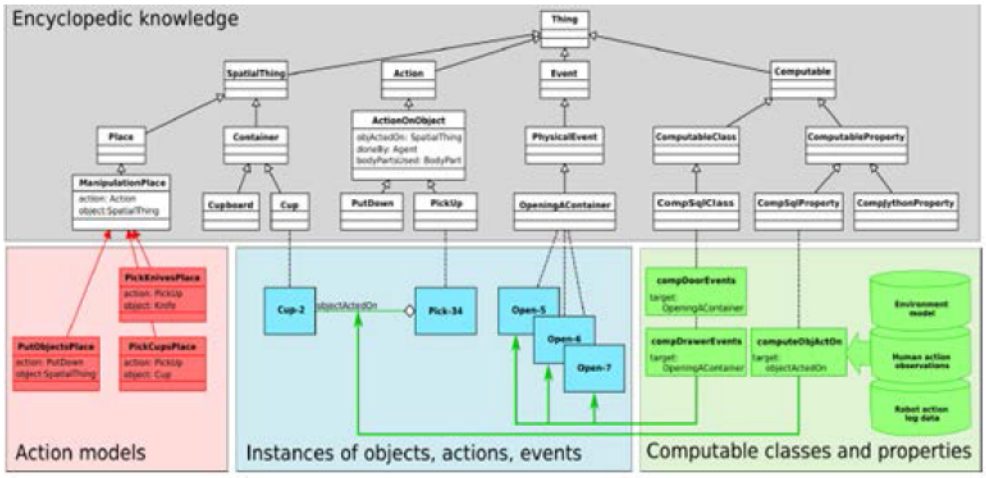


FIGURE 3.8 – Ontologie de KnowRob

Inconvénient :
KnowRob ainsi que K-CoPMan ne prennent pas en compte la structure du système robotique qui devra les supporter, ce qui est défavorable pour la généricité de ces plateformes. En plus, ils utilisent des informations supplémentaires issues d'internet pour la représentation des connaissances, seulement, ces informations ne sont pas objectives. En effet, pour réaliser une tâche, les personnes ne se comportent pas toutes de la même manière, et donc les comportements décrits et disponibles sur le web reflètent plutôt des comportements propres aux individus qui les ont exposés. Par conséquent, si un robot consulte le web pour savoir comment se comporter pour effectuer une tâche, il n'obtiendra pas des actions optimales et bien organisées pour la réaliser. Aussi, KnowRob et K-CoPMan utilisent des données issues d'un tag RFID pour localiser les objets.

3.3.4 Synthèse

Dans cette section, nous avons dressé un aperçu sur les principaux systèmes de représentation de connaissances expérimentés en robotique. OUR-K, la dernière version de la plateforme coréenne, est capable de modéliser l'environnement et de comprendre la scène ; mais lors de la manipulation mobile d'un robot muni de OUR-K, le chemin pour aboutir à la scène est calculé au préalable et donc si un obstacle survient après que le robot soit lancé, il sera incapable de changer de chemin ou de comportement.
Dans la technologie ORO, l'opérateur décrit les caractéristiques des objets nouveaux du système pour les caractériser. Le robot ne demande que le type des objets (nom : soda ; type canette) pour les classer. Dans ORO il n'est pas prévu d'associer de com-

portements spécifiques aux objets (le robot manipule un soda comme une tasse) : la méthode se veut générale et n'est pas spécialement adaptée à la robotique de manipulation ; on saisit les objets toujours de la même manière. Jido et Rosie utilisent les scénarios mis en place avec ORO.
KnowRob, bâti également sur OpenCYC, obtient des connaissances de plusieurs sources dont un site internet et les tags RFID implémentés dans l'environnement, pour connaître l'endroit exact de l'objet, dans un placard par exemple. Même avec ces connaissances KnowRob est incapable de fournir des éléments sur la relation qui peut exister entre les objets pour déduire la technique à adopter pour manipuler un objet sans toucher les autres par exemple.
Ce panorama montre que les travaux de représentation des connaissances utilisant les ontologies en robotique sont diversifiés et considèrent, pour la plupart, des connaissances d'ordre général. Seulement, leur adaptation à des contextes spécifiques reste difficile car ils manquent de généricité.
Notre deuxième contribution se situe dans la proposition d'une modélisation générique de la représentation des connaissances du contexte du robot (facilement intégrable sur différents robots), basée sur les ontologies pour analyser les scènes, comprendre le désir de l'opérateur, l'état interne du robot lui-même et les situations de blocages.

3.4 Raisonnement dans les systèmes robotiques intelligents

Depuis les années 50 et avec l'apparition de l'intelligence artificielle (IA) [106], deux approches de raisonnement ont vu le jour et ont été adoptées dans les travaux de recherche afin de concevoir et mettre en place des machines dites "intelligentes". La première approche appelée en anglais "Making a mind" et qualifiée de symbolique, consiste à doter une machine de mécanismes de raisonnement et ainsi la rendre capable de manipuler des données symboliques représentant les connaissances d'un domaine. La deuxième approche appelée en anglais "Modeling the brain" est qualifiée de connexionniste. Cette approche s'inspire, comme l'indique [111], du fonctionnement du cortex cérébral. L'entité principale de ce type d'approche est un modèle formel du neurone. Un système conçu en se basant sur ce type d'approche est donc constitué de plusieurs neuronnes interconnectés entre eux.
Á partir des années 70, les systèmes robotiques ont commencé à utiliser les approches de raisonnement de l'IA pour faire face à la variabilité et la dynamique du contexte. En effet, pour être considéré comme intelligent, un robot doit être muni d'un système lui permettant de raisonner. Dans la littérature traitant des recherches menées par la communauté robotique, on remarque l'utilisation du raisonnement sur la géométrie des objets de l'environnement pour les tâches liées à la manipulation, et sur la géométrie même de l'environnement pour les tâches liées à la navigation [133]. Les

approches connexionnistes en robotique sont utilisées pour raisonner sur les incertitudes liées à la complexité du contexte. Les approches symboliques permettent à un robot de trouver le plan d'action adéquat à partir d'un état initial, un état final et l'ensemble d'actions qu'il peut faire. Le raisonnement hybride est une mixture entre les différents raisonnements. Généralement, dans la communauté robotique, cet embranchement est conçu entre les raisonnements géométriques et symboliques.
Dans ce qui suit, nous présentons ces principales approches de raisonnement utilisées en robotique.

3.4.1 Approches symboliques

Les symboles sont utilisés en robotique généralement dans le domaine de planification. Le planificateur symbolique utilise le raisonnement symbolique. Depuis plusieurs décennies, ce type de raisonnement est utilisé dans plusieurs projets de recherche. Nous traitons dans cette sous section les technologies mises en place pour effectuer ce type de raisonnement. Ces technologies sont les systèmes à base de règles d'inférences et les arbres de décision.

3.4.1.1 Raisonnement déductif : Les Système à Base de Règle (SBR)

Le raisonnement à base de règles est un raisonnement permettant l'exploitation de plusieurs connaissances acquises des humains afin de réaliser le traitement des problèmes avec des compétences dans des domaines limités. Dans ce genre de système, il existe une considération (ou une séparation) entre les connaissances du domaine et les programmes qui opérationnalisent ces connaissances.
Un SBR est constitué d'un langage d'expression des connaissances, de structures d'accueil pour la connaissance spécifique d'un domaine d'application et d'un moteur d'inférences (ou des structures de contrôle) chargé de raisonner (ou résoudre) sur un problème donné. Nous rappelons succintement ces notions ci-dessous :

a Langage d'expression des connaissances :

Les langages naturels (Symboliques) constituent la meilleure forme pour communiquer ou dialoguer entre l'homme et la machine. Le dialogue Homme à Homme est appelé communication. Dans le dialogue Homme-Machine (cas qui nous concerne) nous distinguons deux types de dialogue : le premier est celui d'instruction du système (opérateur) et le deuxième est le dialogue de consultation du système (utilisateur).

b Structures de connaissances :

Les éléments de connaissances sont organisés en une base de connaissances (BC). On trouve alors les faits sur un problème précis à résoudre, les méthodes d'inférences (ou règles d'inférences) et bien sûr les connaissances sur le domaine envisagé. Nous pouvons distinguer dans la littérature deux types de connaissances :

- Les connaissances appelées assertionnelles qui sont des connaissances décrivant les situations existantes ou qui peuvent se produire
- Les connaissances appelées opératoires (CO) contenant des connaissances représentant ce que le système peut faire dans le domaine. Ce que le système peut faire est désigné par une règle.

c Moteur d'Inférence (MI) :

Un moteur d'inférence est un programme ou un circuit intégré permettant exécuter les actions définies par les règles sur les connaissances existantes dans la BC. Une règle est une connaissance qui organise les faits entre eux. Une règle consiste en un ensemble d'hypothèses ou de prémisses et le reste est noté comme sa déduction. Plusieurs systèmes utilisant ce genre de raisonnement existent en robotique . L'un des systèmes les plus connus est le FlexNav qui est un système permettant à un robot de naviguer pour atteindre son objectif.

3.4.1.2 Raisonnement basé sur les arbres de décision

La constitution des arbres de décision à partir de connaissances est une discipline qui date de plusieurs décennies avec les travaux de Morgan et Sonquist en 1963 [174]. Ils sont construits en une structure hiérarchique ayant la forme d'arbre. Selon [188], cette structure est construite grâce à des algorithmes d'apprentissage à partir d'exemples. L'arbre obtenu représente la classification d'exemples, en s'appuyant sur les connaissances existantes dans la base d'apprentissage. Une définition un peu plus formelle [216] des arbres de décision stipule que c'est un graphe orienté, sans cycles, dont les nœuds portent une question, les arcs des réponses, et les feuilles des conclusions, ou des classes terminales. Généralement, un arbre de décision se construit à l'aide d'ensemble d'apprentissage en raffinant sa structure. Un ensemble de questions est construit afin d'éliminer des sous-ensembles jusqu'à n'obtenir à la fin que des observations d'une seule classe. Les résultats des algorithmes sont les branches de l'arbre et les sous-ensembles sont les feuilles. A l'apparition d'un nouveau cas, la recherche se fait en parcourant un chemin qui commence depuis la racine jusqu'à aboutir à une feuille. Plusieurs méthodes s'appuyant sur les arbres de décision existent dans la littérature. La plupart des méthodes s'appuient sur la méthode ID3 de Quinlan [213] qui est une référence incontournable citée 13607 selon Google Scholar ainsi que la méthode CART [25]. Plusieurs systèmes existent dans la littérature de l'IA qui sont basés sur les arbres de décision. Nous citons les plus connus :

- C4.5 qui utilise la méthode ID3 pour les programmes d'apprentissage pour les machines,
- ID5R qui est un système de construction incrémentale d'arbres de décision permettant de faire l'apprentissage d'exemples sans recommencer tout l'apprentissage dès le début.

- SPINA est un système permettant la construction de graphes de décision. Il implémente une méthode de discrétisation, une méthode qui évite le surapprentissage et permet aussi l'explicitation de règles représentées dans les graphes de décision [264].

3.4.1.3 Raisonnement à partir de cas (RàPC)

Le raisonnement à partir de cas au niveau symbolique en robotique a été utilisé dans un seul système. Ce système est nommé CEDRIC pour "Case-base Enabled Dialogue Extension for Robotic Interaction Control". Le but de ce raisonnement, dans ce système, est de permettre à un système robotique de comprendre le langage naturel et de générer les réponses adéquates dans un système de dialogue homme-robot. CEDRIC a été développé dans le cadre du projet Witas [64] dont le but était de concevoir et mettre en place un véhicule aérien autonome sans pilote ("Unmanned Aerial Vehicle "UAV" ou drône) capable de prendre des décisions rationnelles sur la poursuite de l'exploitation de l'avion, sur la base de diverses sources de connaissances, y compris les connaissances géographiques, préenregistrées, et les connaissances obtenues à partir de capteurs de vision. Ces connaissances sont communiquées à l'UAV par liaison de données.
La phase de recherche du cas similaire est faite de la manière suivante : La parole en entrée est comparée à la partie appelée "problème" de chaque cas de dialogue dans la mémoire de CEDRIC, et une valeur de similitude est alors calculée. Les cas sont alors classifiés selon la valeur de similarité. La valeur de similarité est basée sur la façon dont chaque mot de la parole d'entrée correspond aux mots dans la formulation du problème. Une affectation de poids à chaque mot est faite au préalable et selon le lexique.
La phase d'extraction de cas est exécutée uniquement si l'entrée correspond à un début d'un nouveau dialogue. Ce qui veut dire que CEDRIC n'adapte pas son comportement au cours du dialogue. Le raisonnement à partir de cas est considéré comme la modélisation informatique du raisonnement par analogie, similaire au raisonnement de l'être humain. La formule de calcul de la similarité ainsi que le cycle du RàPC dans le système CEDRIC sont détaillés dans [72][71].

3.4.2 Approches connexionnistes

L'approche connexionniste [132] [24] a vu le jour pour prendre en compte l'incertitude de l'environnement en prenant les meilleures décisions possibles. Cette approche permet donc un raisonnement sur l'incertain. La naissance de ce type de raisonnement provient du fait de l'impossibilité de représenter explicitement les éventualités possibles de l'environnement. Il est adapté à la robotique. Comme vu dans la sous section 2.4.1.1, l'environnement dans lequel un robot autonome évolue est un environnement dynamique complexe et incertain. L'incertitude de l'environnement vient du fait que les connaissances de l'état réel du monde (environnement, robot et opérateur) sont

incomplètes, imprécises, incertaines à propos de l'effet de l'action et de l'évolution de l'environnement. Ce raisonnement permet de prendre en compte les différents résultats des actions et des états du monde partiellement connus.
Ces approches sont probabilistes. Elles sont de plus en plus utilisées en robotique, vu que le monde réel est de nature aléatoire. Les chercheurs de ce domaine insistent sur l'indisponibilité de la gestion de l'incertitude dans les systèmes autonomes intelligents. Cette prise de conscience a suscité, au cours des dernières années, l'intérêt pour les méthodes probabilistes pour les inférences et l'apprentissage dans les systèmes cognitifs et la robotique. Des outils puissants connus comme les modèles de décision, la logique floue [55], les réseaux bayésiens [196][184][101] ou encore les réseaux de neurones [215] ont trouvé une large application dans des domaines allant de l'extraction des données et la vision par ordinateur à la bio-informatique, la psychologie et la robotique mobile. Ces réseaux permettent l'obtention des probabilités des différents événements et de déduire les effets des actions directement des données en entrée sur la base des lois de probabilité ainsi qu'une représentation basée sur des graphes.
Étant donné le succès récent des méthodes probabilistes dans l'Intelligence Artificielle et dans la modélisation du cerveau, des cadres probabilistes sont mis en place pour améliorer la compréhension de l'imitation humaine, mais également de nouvelles méthodes d'apprentissage par imitation dans les robots, pour la modélisation des objets ou des scènes.

3.4.3 Raisonnement géométrique

La cognition en robotique est nécessaire lorsque l'environnement est dynamique, lorsque les risques sont fréquents, ou lorsque la complexité de la tâche est grande. Les principales missions telles que la télé-opération, les véhicules autonomes, et l'exploitation des centrales nucléaires sont les principaux domaines d'application de la robotique intelligente. Une caractéristique particulièrement distinctive des systèmes cognitifs soulignée ici est le rôle des connaissances géométriques (la représentation des modèles géométriques) et son utilisation dans la planification des tâches. Ce type de raisonnement suggère la considération de l'identification de l'environnement et des objets le constituant, l'identification des relations entre ces objets et la production de plans d'actions pour atteindre le but fixé par l'opérateur. Le raisonnement géométrique couvre les questions de la représentation et la manipulation de modèles décrivant les objets géométriques. Il a une application générale sur l'ensemble des disciplines qui traitent des objets physiques du monde réel. Bien qu'il existe des différences dans les besoins d'information des différentes disciplines, les points communs en matière d'information géométrique sont marqués. Cette approche prend comme point de départ ces points communs et vise une architecture pour le raisonnement géométrique. La méthodologie de recherche a deux aspects connexes. La première est une recherche de concepts autour desquels une architecture peut être organisée, et le second est le développement d'un prototype de système comme un moyen pour l'exploration et

comme une démonstration de concept.

3.4.3.1 Raisonnement défini par un modèle de cognition spatiale

Le raisonnement géométrique pour la planification de tâches en robotique se base sur des travaux effectués dans plusieurs disciplines. Les travaux marquant le raisonnement géométrique sont les travaux [131][244][158], visant à la création d'un modèle de cognition spatiale similaire à celui des humains en utilisant des concepts de représentations comme ceux élaborés dans [147]. Ces concepts sont liés dans un réseau à travers lequel une propagation est utilisée pour effectuer les inférences. Une organisation hiérarchique implicite est fournie en utilisant les relations de confinement sur les entités spatiales.

3.4.3.2 Raisonnement géométrique à base de contraintes

Brooks [27] a conçu le raisonnement géométrique à base de contraintes dans le cadre du système de vision ACRONYM. Le raisonneur maintient une hiérarchie de classes (Brooks l'appelle graphe de restriction) des représentations géométriques basées sur des cylindres généralisés. Les paramètres qui déterminent un cylindre généralisé sont limités par l'utilisation de contraintes algébriques. Le raisonnement géométrique est accompli par des méthodes algébriques et numériques afin de déterminer les limites de satisfiabilité des ensembles de contraintes et des extremas des fonctions objectives. Davis [58] a étendu la procédure d'inférence spatiale métrique, une technique pour tirer des conclusions sur les positions relatives des objets basée sur des descriptions simples, décrites précédemment par McDermott [160].

3.4.3.3 Raisonnement géométrique déductif

Le raisonnement géométrique déductif est décrit dans Wing et Arbab [257]. La principale contribution de leur proposition est l'établissement d'un langage clair pour le raisonnement géométrique déductif. Les grammaires de forme [235][99] fournissent des moyens pour décrire les concepts d'objets géométriques comme des ensembles de règles de forme, de symboles, et de forme initiale. L'ensemble de ces quatre concepts définissent un langage de formes. Cette grammaire de forme est mise en place dans les systèmes à base de règles. Avec l'ajout d'une structure de contrôle, la grammaire de forme devient un système de production d'une représentation spéciale.

3.4.3.4 Raisonnement à partir de cas (RàPC)

En robotique, le raisonnement à partir de cas ou RàPC n'est pas utilisé comme raisonnement sur les symboles. Ce type de raisonnement est en fait un raisonnement par

analogie, donc plus proche du raisonnement humain. Il est utilisé dans le raisonnement géométrique. Les travaux que nous avons trouvés et explorés dans nos recherches sont principalement ceux issus des domaines de planification de mouvement, d'évitement d'obstacle et l'interaction Homme Machine.

a Planification de mouvement

Dans [130], le RàPC est effectué afin de déterminer le chemin à suivre entre deux points donnés. La base de cas est l'ensemble des chemins possibles et la distance entre deux chemins est calculée selon l'équation :

$$D(P_1, P_2) = max_{c_i \in P_1}(min_{c_j \in P_2}(d(c_i, c_j)))$$

où c_k représente les points d'un chemin.

b Evitement d'obstacle

Dans les travaux de R. Ros [219], la question d'évitement d'obstacle est traitée dans un environnement représenté selon une topologie à l'aide de RàPC. Le cas dans ces travaux est représenté par un tuple :

$$Cas =< L, O, \delta, \alpha, a >$$

où :

- L est l'ensemble des nœuds du graphe représentant les identifiants du repère. Ils sont notés $L_i|L| \geq 2$
- O représente l'ensemble des obstacles O $\subseteq$ L*L.
- δ est la fonction d'étiquetage totale pour régler la distance entre deux points du repère, qui est δ : L*L$\rightarrow$ R.
- α est la fonction d'étiquetage totale de réglage de l'angle entre deux points de repère, l_i et l_j, et de l'axe de référence X_{l_i} (la ligne passant par le repère l_i avec la même direction que l'axe des x du robot).
- a représente les actions à faire.

La similarité calculée pour trouver le chemin idéal est une similarité géométrique basée sur le voisinage d'un nœud. Le calcul de cette similarité est décrit dans [219].
Un autre travail en robotique utilise le RàPC pour faire de l'évitement d'obstacle [246]. Cela consiste dans le suivi des valeurs de capteurs ultrason. Si elles changent (Sensor Reading "SR"), le raisonnement par analogie se déclenche. Un cas dans ces travaux représente les valeurs possibles des capteurs.

$$Cas =< SR_1, SR_2, SR_3, ..., SR_n >$$

La distance de similarité utilisée pour trouver la solution est la distance de Manhattan [207].

c Interaction Homme Machine

Dans [237], le raisonnement à partir de cas est fait en continu. Il est élaboré afin de trouver le comportement du robot avec l'humain. Pour ce faire, il se base sur un cas contenant la position du robot, sa vitesse, l'orientation de la personne et les indications de la personne. Cette analogie se base sur le calcul de la distance euclidienne [13].

$$Cas =< Pos, V, Orientation, Indication >$$

3.4.4 Synthèse

Dans la communauté robotique visant à rendre le robot intelligent, le raisonnement utilisé est généralement hybride où le raisonnement symbolique est très souvent accompagné du raisonnement géométrique [5][37] et aussi associé à des approches connexionnistes [136][188]. Nous avons remarqué qu'en robotique, il n'est pas de coutume de profiter de l'expérience passée à l'aide du raisonnement à partir de cas (RàPC) qualifié de rapide et non coûteux [59]. Les rares travaux existants profitant du RàPC n'évoquent pas la question de prise en compte de la dynamicité et la complexité du contexte. Nous trouvons que le développement et l'utilisation de ce type de raisonnement s'adaptent bien à notre problématique car ce type de raisonnement offre à un robot dans un même cadre un raisonnement sur les symboles et la résolution de problème.
La troisième contribution de cette thèse concerne la conception du raisonnement sur les symboles représentant la situation courante du robot au moment du blocage.

3.5 Conclusion et Synthèse générale

Comme vu dans 2.6, l'intégration matérielle et logicielle (algorithmes et modélisations) est un enjeu majeur dans la communauté robotique. Les roboticiens considèrent qu'il y a une énorme divergence entre l'évolution des applications dans le domaine de l'informatique et de l'intelligence artificielle et l'évolution des applications dans le domaine de la robotique. En effet, comme le dit le vice président, Herman Bruyninx de EuRobotics (European Robotics) dans [31] : "The best integration's system is done on the smartphone". Dans ce contexte, il rêve d'un avenir de la robotique similaire à celui des téléphones intelligents. Cela est dû, en particulier, au manque de grandes entreprises leader dans le domaine, comme Google pour le domaine de l'informatique. Aujourd'hui, conscient de l'opportunité qui se présente, Google s'est mis à la robotique en achetant des robots de très haut niveau de part leur mécanique afin de leur ajouter des compétences en informatique.
Quant au domaine de l'intelligence artificielle, il existe plusieurs travaux qui sont très bien élaborés mais ils ne s'occupent pas de l'interfaçage avec les contrôleurs d'un robot. De plus, il y a un manque de standardisation, les équipes ne peuvent pas avoir les

mêmes bases de comparaison, ce qui rend la recherche dans l'intersection de ces domaines difficile. Nous confrontons dans ce tableau 3.1 nos contributions avec les robots les plus au point côté autonomie dans la communaté robotique. Leurs concepteurs et développeurs les ont intégrés afin de les rendre homogènes avec leurs environnements.

Contributions / *Robots*	**Architecture d'intégration de la capcité de compréhension**	**Représentation des connaissances**	**Raisonnement**
Care-O-Bot [218]	Oui	Non	Symbolique pour la planification
Herb [233]	Non	Non	Symbolique
PR2 [82]	Non	KnowRob et K-CopMan	Symbolique
Justin [136]	Non	Non	Géometrique et Symbolique
Johnny [26]	Oui (Architecture hybride simple)	Non	Symbolique (Arbre de décision) pour la planification
Pioneer robot [144]	Oui (Architecture hybride simple)	Oui (Plateforme OUR-K)	Symbolique à base de cas
Spartacus [166]	Architecture MBA	Non	Symbolique (Arbre de décision) pour la planification
Drône de l'université de Linkoping suite du projet WITAS UAV [200]	Non (Reactive)	Non	Sur l'incertain et symbolique à partir de cas

TABLE 3.1 – Contributions de nos recherches

Dans l'état de l'art dressé dans ce chapitre, l'accent a donc été mis sur les architectures d'intégration des fonctions rendant le robot capable de comprendre la

représentation de connaissances à base d'ontologies et le raisonnement dans les systèmes robotiques. De nombreux projets existent dans le cadre de l'intégration de la compréhension dans l'architecture d'un robot. Ces projets portent sur des robots évoluant dans des environnements libres "Outdoor", par exemple le projet WITAS de l'université de Linkoping [64]. En Indoor, il existe des robots dotés de la capacité de compréhension tels que Care-O-Bot, HERB, PR2, Johnny, Jido, Rosie, Justin, DESIRE, OUR-K. En effet, le robot Care-O-Bot, par le biais de la fonction de perception, est capable de modéliser l'environnement en 3D. Lorsqu'il est près d'une porte (repérée par sa position), Care-O-Bot l'ouvre et passe (la fonction d'agir). Il ne fait pas de contrôle si la porte est mi-ouverte et qu'elle lui permet de passer (pas de contrôle d'exécution). Contrairement à Care-O-Bot, le robot HERB est incapable d'agir sur les obstacles (c'est-à-dire qu'il ne peut pas ouvrir une porte et il est obligé de relancer l'action à la rencontre d'un obstacle). De son côté, PR2 est capable de modéliser l'environnement (fonction de perception) et manipuler des objets (fonction de planification), cependant il est incapable d'agir sur les obstacles. S'il trouve un obstacle il planifie de nouveau sa trajectoire. Pour sa part, Johnny le serveur est incapable de modéliser l'environnement pour savoir quel plan mettre en œuvre pour attraper les objets désignés par l'opérateur (fonction raisonnement sur l'objectif). Le système robotique Justin qui, bien que muni de la technologie DESIRE, qui est l'une des premières technologies ayant utilisée la sémantique dans la robotique de manipulation, n'effectue pas de reconnaissance de scènes. Quant à la dernière version de la plateforme coréenne OUR-K, capable de modéliser l'environnement et de comprendre la scène ; certains manques peuvent être notés comme par exemple le fait que lors du déplacement d'un robot muni de OUR-K, le chemin pour aboutir à la scène est calculé au préalable et donc si un obstacle survient après que le robot soit lancé, le robot est incapable de changer de chemin ni de comportement (fonction agir).

Nous pensons qu'un système robotique intelligent doit être en mesure de générer un plan d'action afin de satisfaire l'opérateur, de contrôler son exécution et de réussir à affronter les difficultés rencontrées, tout en étant conscient des situations par lesquelles il passe. Ceci se traduit par son aptitude à détecter une situation de blocage (problème), effectuer l'analyse et l'interprétation de la situation blocage, stocker le résultat pour une situation future similaire (reconnaissance de situation) et générer un plan d'action qui englobe le plan initial adapté (déclenchement de comportements) pour résoudre ce problème.

Le tableau 3.1 récapitule la vision des systèmes étudiés par rapport à nos contributions envisagées et énoncées dans les synthèses élaborées ci-dessus, à savoir (1) l'intégration de l'ensemble des fonctions augmentant la capacité de compréhension (2) l'adoption d'une représentation de connaissances générique et (3) l'adoption d'un raisonnement augmentant la fiabilité de compréhension en effectuant une dualité entre un raisonnement déductif et un raisonnement par analogie.

Nous déduisons d'après ce qui précède, qu'en plus de ce qui est proposé dans les systèmes existants, l'approche que nous envisageons de mettre en place couvre et

intègre dans un même cadre sémantique toutes les fonctions délibératives intervenant dans la compréhension dynamique du contexte fondée sur un raisonnement dual qui tient compte des expériences passées.
Dans la partie suivante, nous présentons nos contributions pour appréhender la probématique posée dans le cadre de cette thèse.

Deuxième partie

COMPRÉHENSION DYNAMIQUE DU CONTEXTE DANS UN SYSTÈME ROBOTIQUE

Chapitre 4

APPROCHE RSAW POUR LA COMPRÉHENSION DU CONTEXTE EN ROBOTIQUE

Sommaire

4.1 Introduction

Les technologies de l'information et de la communication sont constamment en évolution, notamment dans le domaine pluridisciplinaire de la robotique. En effet, comme nous l'avons montré dans la première partie de ce manuscrit, plusieurs travaux de recherche ont été menés ces dernières années dans les domaines spécifiques de la cognition, l'interaction homme-robot, la navigation, la manipulation, la perception et les architectures logicielles des robots. Aussi, la sensibilité au contexte devient de plus en plus une caractéristique essentielle des systèmes informatiques ubiquitaires et intelligents en général et des systèmes robotiques plus récemment.
En effet, la sensibilité au contexte ne date pas d'aujourd'hui, elle a existé et a été utilisée depuis le début des années 1990 de différentes manières et dans différents domaines. En informatique, l'accent mis sur la sensibilité au contexte a évolué à partir des applications de bureau, en passant par les applications Web et l'informatique mobile, jusqu'à l'internet des objets (IoT, pour Internet of Things en langue anglaise) au cours de la dernière décennie.
Nos recherches se situent dans le cadre de l'intégration et la combinaison de ces avancées technologiques afin d'augmenter l'autonomie d'un robot par la compréhension du contexte dans lequel il évolue.
Dans ce sens, nous proposons une nouvelle approche permettant à un robot, évoluant dans un environnement dynamique, d'être sensible au contexte en le rendant conscient de la situation de blocage dans laquelle il se trouve, afin de la comprendre de façon autonome et trouver une solution pour la dépasser en vue de satisfaire le désir de l'opérateur.
Cette approche, dénommée **Robot Situation AWareness (RSAW)** propose un cadre d'analyse sémantique pour reconnaître, comprendre et réagir en cas de situation de blocage. Les retombées attendues d'une telle approche sont intéressantes, ceci nous a conduits à penser surtout à sa généralité et son intégration dans les systèmes robotiques.
Dans ce chapitre, nous commençons par définir la sensibilité au contexte en robotique puis notre approche pour améliorer la capacité de compréhension d'un robot afin de minimiser l'intervention de l'opérateur. La dernière section est réservée à l'intégration de notre approche dans un système robotique.

4.2 Sensibilité au contexte

L'utilisation du contexte est devenue plus populaire avec l'introduction du terme «informatique ubiquitaire» par Mark Weiser [254], alors que le terme «sensible au contexte» a été utilisé pour la première fois par Schilit et Theimer [228] en 1994. Depuis, la recherche sur la sensibilité au contexte s'est imposée et a été établie comme un domaine de recherche à part entière en informatique.
Dans cette section, nous étudions le contexte dans lequel un système robotique évolue

ainsi que les concepts nécessaires à sa compréhension.

4.2.1 Définition du contexte

Plusieurs travaux utilisant la notion de contexte dans plusieurs domaines de recherche, ont posé leur propre définition du contexte. La définition la plus commune est celle qui stipule que le contexte est un ensemble d'informations définissant l'environnement entourant une activité [95]. En psychologie cognitive, le contexte est défini en se basant sur la théorie de l'apprentissage chez l'humain [135] qui stipule que *"Le sens (meaning) d'un symbole ou d'un signe ne peut être établi d'une façon plus ou moins précise, que si ce processus de construction de sens est supporté par une information additionnelle venue de l'environnement qui entoure ce processus...le champ concret de construction d'un sens, dans un moment particulier, est généralement référencé par le contexte"* [247][69].
En informatique, plusieurs définitions existent. Elles peuvent être regroupées en deux catégories : Celles qui définissent le contexte par l'énumération des ensembles qu'il comporte [228][29] et les autres qui le définissent par l'abstraction et la généralisation de l'information [40][33][1].
La principale critique, que nous partageons, autour de la première catégorie se résume dans le fait que cette dernière est, d'une part très restrictive et d'autre part délimitée ou non valable pour toutes les applications [95]. Les chercheurs partisans de la deuxième catégorie proposent des définitions tournant autour de celle d'Abowd et Dey [1] stipulant que le contexte est *"toute information pouvant être utilisée pour caractériser* ***la situation d'une entité.*** *Une entité est une personne, un endroit ou un objet (physique ou applicatif) qui peut être considéré significatif à l'interaction entre l'utilisateur et l'application, incluant l'utilisateur et l'application eux-mêmes"*.
Dans nos travaux de recherches, nous ne proposons pas d'offrir une nouvelle définition du contexte mais de travailler avec la dernière définition, celle de Dey, la plus objective, la plus générique et que plusieurs chercheurs ont également adopté. En se basant sur cette définition, il apparaît que pour être sensible au contexte, la situation requiert une importance significative. C'est ce qui a conduit notre réflexion à centrer nos recherches sur la notion de situation. Ainsi, dans nos travaux, la compréhension du contexte implique la compréhension des situations dans lesquelles se trouve le robot. En particulier lorsque ce dernier se trouve bloqué, la compréhension des situations de blocage est vitale.
Par conséquent, dans ce qui suit nous nous focalisons sur la notion de situation en vue de la définir et la caractériser.

4.2.2 Notion de situation

Dans cette sous-section, nous présentons la définition de la situation, les concepts de la situation ainsi que les catégories de situation considérées dans cette thèse.

4.2.2.1 Définition d'une situation

La situation a été introduite dans plusieurs domaines tels que le domaine de la psychologie, du management, ou encore de l'informatique. Dans chacun de ces domaines, plusieurs définitions sont données. Ces définitions sont généralement spécifiques aux applications dans lesquelles les situations sont mises en œuvre. Dans notre cas, nous avons plutôt cherché à nous rapprocher d'un consensus assez large en adoptant une définition générale telle que celle donnée dans Larousse [151], qui définit la situation comme étant la "Manière dont quelque chose, un lieu est placé par rapport à d'autres choses, d'autres lieux".
De cette définition découle celle adoptée dans notre recherche et que nous centrons sur le robot. En effet, pour nous, la situation d'un robot à un instant donné représente la sémantique associée à (1) l'état du robot défini par les caractéristiques pertinentes de ses composants mécaniques, (2) l'état de son environnement et (3) celui de son opérateur exprimé par son profil et son désir. Dans une situation, l'état de l'environnement est défini par la localisation du robot ainsi que les objets et leurs emplacements dans la scène. La scène correspond à l'espace visible par le robot.
Durant son évolution dans l'environnement, conformément à un plan d'action pour répondre au désir de l'opérateur, le robot passe par différentes situations dans de brefs intervalles de temps. Ce passage est occasionné par une modification substantielle des valeurs associées à un des états relatifs à la situation. Ce changement de situation est lié à l'arrivée d'un événement. En effet, un événement est défini dans la littérature comme étant un stimulus auquel l'entité doit répondre. La réponse est faite par le déclenchement d'une action qui permet la transition d'une situation à une autre. L'événement est donc le déclencheur d'actions.
Dans un environnement imprévisible, il est probable que le robot se trouve dans une situation de blocage non prévue initialement dans son plan d'action. Dans ce cas, il ne peut pas réussir l'action en cours pour passer à la réalisation de l'action suivante. L'intervention de l'opérateur est généralement exigée pour dépasser cette situation. Nous nous intéressons dans nos recherches à ce type de situation et considérons que l'événement associé à ce type de situation de blocage est une anomalie. Cet événement est alors appelé dans nos travaux "Evénement_Anomalie".

4.2.2.2 Concepts d'une situation

D'après ce qui précède, nous formalisons la définition de la situation comme un triplet composé par l'état du robot (R), l'état de son environnement (E) et celui de l'opérateur (OP). Soit :

$$\mathbf{S = (R, E, OP)}$$

Dans ce qui suit, nous développons ces concepts.

a État du robot

Comme vu dans la section 2.3.3.2 de l'état de l'art, les principales fonctionnalités dans les robots sont la manipulation (perçage, manipulation d'objet, tracking, ou autre,..) et la mobilité (navigation, marche, galope ou autre..).
Pour garder un aspect général de notre proposition, nous considérons qu'un robot, en concordance avec ses fonctions, est composé d'un bras robotisé pour la manipulation et/ou d'une partie servant à la mobilité. L'état du robot est ainsi défini par l'état de ses parties comme décrit ci-dessous.

Bras robotisé Les bras robotisés disposent d'un certain nombre de Degrés De Liberté (DDL) permettant de reconnaître leur position. Les bras les plus courants dans la littérature sont les bras à 6DDL. Ces bras à 6 DDL sont généralement utilisés dans la manutention, l'aide à la personne ou en télé opération. La Fig. 4.1 montre les DDL d'un bras robotisé.
Ainsi nous modélisons l'état courant du bras robotisé par un vecteur contenant les valeurs de chaque DDL. Le vecteur représentant l'état courant est :

$$V_{Bras} =< q_1, q_2, q_3, q_4, q_5, q_6 >$$

où q_i représente la valeur de la DDL i. Pour pouvoir effectuer la manipulation, un

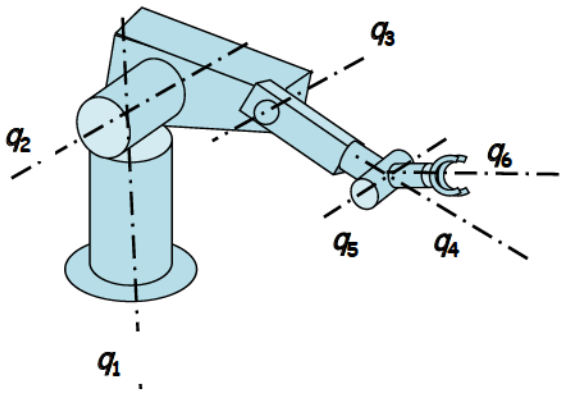


FIGURE 4.1 – Degré de liberté d'un robot

bras robotisé doit posséder un effecteur.

Effecteur Dans la littérature, il existe différents types d'effecteurs. Un effecteur peut être sous la forme d'une main à 5 doigts, d'une pince à deux ou 3 doigts, etc.. Nous concevons qu'un effecteur comporte au plus 5 doigts et représentons donc son état courant par un vecteur composé des angles (en degré) entre l'axe central (Z)

sortant de l'effecteur et les doigts de ce dernier.

$$V_{Pince} =< F_1, F_2, F_3, F_4, F_5 >$$

où $F_i \in [0, 50]$ représente la valeur de l'angle en degré fait par le i ème doigt avec l'axe central Z. La Fig. 4.2 montre un exemple de pince comportant 3 doigts. Dans ce cas, les valeurs associées aux deux doigts (F_4, F_5) manquants sont à 0.

FIGURE 4.2 – Doigts d'un robot : Exemple du bras Jaco de Kinova

Partie mobile La partie mobile permet à un robot de se déplacer et changer de localisation dans l'environnement. Pour la caractériser, nous avons défini les concepts généraux suivants :

- Le niveau de la batterie est un pourcentage
- La distance entre la partie mobile et sa destination est un nombre réel mesuré en centimètre
- La vitesse de mobilité est un nombre réel mesuré en m/s

$$V_{Mobile} =< M_{Battery}, M_{Distance}, M_{Velocity} >$$

b État de l'environnement

L'environnement a diverses significations selon le cadre dans lequel il est utilisé. Dans nos travaux, l'intérêt est porté sur l'environnement robotique, défini comme étant l'ensemble des éléments entourant l'entité (Robot) et dont un sous-ensemble contribue directement à subvenir aux besoins de cette entité.
Nous considérons que l'environnement robotique est défini par la scène visible par les caméras du robot. La scène est constituée par l'ensemble des objets physiques se trouvant à la portée des capteurs de l'entité robot.

Ainsi, l'état de l'environnement du robot concerne les objets se trouvant devant le robot (la scène) ainsi que sa localisation dans l'environnement. L'état de l'environnement est donc défini par :

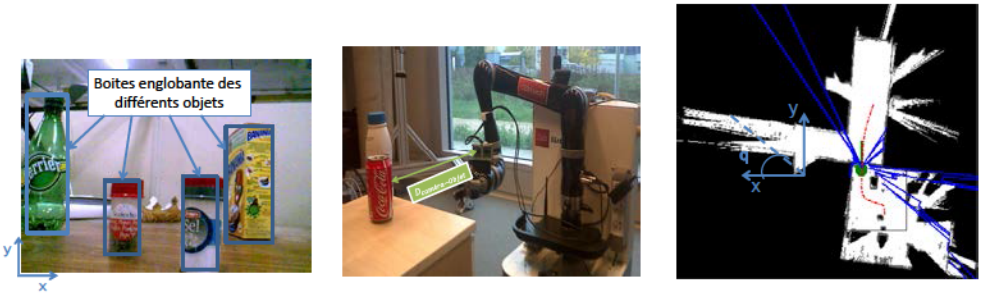
- Le vecteur de l'état des objets. Il est défini par :
  - Le nom de l'objet défini par un symbole représentant la sémantique de l'objet telle que représentée dans le système de reconnaissance du robot.
  - La boite englobant l'objet dans l'image de la scène $(X_{min}, X_{max}, Y_{min}, Y_{max})$. Un exemple de scène durant la phase de manipulation est donné dans la Fig. 4.3)
  - La distance entre la caméra et l'objet donnée par $D_{Camera-Objet}$.(Fig .4.3)
- Le vecteur représentant l'emplacement du mobile dans la carte de l'environnement.

$$V_{location} =< X, Y, \Theta >$$

. Où :

  - Dans un environnement extérieur (Outdoor) : X et Y sont les coordonnées GPS correspondant respectivement à la latitude et la longitude de la position du robot. Θ correspond à la direction du déplacement.

  - Dans un environnement intérieur (Indoor) : X et Y sont les coordonnées correspondant aux coordonnées cartésiennes du robot dans la carte de l'environnement, et Θ correspond à l'orientation du robot par rapport à l'axe X comme le montre la Fig. 4.3 illustrant un exemple d'environnement Indoor.



(a) Boite englobant l'objet ; Exemple d'une scène réelle contenant quatre objets

(b) Exemple de carte de l'environnement et le repère selon lequel la localisation de la base mobile est déterminée

(c) Exemple de carte de l'environnement et le repère selon lequel la localisation de la base mobile est déterminée

FIGURE 4.3 – Exemple de l'état de l'environnement

c État de l'opérateur

Nous considérons dans nos recherches que l'état de l'opérateur est matérialisé par son profil précisant des caractéristiques qui lui sont propres telles que ses paramètres

d'accès, son niveau d'expertise à manipuler les robots ainsi que par la tâche qu'il désire que le robot accomplisse. Une telle tâche se traduit par une mission attribuée au robot, comme par exemple les missions ≪Ramener un objet se trouvant dans une localisation ≫ dans le domaine de l'assistance aux personnes dépendantes, ou ≪ Mettre Objet1 dans Bac1 et Objet2 dans Bac2 ≫ dans la manutention ou bien "Déposer Bistouri sur la peau" en co-manipulation dans le domaine de la chirurgie.

Dans ce type d'exemples que nous considérons dans ce travail, le robot doit se déplacer pour manipuler un objet. Ainsi, nous modélisons le désir de l'opérateur par le couple ci dessous :

$$< objet, localisation >$$

. Si la tâche affectée au robot concerne uniquement la fonction de mobilité, l'état de l'opérateur sera défini uniquement par une instance du vecteur de localisation. Si elle concerne uniquement la fonction de manipulation, l'état de l'opérateur sera défini uniquement par une instance de l'objet (nom, localisation dans la scène) à manipuler. Après avoir présenté les concepts d'une situation, nous nous focaliserons dans la suite sur les situations de blocage.

4.2.3 Notion de situation de blocage

Dans cette section, nous définissons la situation de blocage (SB) ainsi que les catégories de SB traitées dans cette thèse.

4.2.3.1 Définition d'une situation de blocage

Un système robotique, selon sa vocation, évolue dans son environnement pour répondre aux désirs des opérateurs. Ce système robotique peut disposer d'une partie mobile pour assurer les actions de déplacement (navigation), d'un bras robotisé et/ou d'un effecteur pour assurer des actions de manipulation, et de capteurs extéroceptifs pour assurer les actions de perception.
Au cours de son évolution dans l'environnement, un système robotique, en exécutant un plan d'actions préalablement établi, peut rencontrer des obstacles ou des problèmes qu'il peut éviter ou résoudre à travers les fonctions de contrôle de bas niveau (cf. section a) et un générateur de plan d'action. Lorsque ces fonctions ne permettent pas au robot d'achever l'action courante, le plan d'action du robot est bloqué.

En robotique, la notion de situation de blocage est introduite particulièrement dans le cadre des multi-robots [193][214], lorsque typiquement, le blocage d'un robot est causé par l'autre, et que le planificateur est incapable de trouver une solution pour la résoudre. Dans ce cadre, une situation de blocage est définie comme étant une situation dans laquelle le planificateur ne peut pas trouver de solution, même s'il en existe. En informatique, un programme qui s'exécute (processus) peut se trouver

bloqué, ou en situation de blocage, quand il est dans l'incapacité, pour une raison logique, de poursuivre son exécution [1] [89].
Dans notre cas, nous nous intéressons à un robot qui doit exécuter un plan d'action connu pour répondre au désir de son opérateur (consistant à « ramener un objet qui se trouve dans l'environnement »).
Définition :
Le système robotique se trouve bloqué lorsqu'il n'arrive pas à terminer l'exécution d'une action. De ce fait, le système robotique ne passe pas à l'état suivant impliquant le commencement de l'action suivante. Il se trouve alors face à une situation imprévue dans le plan d'action.
Considérant que le plan d'action d'un tel système robotique est un programme informatique particulier, de ce fait, nous proposons d'appeler « situation de blocage » toute situation imprévue rencontrée par le système robotique.

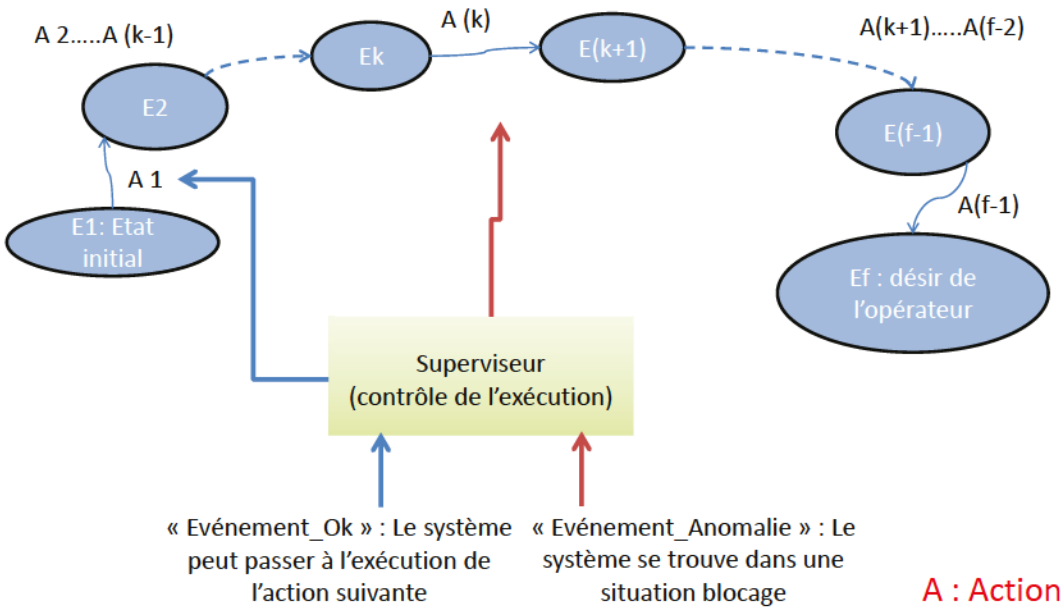


FIGURE 4.4 – Blocage d'un système robotique

La Fig. 4.4 illustre le blocage d'un système robotique, entre deux états (Ek, E(k+1)), à l'action (A(k)). Ceci est détecté grâce à l'arrivée de l'événement "Evenement_Anomalie" au superviseur (contrôle d'exécution). Durant un passage sans

1. http ://www.lb.refer.org/chebaro/page2_11.htm

blocage entre deux états (action aboutie), le contrôle d'exécution reçoit un "Evenement_OK" .

4.2.3.2 Catégories des situations de blocage

Comme présentée plus haut, une situation de blocage est provoquée par le passage à une situation imprévue dans le plan d'action en cours d'exécution, impliquant le blocage du robot.
Pour structurer le traitement correspondant aux anomalies détectées, une catégorisation des situations de blocage est faite en fonction de leur provenance.

La Fig. 4.5 montre la classification (considérée dans cette thèse) des situations de blocage, selon la cause du blocage. Nous nous sommes intéressés aux trois catégories :

- Blocage causé par l'"**Etat du robot**" : dans ce cas, le blocage peut être dû à un problème rencontré soit au niveau de la fonctionnalité de mobilité (roues, jambes ou pattes bloquées, batterie faible), soit au niveau du bras du robot (Bras cassé, préhenseur bloqué)

- Blocage causé par l'"**Environnement**" : dans ce cas, le blocage peut être constaté pendant la navigation (porte fermée, carte modifiée) ou pendant la manipulation dans une scène bloquante (existence d'un objet devant, sur/dans, à droite/à gauche de l'objet désiré)

- Blocage causé par une "**Incompréhension**" : dans ce cas, le blocage peut être constaté lorsque les objets de la scène sont non connus ou lorsque le robot ne comprend pas l'objet désiré par l'opérateur.

Après avoir présenté la terminologie et les concepts préalables adoptés dans notre recherche, il apparaît que la notion de situation de blocage joue un rôle important dans la compréhension du contexte, notamment lorsque le robot évolue dans un environnement complexe et imprévisible. C'est ainsi que nous avons voulu mettre au cœur de notre réflexion la relation que peut avoir le robot avec les situations critiques dans lesquelles il peut se trouver en vue de lui permettre de les comprendre et d'agir en conséquence.
Dans la littérature, nous avons été interpelés par un terme très important dont sa définition et son domaine d'application nous ont permis d'établir notre approche de compréhension. Ce terme est "Situation Awareness" ou SA. Dans la recherche menée dans le cadre de cette thèse et en vue de rendre un robot plus autonome, nous proposons une approche qui vise à rendre un robot conscient de la situation.
Pour ce faire, nous avons élaboré l'approche RSAW que nous présentons dans la section suivante.

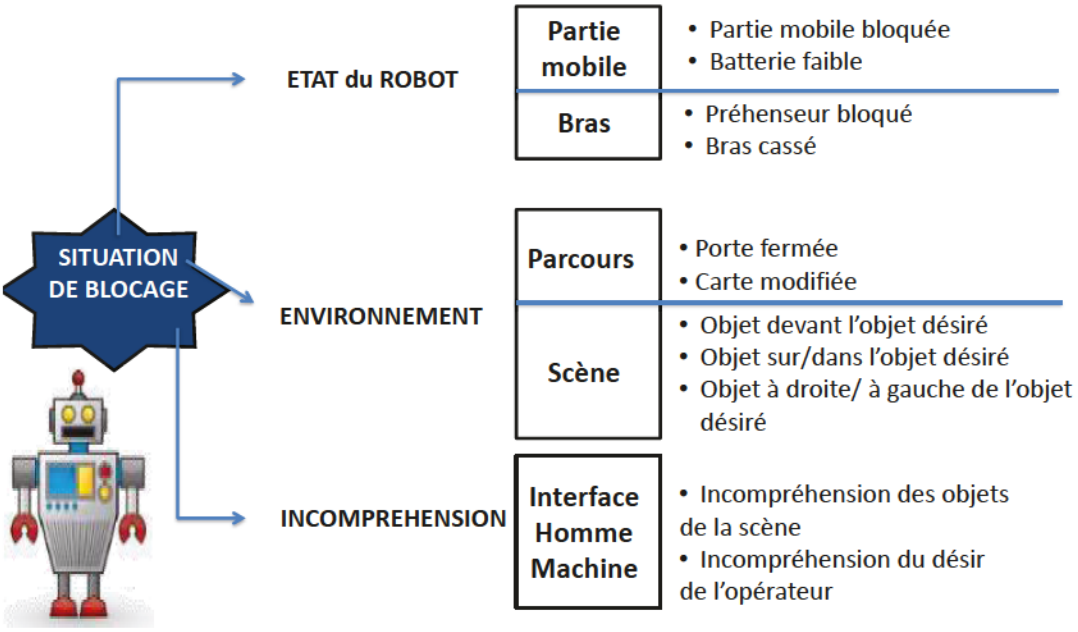


FIGURE 4.5 – Catégorie des situations de blocage

4.3 RSAW : La conscience de situation pour les robots autonomes

Dans ce qui suit, nous abordons le principe du SA annoncé précédemment ainsi que le principe de notre approche nommée RSAW, Robot Situation AWareness qui vise à améliorer la capacité de compréhension des robots.

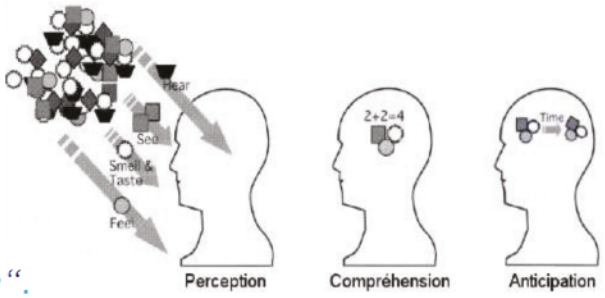
4.3.1 Situation Awareness (SA)

Dans cette sous-section, nous abordons le principe du SA.

4.3.1.1 Définition du concept de SA

Le concept de la "conscience de la situation", plus connue par son appellation anglophone "Situation Awareness" (SA), est reconnu, depuis plus d'une vingtaine d'années, en psychologie comme un élément clé des processus cognitifs en situation

"the perception of elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future".



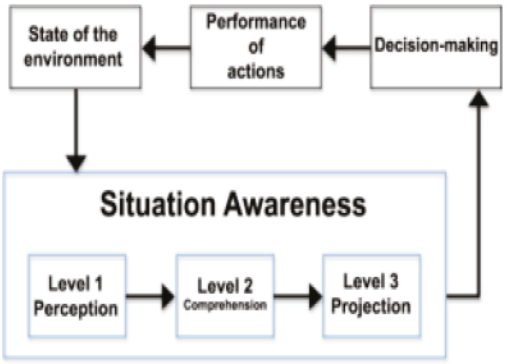


FIGURE 4.6 – Principe de SA [74]

dynamique.
SA est définie par Endsley [74] comme étant un modèle de processus pour la reconnaissance de la situation en cours. Comme le montre la Fig. 4.6, ce processus comprend trois activités ou niveaux organisés hiérarchiquement et temporellement : Le premier niveau concerne la perception des éléments de l'environnement dans un volume spatio-temporel, au deuxième niveau se fait la compréhension de leurs significations, le troisième niveau consistant en la projection de leurs états dans un avenir proche.
SA partage le travail entre l'opérateur et la machine. La machine collecte les éléments de la situation, l'opérateur comprend ces éléments pour les projeter dans le futur.

4.3.1.2 Applications de SA

Comme défini par Endsley [75], SA est un modèle permettant de disposer d'un système pour aider à comprendre ce qui se passe à l'instant (situation courante) et de prendre des décisions en conséquence. Un tel système a pu aider l'opérateur dans des conditions extrêmes dans différents domaines. En effet, d'après le site de l'entreprise

"Situation Awareness Technologies" [2] d'Endsley, SA a été appliqué dans les principaux domaines des systèmes électriques, des stratégies militaires (champs de bataille ; bombardier), de l'aviation, de la défense de la terre, du contrôle de trafic aérien, des transports, des systèmes de pilotage automatique, de la médecine, de l'énergie (Gaz et pétrole), des drônes, des interventions d'urgence, des systèmes de télé-opération, etc.
Les connaissances de la situation associée aux conditions extrêmes, calculées par un système basé sur SA, permet à l'opérateur de faire face, entre autres, aux conflits dans les systèmes où l'opérateur doit être précis et pas induit en erreur.

En particulier, des chercheurs en robotique ont exploré SA afin de faciliter les interfaces homme-robot comme dans les travaux [191], [184], [217] et [36]. D'autres travaux se sont intéressés aux drones pour les contrôler à travers des PDA "Personal Digital assistant" [211] ou pour évaluer l'activité des opérateurs des drones [65] [258]. D'autres expériences de SA consistaient dans le suivi d'activité des opérateurs de robots mobiles [34][3].

Bien que dans la majorité des applications SA organisent la compréhension de la situation courante, l'action de la machine se limite à la perception des éléments de la situation et c'est l'opérateur qui doit jouer un rôle primordial pour la compréhension des éléments et leur projection dans le futur. Cependant, un effort louable est fourni en robotique où l'autonomie des robots exige de faire moins appel à l'opérateur.
Nous pensons que le concept SA est plus puissant que cela puisqu'il implique une analyse cognitive des situations d'un système informatique en général et d'un système robotique en particulier. Cette analyse cognitive doit être basée sur un langage et doit permettre la modélisation des situations, leur compréhension ainsi que la prise de décision. Nous adoptons cette vision pour la résolution des situations de blocage décrite plus haut [146][76].

4.3.2 Principe de RSAW

Notre proposition RSAW est une instanciation du modèle de SA qui vise à réorganiser les tâches entre l'opérateur et le robot. RSAW vise, d'une part, la simplification du rôle de l'opérateur en robotique et d'autre part, de rendre possible la compréhension dynamique du contexte pour l'aide à l'opérateur en robotique. Pour répondre à ce principal objectif, l'intérêt est porté sur la compréhension des situations dans lesquelles le robot peut se trouver, plus particulièrement nous considérons les situations de blocage.
La généricité et l'évolutivité sont des caractéristiques recherchées lorsque l'on veut disposer d'une approche réutilisable dans différents contextes tout en assurant sa pérennité. C'est dans ce sens qu'a été élaborée l'approche RSAW pour l'intégration

2. https ://www.satechnologies.com/

de la capacité de compréhension permettant aux robots d'interpréter et de raisonner sur les situations de blocage pouvant être rencontrées.
En effet, **la généricité** dans RSAW est respectée grâce au paramétrage offert des caractéristiques de l'environnement, du robot et de son opérateur. De plus, le travail réalisé a considéré uniquement certains types de situations de blocage, **l'évolutivité** de RSAW permettra l'apprentissage de nouvelles situations.
Dans l'état actuel des choses, la principale intervention de l'opérateur en robotique se fait pour aider le robot à surpasser le blocage dans lequel il se trouve. Grâce à l'Interface Homme-Machine, l'opérateur indique ses besoins au robot qui génère et exécute de manière autonome le plan d'action correspondant afin de le satisfaire. Dans un environnement imprévisible, il est probable que le robot se trouve dans une situation de blocage. Dans ce cas, il ne peut pas réussir l'action en cours et passer à la réalisation de l'action suivante et l'intervention de l'opérateur est exigée pour dépasser le blocage.
Idéalement, un robot autonome doit être capable de remplir les actions pour lesquelles il a été programmé et se doit de surmonter seul toutes les situations de blocage qu' il peut rencontrer. Dans ce sens, RSAW se propose de réorganiser le partage des tâches entre l'opérateur et le robot pour minimiser son intervention. En suivant le modèle de

	Robot (Machine)	Opérateur (Homme)
Situation Awareness (SA)	Perception des éléments de la situation	Compréhension des éléments de la situation Projection des éléments de la situation dans le futur
Robot Situation Awareness (RSAW)	Perception des éléments de la situation Compréhension des éléments de la situation Projection des éléments de la situation dans le futur	Vérification de la compréhension de la situation Vérification de la projection des éléments de la situation dans le futur

TABLE 4.1 – Répartition des tâches entre l'opérateur et la machine dans les approches SA et RSAW

SA, l'approche RSAW vise à obtenir un robot conscient de la situation courante. Ce

qui veut dire, le rendre capable de reconnaître les données capteurs, de les comprendre et d'adapter ou re-planifier un plan d'action pour surpasser les situations de blocage afin d'accomplir l'objectif fixé par l'opérateur. Le rôle de l'opérateur se réduit alors au contrôle et à la vérification des différentes phases suivies par la machine. Le Tableau. 4.1 illustre le partage des tâches entre l'opérateur et le robot (Machine) proposé par RSAW par rapport à SA.
RSAW propose une démarche à suivre pour contrôler et traiter les situations de blocage que peut rencontrer un robot qui évolue au cours du temps dans un environnement dynamique. Ce processus illustré dans la Fig. 4.7, utilise les données issues

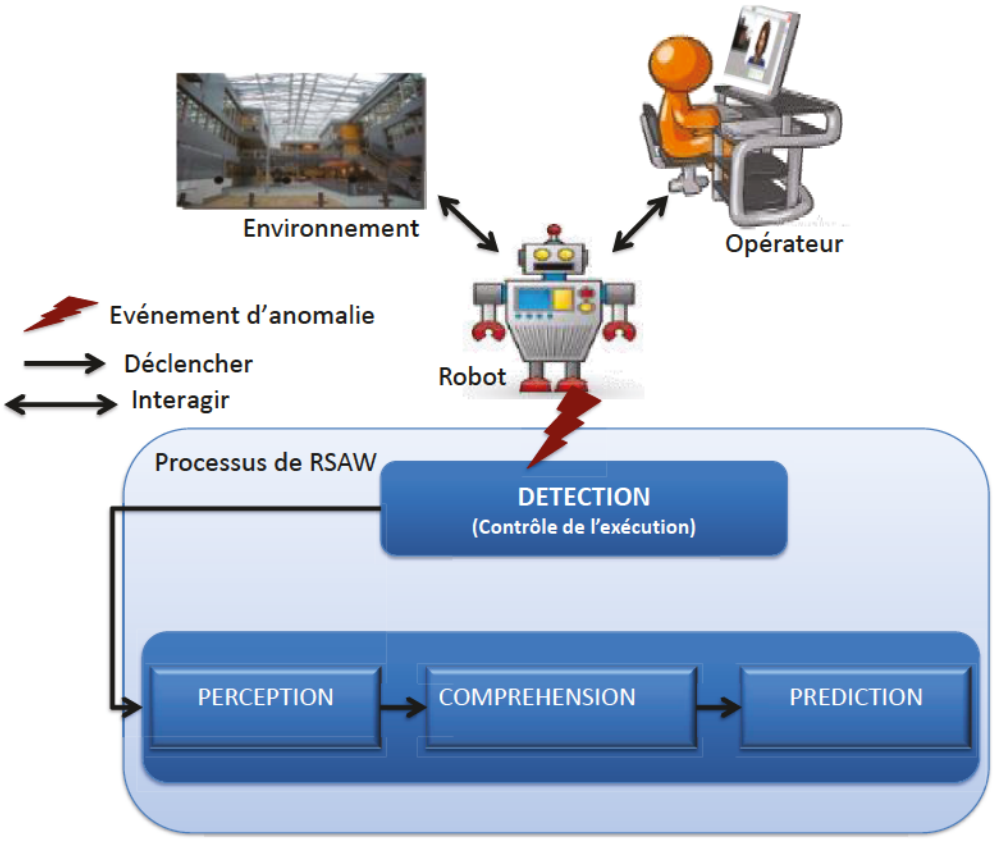


FIGURE 4.7 – Processus de RSAW et son interaction avec le contexte du robot

des capteurs du robot ainsi que ses modules logiciels de base tel que le planificateur, le séquenceur, etc... Il présente, en plus des phases de SA (perception, compréhension et projection), une phase préalable de détection qui contrôle l'exécution du plan d'action par le robot et l'évolution de l'environnement. Cette phase a été ajoutée dans RSAW comme déclencheur de la phase de lancement de l'instance de SA pour optimiser le temps de calcul. En effet, la perception, la compréhension et la projection d'une si-

tuation ne sont opérées que lorsque le robot se trouve dans une situation de blocage. Ainsi, un robot équipé avec RSAW offre la possibilité d'augmenter son autonomie par la détection des situations de blocage, puis leur traitement (perception-compréhension-projection). Le traitement des situations de blocage dans RSAW, d'après notre compréhension de SA et afin de rendre la compréhension des situations dynamique, se manifeste dans trois phases :

- La perception ou la reconnaissance des données capteurs. Cette reconnaissance consiste en la transformation des données capteurs (signaux) en des symboles. Par exemple transformer l'image de la scène en des symboles représentant les objets se trouvant dans l'image.

- La compréhension qui consiste en l'interprétation des symboles fournis par la perception et la préparation de la prise de décision pour le déblocage du robot. Par exemple trouver les relations spatiales existantes entre les objets de la scène.

- La projection consiste en l'analyse du résultat fourni par la phase de compréhension. Elle peut proposer l'adaptation de la solution (plan d'action existant) d'une situation passée similaire ou la re-planification du plan d'action (nouveau plan d'action).

Les phases, décrites précédemment, sont réalisées par le robot pour augmenter son autonomie et surpasser les situations de blocage. Ce processus est entièrement exécuté par le robot et le rôle de l'opérateur est restreint. C'est ainsi que l'approche RSAW aide l'opérateur en robotique.

4.4 Intégration de la capacité de compréhension dans les robots autonomes

L'intérêt de ce travail de thèse est de rendre un robot plus autonome en lui ajoutant la capacité de compréhension. Dans ce sens, RSAW utilise les capacités du robot et les complète par la capacité de compréhension. Son automatisation devrait, par conséquent, favoriser son intégration dans un système robotique.
Comme vu dans l'état de l'art, plusieurs plateformes d'intégration logicielle de systèmes robotiques ont vu le jour (comme ROS, OROCOS, YARP,...). Seulement la plupart d'entre elles sont incompatibles les unes avec les autres pour différentes raisons, dont essentiellement l'utilisation de protocoles de communication spécifiques, des codes sources propriétaires, des systèmes d'exploitation différents, des concepts architecturaux différents, etc. [187].
Nous pensons que pour être facilement intégré dans une plateforme robotique, un composant doit être conçu et construit de façon à satisfaire certaines caractéristiques de qualité telles que l'adaptabilité, la portabilité et la réutilisabilité.

Dans ce sens, notre approche RSAW se veut générique et intégrable dans une plateforme spécifique ou hétérogène, en minimisant le coût d'intégration. De ce fait et étant donnée la complexité des systèmes robotiques, nous avons étudié la problématique d'intégration de RSAW au niveau conceptuel/architectural, indépendamment de toute implémentation spécifique. C'est ainsi que l'intégration de RSAW dans une plateforme se fait, d'une part, en considérant son agencement et ses interactions avec les composants existants et entre les niveaux de l'architecture du système robotique, et d'autre part, en adoptant une représentation des connaissances unique favorisant la compréhension. La structure des connaissances devra garantir la cohérence tout au long du processus de RSAW et une véritable intégration sémantique des informations considérées. Elle sera donc alimentée par certains composants et exploitée par les autres.
Dans ce qui suit, nous présentons notre approche pour l'intégration de la capacité de compréhension, matérialisée par RSAW, dans un robot.

4.4.1 Architecture d'intégration de RSAW

Les architectures, existantes aujourd'hui dans l'état de l'art, conçues pour les robots autonomes et qui illustrent bien l'évolution dans ce domaine ont le mérite d'avoir structuré les systèmes robotiques. Deux catégories d'architecture sont retenues actuellement. Il s'agit des :

- Architectures en couches, où chaque couche regroupe les modules logiciels du robot selon leur objectif (délibératif/décisionnel, fonctionnel ou de contrôle). Le nombre de couches varie selon les approches établies par leur constructeur. Il s'agit de l'architecture la plus utilisée. On la retrouve par exemple dans l'architecture LAAS (LAAS Architecture for Autonomous Systems) développée par le LAAS/CNRS et qui propose trois couches [116], et CLARAty (Coupled Layer Architecture for Robotic Autonomy) développée par NASA JPL et CMU [250] basée sur deux couches.

- Architectures multi-agents, où un agent peut être un module fonctionnel, un planificateur, un système de diagnostic, etc. Cette architecture est surtout utilisée par l'architecture IDEA proposée par l'équipe de la NASA Ames [179].

Certes, chaque architecture a essayé de considérer l'autonomie à partir d'un certain angle. L'architecture en couches met surtout l'accent sur les fonctionnalités alors que les systèmes multi-agents mettent au centre le décisionnel.
Comme vu précédemment, notre approche RSAW regroupe dans un processus les fonctions nécessaires pour la compréhension des situations. Ces fonctions doivent intéragir avec les composants logiciels de base du système robotique.
C'est ainsi que l'intégration de RSAW dans ces architectures exige de disposer de deux

composants logiciels importants : Le générateur de plan d'action et le séquenceur des tâches sont décrits dans la suite.

4.4.1.1 Générateur de plan d'action

La génération d'un plan d'action est faite au moyen du planificateur. En effet, la génération de plan d'action ou planification de tâches dans le domaine de l'IA s'occupe de la résolution des problèmes dans un domaine donné. La solution aux problèmes est le plan d'action qui est une suite d'actions à effectuer pour accomplir l'objectif. L'entrée du planificateur est composée de deux fichiers comme illustré dans 4.8.

- Le domaine contient :
  - Prédicats : Les propriétés pouvant être appliquées aux objets.
  - Actions : Les actions à effectuer pour atteindre le but.
- Le problème contient.
  - Objets : Les objets du monde.
  - État initial : L'état par lequel on commence.
  - Spécification du but : Le but du scénario.

Le schéma de planification est décrit dans la Fig. 4.8. Dans les architectures à couches,

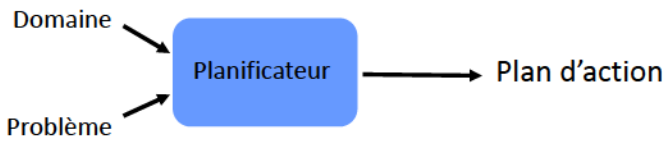


FIGURE 4.8 – Schéma de planification

le générateur de plans d'action est situé en général au niveau délibératif (décisionnel). Dans les architectures Multi-Agents, il correspond à l'agent planificateur.

4.4.1.2 Séquenceur de tâches

Le séquenceur envoie les ordres au robot qui fournit en retour des informations sur son état. Un plan d'action constitue la description globale de la tâche complexe à réaliser, comme par exemple "aller chercher à boire dans la cuisine". Il se décompose en séquences. Chaque séquence correspond à une tâche plus ou moins simple à effectuer telle que "se déplacer", "manipuler un objet".
A chaque séquence est associée une entité active appelée **agent** qui va contrôler la progression d'une séquence et nous indiquer dans quel état se trouve le système. Ces agents peuvent agir en parallèle, du moment qu'ils n'utilisent pas la même ressource (par exemple deux agents ne peuvent ordonner au bras de bouger dans deux directions

différentes, mais un agent contrôlant le bras et un autre la base peuvent s'exécuter en même temps).
A chaque séquence peut être associée une ou plusieurs actions. Les **actions** vont modifier l'état général du système. Elles font référence à des fonctions codées dans le niveau réactif ou fonctionnel du robot.
Les **évènements** sont porteurs d'informations instantanées. Ils peuvent intervenir dans plusieurs séquences différentes. Ils peuvent servir à exécuter une action dans un état donné qui ne doit être exécutée que si un événement particulier se produit. Ils servent également pour passer d'un état du robot à un autre. Dans ce cas, une transition dédiée est associée. Une **transition** est une connexion unidirectionnelle reliant deux états.
Dans les architectures à couches, le séquenceur est situé en général au niveau de supervision (contrôle de l'exécution). Dans les architectures Multi-Agents, chaque agent possède son propre séquenceur.

4.4.2 Intégration de RSAW

La première phase, "Détection" dans RSAW, contrôle l'exécution du plan d'action géré par le séquenceur du robot ainsi que l'évolution de l'environnement. Elle vérifie périodiquement les caractéristiques nécessaires à l'apparition d'un "Evenement_Anomalie" indiquant que le robot est dans une situation de blocage.
Cet événement peut être détecté dans l'un des deux modes de fonctionnement du robot (la manipulation et/ou le déplacement) :

- Durant la manipulation, lorsque le robot ne trouve pas l'objet demandé par l'opérateur ou bien lorsqu'il localise l'objet à saisir, mais ne peut pas l'atteindre car il est gêné par un autre objet (se trouve devant, en haut, à droite ou à gauche de celui-ci).
- Durant le déplacement lorsque le robot est en mouvement et rencontre un obstacle qu'il ne peut pas surmonter avec son logiciel d'évitement d'obstacle.

Cette première phase de RSAW doit être en étroite collaboration avec le séquenceur (voir section 4.4.1.2) afin qu'elle puisse détecter les événements engendrés par le blocage du robot.

La conscience de la situation dans RSAW comporte les phases nécessaires (les phases perception-compréhension-projection) pour comprendre les situations de blocage et préparer la génération de nouveaux plans d'action pour les surpasser. D'une part, ces phases ne peuvent pas être dissociées et s'enchainent pour former un composant unique et d'autre part, la phase de projection interagit avec le générateur de plan d'action du robot, présenté dans la section précédente.

Dans ce qui suit, nous présentons l'intégration de RSAW dans une architecture à couches ainsi que dans une architecture Multi-Agents.

4.4.2.1 Intégration de RSAW dans une architecture à couches

Dans les architectures à couches, le niveau dans lequel la détection de RSAW est opérationnelle est le niveau de supervision (contrôle d'exécution). Le composant

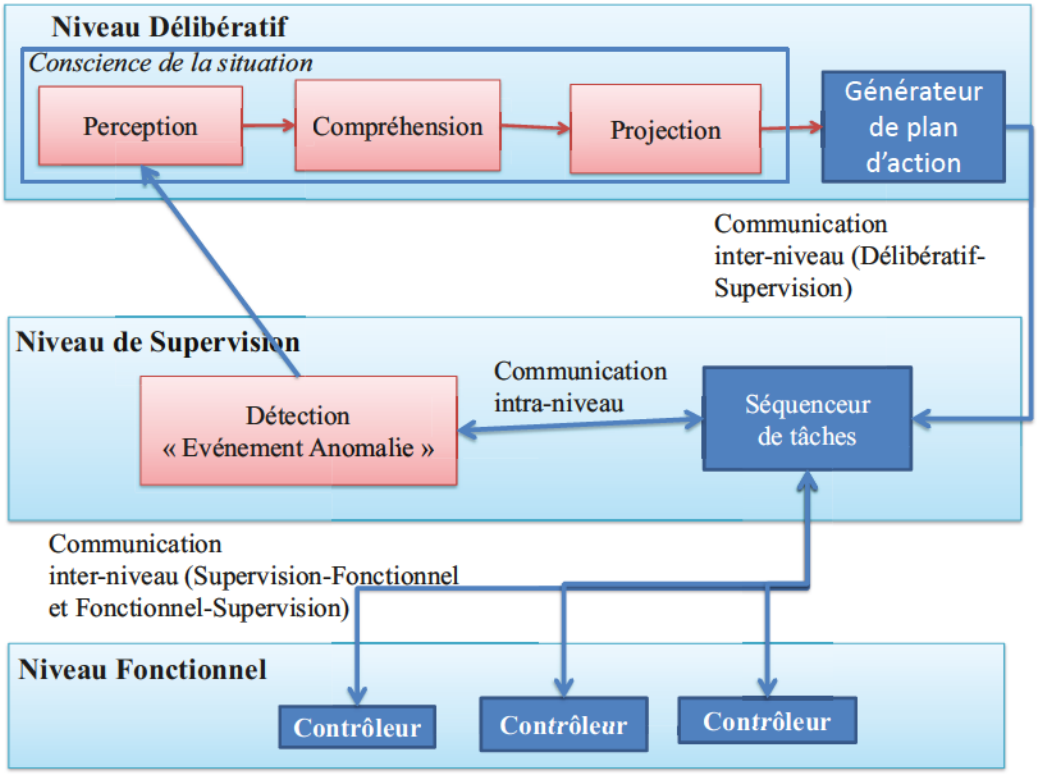


FIGURE 4.9 – Intégration de RSAW dans une architecture à couches

"conscience de la situation" de RSAW est représenté par les phases de percetion, de compréhension et de projection. Ce composant est donc inscrit dans les architectures à couches au niveau délibératif. Il communique avec le générateur de plan d'action pour l'établissement d'un nouveau plan d'action qui sera fourni au séquenceur de tâche pour la supervision. Il devient opérationnel par la réception de "Evenement_Anomalie" du composant "détection".
La Fig. 4.9 montre l'intégration des différentes phases de RSAW dans une architecture à couches.

4.4.2.2 Intégration de RSAW dans une architecture Multi-Agents

Dans une architecture Multi-Agents, la phase de détection doit être déployée au sein de chaque agent pour qu'elle puisse prendre en compte leurs caractéristiques

spécifiques.
Le composant "conscience de la situation" est conçu, dans ce type d'architecture, comme un agent à part entière qu'on nomme agent délibératif. Il communique avec le composant "détection" de chaque agent. La Fig. 4.10 montre l'intégration des

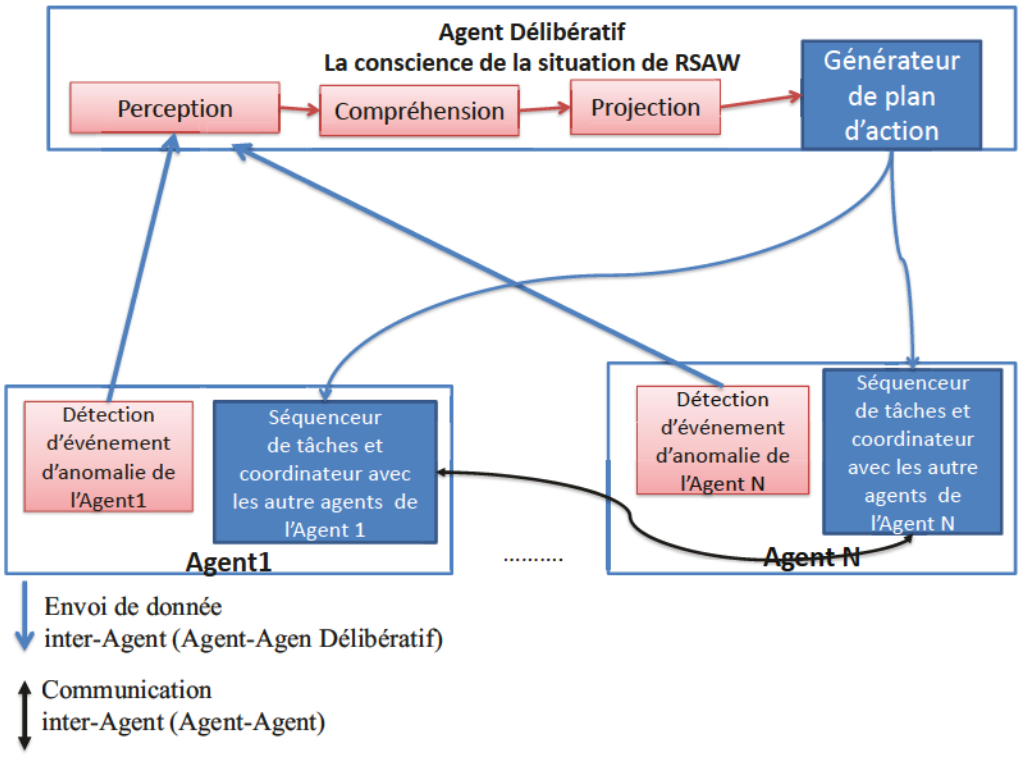


FIGURE 4.10 – Intégration de RSAW dans une architecture Multi-Agents

différentes phases de RSAW dans une architectures Multi-Agents. Dans la figure, nous notons les agents "Agent 1" jusqu'à l'"Agent N" des agents appartenant à un système Multi-agents et occupant des tâches précises dans l'environnement.

4.5 Conclusion

Ce chapitre présente notre principale contribution à la compréhension dynamique du contexte pour l'aide à l'opérateur en robotique. Il s'agit de l'approche, baptisée Robot Situation AWareness (RSAW), visant à rendre un robot conscient de la situation de blocage dans laquelle il se trouve pour la comprendre et la surpasser. RSAW s'inspire de la notion de conscience de la situation ou Situation Awareness (SA) reconnue comme un élément clé des processus cognitifs en situation dynamique. En

premier lieu, ce chapitre a défini la notion de contexte et sa relation avec la notion de situation dans nos travaux de recherche, puis il a présenté la notion de SA pour finir avec la présentation de RSAW et son intégration dans les architectures existantes de robots autonomes.

RSAW est déclenchée suite à la détection d'une situation de blocage. Après une première phase de reconnaissance, la phase cruciale de compréhension est entamée permettant par la suite au robot de prendre des décisions. La compréhension des situations constitue un élément fondamental dans RSAW, elle exige une bonne représentation des connaissances sur laquelle une approche de raisonnement sera exécutée pour faire intervenir le moins possible l'opérateur.
Dans les chapitres suivants la phase de compréhension est détaillée, d'abord en terme de représentation des connaissances, puis en termes de raisonnement.

Dans ce qui suit nous proposons une approche pour la phase de compréhension dans RSAW se manifestant par une représentation des connaissances favorisant la bonne compréhension de la situation.

Chapitre 5

REPRÉSENTATION DES CONNAISSANCES DANS RSAW

Sommaire

5.1 Introduction

La recherche menée dans le cadre de cette thèse, se concentre sur la capacité des robots à comprendre et à dépasser les situations de blocage de manière autonome en minimisant l'intervention de l'opérateur.
Comme présenté dans le chapitre précédent, notre approche générique, RSAW, permet à un robot d'être conscient des situations de blocage dans lesquelles il peut se retrouver durant son évolution dans un environnement dynamique et imprévisible.

Nous avons également montré comment RSAW peut être intégré dans un système robotique.
La conscience de situation est un processus d'activités comportant les fonctions délibératives (**perception** (reconnaissance), **compréhension** (interprétation, raisonnement sur l'objectif) et **projection**(planification et réaction)) garantissant une meilleure capacité de compréhension (cf. section autonomie de premier chapitre de l'état de l'art) à un robot pour aboutir à une prise de décision adéquate selon la situation de blocage.
L'activité critique de ce processus concerne la fonction de compréhension, en deux étapes, permettant à un robot d'interpréter les données de la situation de blocage et de préparer la prise de décision pour la surpasser en vue de répondre à son opérateur.
Cette activité de compréhension est considérée, en Intelligence Artificielle, comme une activité interprétative qui vise à construire un réseau de relations entre les différents concepts de la situation. Ce réseau dépend à la fois des connaissances de celui qui fait cette activité et des relations établies par la représentation de la situation.
Cette activité s'adonne ainsi à des tâches nécessitant une organisation de connaissances, un raisonnement conséquent et de l'apprentissage. La compréhension passe donc par la construction de modèles capables de réaliser des raisonnements.
Dans ce chapitre, nous dressons, tout d'abord, les connaissances à resprésenter ainsi que nos choix dans la représentation des connaissances et le formalisme de description et modélisation adoptée. Ensuite, nous présentons la représentation d'une situation dans RSAW ainsi que la modélisation des connaissances. Dans la modélisation des connaissances, nous décrivons les ontologies mises en place.

5.2 Connaissances à représenter : Langage symbolique commun dans RSAW

Pour permettre à un système robotique de reconnaître les données relatives aux situations lors de la détection d'une anomalie et de générer le comportement adéquat, notre approche RSAW offre la possibilité d'interpréter les données reconnues par le robot et d'en construire des connaissances sur lesquelles un raisonnement est mis en œuvre.
Ces connaissances représentent les valeurs associées aux concepts de la situation. L'idée est que, comme cela est indiqué dans la section 4.3.2, dans la phase de compréhension, l'interprétation traduit les données perçues dans un langage symbolique commun avant leur exploitation dans la préparation de la prise de décision. En effet, l'objectif des langages symboliques est de représenter des connaissances d'une façon homogène. Un tel langage doit permettre la spécification des conventions et des règles. C'est ainsi que nous avons conçu d'abord le langage, en construisant un lexique regroupant les symboles utilisés dans la description des concepts des situations de blocage (Etat du robot, Etat de l'environnement et Etat de l'opérateur) puis les relations entre ces

symboles permettant de décrire les liens sémantiques entre les concepts.
En fait, le lexique utilisé dans le langage symbolique commun est celui utilisé pour définir la syntaxe des concepts de la situation et de leurs propriétés (cf. annexe A.1). Il est à remarquer que ce lexique peut être enrichi pour permettre la description de nouvelles connaissances. Ce langage est défini par des symboles de base permettant la représentation de la connaissance relative à chaque concept de la situation comme définie dans le chapitre 4. En guise d'illustration, quelques exemples de symboles définis pour la représentation des concepts et leurs relations existantes entre eux sont schématisés ci-dessous :

1. Pour l'état d'un robot qui comporte un bras, une main et/ou une partie mobile, les symboles ont été définis pour chacune de ses parties. Parmi les symboles définis on trouve :
   - concernant les connaissances relatives à l'état du bras :
     - **Init_Position** indique que le bras est dans une position initiale : **Init_Position(Arm)**
     - **Transport_Position** indique que le bras est dans une position favorable à la mobilité du robot : **Transport_Position(Arm)**
   - connaissances relatives à l'état de la main :
     - **close** indique que la pince est fermée : **Close(Gripper)**
     - **open** indique que la pince est ouverte : **Open(Gripper)**
   - concernant les connaissances relatives à l'état de la partie mobile :
     - **at_Position** indique la position du robot associé à une connaissance relative à la localisation de l'environnement. **at_Position(Robot, Kitchen)** précise la relation qui associe la position du robot à une connaissance de l'environnement représentée par la localisation (ici dans l'exemple, Kitchen) ;
     - **In_Motion** indique si le robot est en mouvement ou pas : **In_Motion(Mobile)**.
2. Pour l'état de l'environnement qui comporte une localisation et des objets, les symboles ont été définis pour chacune de ses parties. Parmi les symboles définis on trouve :
   - les symboles relatifs à la localisation : **Kitchen** ; **Lunchroom**
   - les symboles relatifs aux objets exprimant les relations possibles entre les noms des objets réels de la scène, comme par exemple :
     - **isLeft(objet1, objet2)** indique qu'un objet1 est à gauche d'un objet2 dans la scène
     - **isFront(objet1, objet2)** indique qu'un objet1 est devant d'un objet2 dans la scène
     - **isBehind(objet1, objet2)** indique qu'un objet1 est derrière d'un objet2 dans la scène.
3. Pour l'état de l'opérateur qui est défini par son profil et son désir, les symboles utilisés pour exprimer par exemple son désir sont les noms des objets et leurs localisations, tels que définis dans le langage utilisé par l'IHM utilisée par l'opérateur.

- **operat or-Desire-Is(Operator,objetX)** indique que le désir de l'opérateur est l'objetX.

La représentation des connaissances adaptées se traduit par un besoin de disposer d'une vue d'ensemble cohérente suffisamment détaillée des principales connaissances et de leurs liens. Dans la suite nous présentons les choix que nous avons adoptés pour le faire.

5.3 Motivation et choix

Cette section s'intéresse aux différents choix adoptés pour la représentation des connaissances dans RSAW.

5.3.1 Choix de l'ontologie

Les systèmes robotiques déployés dans les différents environnements complexes sont confrontés aux problèmes de la représentation et du raisonnement sur des connaissances incomplètes du domaine, acquises à partir des données capteurs et de l'aide d'un opérateur. Une bonne représentation de connaissances devrait minimiser l'intervention de l'opérateur car elle permet de pallier l'absence et/ou le bruitage des données capteurs [21]. C'est ainsi que RSAW, dans une première phase de perception, reconnait la situation dans laquelle se trouve le robot en vue de son interprétation et de préparation à la prise de décision [260].
D'après la littérature autour de l'identification de la situation et l'étude faite au niveau de notre état de l'art (cf. section 3.3.1), nous retenons les deux types d'approches qui ont généralement été adoptés dans les systèmes informatiques. Il s'agit des approches basées sur les spécifications et celles basées sur l'apprentissage. Les concepteurs utilisant une approche basée sur les spécifications doivent tenir compte des données capteurs et des connaissances du domaine (opérateur ; environnement). Ceux utilisant les approches basées sur l'apprentissage appliquent des techniques de Machine Learning.
Le Tableau. 5.1 représente un récapitulatif des techniques utilisées dans les deux approches. Ce tableau montre que l'utilisation des approches basées sur l'apprentissage s'adaptent mieux à un contexte dans lequel les données capteurs sont incertaines ou lorsque l'association entre les situations et les données capteurs est complexe. Les techniques utilisées dans ce type d'approche sont généralement les réseaux bayésiens, les chaines de markov, les arbres de décision ou encore les réseaux de neurones.
Les approches basées sur les spécifications offrent quant à elles, dans un contexte de faibles nombre de capteurs, une interprétation facile des données ainsi qu'une association moins complexe entre la situation et les données capteurs. Les techniques utilisées dans ce type d'approche sont les réseaux sémantiques, les Frames ou encore les ontologies. Notre contexte s'adapte plus à l'utilisation d'une approche basée sur

Approches	Contexte-d'utilisation	Techniques
Approches basées sur l'apprentissage	• L'incertitude des-données-capteurs-est- très-élevée. • L'association-complexe entre situations et-données-capteurs	Réseau-bayésien Chaine de markov Etc..
		Arbre de décision Réseaux-de neurones Etc..
		Méthodes-stochastiques- Etc..
		Arbres-des-suffixes Etc..
Approches basées sur les spécifications	• Faible nombre de capteurs • Interprétation-facile des-données • Clarté entre les données-capteurs et-la situation	Programmation-logique
		Logique Spatio-temporelle
		Ontologies

TABLE 5.1 – Approches d'identification de situation

les spécifications du fait que le nombre de capteurs utilisés est limité et l'association entre la situation et les données capteurs est facilement identifiable. De plus, d'après la section 3.3.1 de l'état de l'art, l'utilisation des ontologies s'avère avantageuse dans le cas des systèmes robotiques. C'est ainsi que nous avons choisi de représenter les connaissances avec les ontologies. En effet, les ontologies offrent dans un même cadre une puissante représentation des connaissances du domaine ainsi qu'une possibilité de raisonnement. Elles favorisent la modularité, l'évolutivité et l'interopérabilité en fournissant un accès commun à l'information et une compréhension commune des concepts et permettent également la réutilisation des sources de connaissances. Aussi, les tâches de raisonnement sont moins coûteuses en termes de calcul lors de l'utilisation des ontologies [156].

5.3.2 Choix du formalisme

Plusieurs formalismes de représentation des ontologies ont été développés durant les dernières années. Dans notre cas, étant donnée l'hétérogénéité des environnements de développement utilisés dans les systèmes robotiques, nous avons fait le choix d'utiliser un formalisme standard pour faciliter l'intégration de RSAW dans un robot (cf. section 4.4). Les standards RDF, RDFS ou encore OWL qui favorisent les échanges des informations dans le web sémantique [220] peuvent être utilisés dans le contexte de nos travaux.
RDF pour Ressource Description Framework est un modèle de graphe destiné à décrire d'une manière formelle les ressources Web. Un document en notation RDF contient des triplets sous la forme (Sujet, Prédicat, Objet), l'identification des éléments de triplet est faite d'une manière unique par les URI (Uniform Resource Identifier).
RDFS pour RDF Schema est une extension de RDF au niveau de l'expression de la sémantique. Ce formalisme, écrit en RDF, offre des mécanismes pour des groupes de ressources connexes et les relations entre elles. Ces ressources sont utilisées pour déterminer les caractéristiques d'autres ressources, telles que les domaines et les propriétés [1].
OWL pour Ontology Web Language est un langage de représentation des connaissances construit sur le modèle de données de RDF. Ce formalisme est né à cause de la nécessité de modéliser des connaissances dans des domaines complexes pour pallier les limites de RDF et RDFS. Il fournit les moyens de définition des notions complexes comme la disjonction, la combinaison booléenne et les restrictions entre les classes [2].

Notre contexte ne nécessitant pas la définition de notions complexes, notre choix dans nos travaux s'est donc porté sur l'utilisation du formalisme RDFS qui permet d'avoir une conception compréhensible, manipulable et communicable, ce qui améliore la qualité de la conception. Pour l'extraction et la manipulation des connaissances décrites avec RDFS, nous utilisons le langage de requêtes SPARQL (SPARQL Protocol and RDF Query Language) [3].

5.3.3 Choix de modélisation

Une conception de qualité est importante pour la généralité, la portabilité, l'évolutivité et la pérennité des systèmes, telles que recherchées dans ce travail de thèse. Elle repose généralement sur différents modèles et paradigmes, à différents niveaux et différents points de vue. Nous avons fait le choix d'utiliser le paradigme orienté objet à travers le langage standard UML (Unified Modeling Language [221]) pour la modélisation dans RSAW.
Par ailleurs, nous souhaitons avoir la possibilité de manipuler, échanger, intégrer ou

1. http ://www.w3.org/TR/rdf-schema/
2. http ://www.w3.org/TR/owl-guide/
3. http ://fr.wikipedia.org/wiki/SPARQL

transformer les modèles de connaissances en vue de réduire les efforts et les coûts de leur développement et de leur intégration. Comme vu dans le chapitre précédent, nous souhaitons d'une part que RSAW soit indépendant le plus possible de l'environnement de développement du robot, et d'autre part, que les situations soient abordées d'un point de vue ou d'un autre (Robot, Environnement, Opérateur) avec la possibilité de mécaniser le raisonnement sur les connaissances représentées. Ceci exige une représentation de connaissances à différents niveaux d'abstraction, suffisamment précise pour être interprétée ou transformée de façon dynamique, tout en étant représentative du monde réel. Nous sommes donc confrontés à des transformations consécutives de modèles jusqu'à l'obtention d'un système exécutable dans un contexte spécifique. C'est ainsi que nous avons choisi d'adopter la méthodologie de l'Ingénierie Dirigée par les Modèles (IDM) ou encore Model Driven Engineering (MDE) qui offre une séparation entre le réel et l'architecture ainsi qu'une stabilité des algorithmes [16]. L'IDM est une forme d'ingénierie générative qui se distingue par une démarche dans laquelle tout ou une partie de l'application informatique est engendrée à partir de modèles. Elle offre la possibilité de :

- Concevoir des applications tout en ayant une abstraction des technologies cibles ;
- Assurer la durabilité des applications conçues ;
- Simplifier la maintenance et l'adaptation aux changements ;
- Augmenter la productivité ;
- Cibler les différentes plateformes d'exécutions depuis une seule conception ;
- Réutiliser ce qui existe ;
- Contrôler la simulation et les tests à différents niveaux.

Elle propose à travers les concepts de méta modèle et de transformation de modèles les moyens d'opérationnaliser la création et la manipulation des modèles. Un méta modèle est une description plus ou moins abstraite d'un langage de modélisation utilisé pour décrire un système. Sous la forme de règles définies au niveau des méta modèles, les transformations de modèles permettent de mettre en relation des modèles.

5.4 Représentation de la situation

La compréhension du contexte dans RSAW, comme vu dans le chapitre précédent, revient à la compréhension des situations de blocage par lesquelles passe le robot. Cette activité se focalise sur l'interprétation des données de la situation de blocage et la préparation de la prise de décision pour la surpasser. Les données de la situation ont été regroupées de manière à permettre le traitement de chacun de ses composants (Etat du robot, Etat de l'environnement et Etat de l'opérateur), selon son point de vue et de façon indépendante. C'est ainsi que nous nous sommes intéressés à la modélisation de la situation qui peut être abordée de différents points de vue impliquant différents niveau d'abstraction et traduisant une disposition des connaissances qui favorise leurs interactions.
Dans cette sous-section, nous abordons la modélisation de la situation dans RSAW

ainsi que l'interconnexion entre ces connaissances suivant l'IDM.

5.4.1 Modèlisation de la situation dans RSAW

Pour concevoir la représentation des connaissances, nous choisissons de partir de la définition de la situation donnée dans la section 4.2 puisqu'il s'agit du concept fondamental de RSAW.
Comme précisée au niveau des choix adoptés, la situation est modélisée au moyen du standard UML. La Fig. 5.1 représente le diagramme de classe UML des composants de la situation qui montre que la situation est constituée de l'état du robot, de l'état de son environnement et de l'état de l'opérateur. L'ensemble formé par la situation et ses composants est une source de connaissances interconnectées qu'il faut interpréter et utiliser dans un raisonnement adéquat pour aider à la génération de plan d'action. Ces connaissances sont représentées par quatre ontologies : S-Ontology pour représenter les concepts associés aux situations de blocage, R-Ontology pour représenter les concepts liés au robot, E-Ontology pour représenter les concepts liés à l'environnement et OP-Ontology pour représenter les concepts liés à l'opérateur. Ces ontologies sont organisées en niveaux conformément à la méthodologie de l'IDM.

5.4.2 Organisation des connaissances

Il est à constater que l'analyse d'une situation revient à l'analyser selon les trois points de vue état du robot, état de l'environnement et état de l'opérateur. Ceci implique que les modèles du robot, de l'environnement et de l'opérateur soient conformes au modèle de la situation. Ce dernier peut être considéré comme un méta modèle représentatif des trois autres modèles. L'application de l'IDM dans ce cadre distingue deux niveaux, le premier (M2 ou niveau méta modèle) contient l'ontologie relative à la situation S-Ontology. Le deuxième niveau (M1 ou niveau modèle) contient les ontologies R-Ontology, E-Ontology et OP-Ontology, ce qui garantit la conformité entre les modèles. La Fig. 5.2 illustre cette organisation de la représentation des connaissances dans RSAW.

5.5 Modélisation des connaissances dans RSAW

5.5.1 Niveau méta modèle : Ontologie de la situation (S-Ontology)

Comme vu précédemment, ce niveau contient l'ontologie des situations S-Ontology. Le principal concept de S-ontology est, comme son nom l'indique, la situation dont une modélisation est donnée dans la section 5.4.1. Cette modélisation fait ressortir les relations entre la situation et les états du robot, de l'environnement et de l'opérateur. Donc nous avons élaboré le paradigme de la situation qui comporte en soi les trois paradigmes : centré sur le robot, l'environnement et l'opérateur. La figure 5.3 illustre le méta modèle S-Ontology. Cette ontologie est composée de trois principaux concepts :

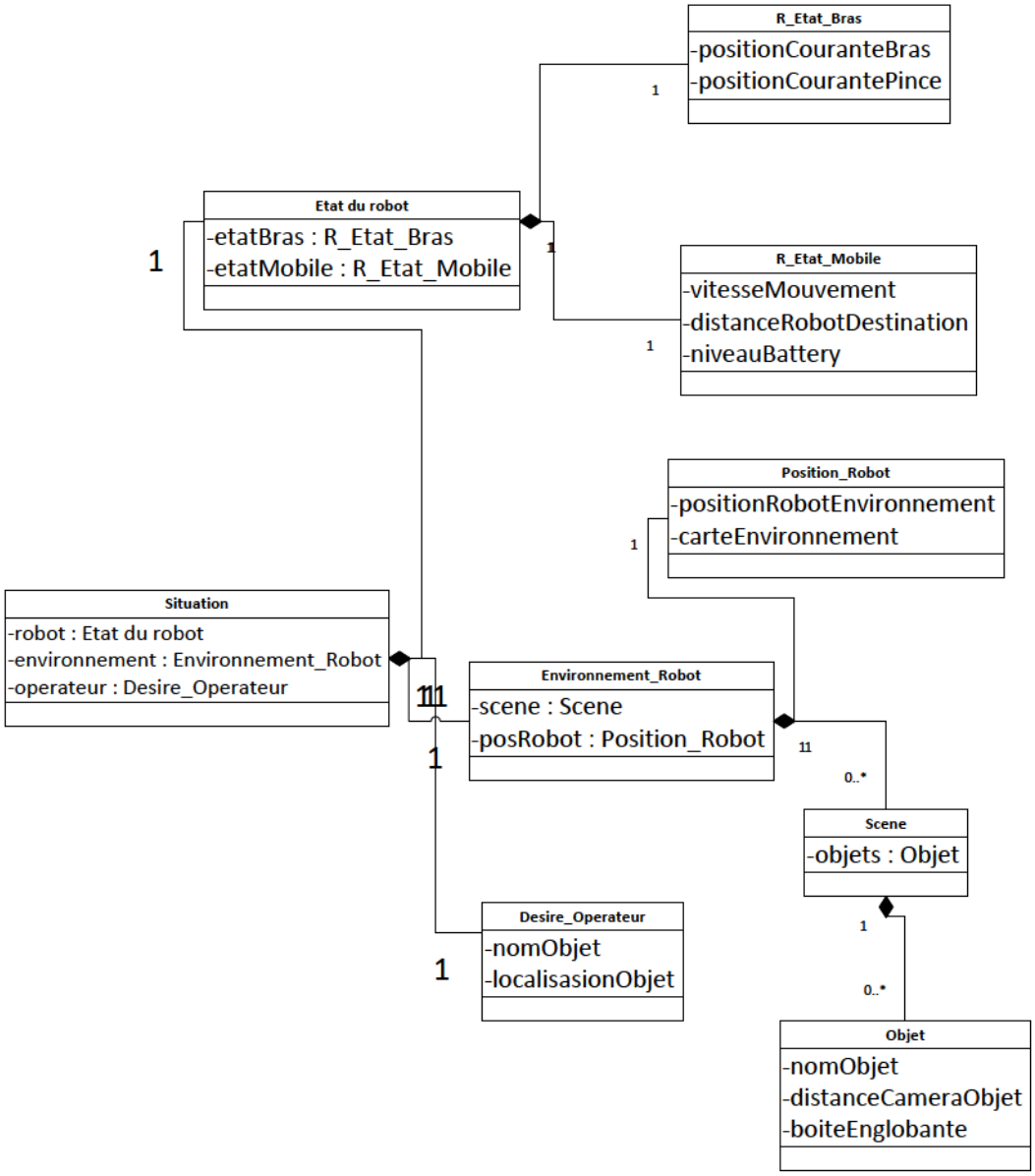


FIGURE 5.1 – Diagramme de classe représentant une situation

le robot, l'environnement et l'opérateur. Les trois concepts **"Robot"**, **"Environnement"** et **"Operator"** sont liés au concept **"Situation"**.

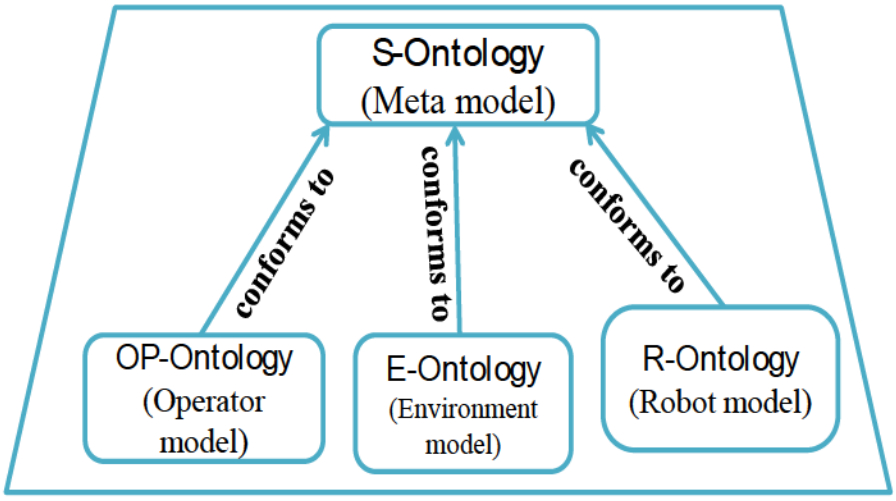


FIGURE 5.2 – Représentation des connaissances dans RSAW

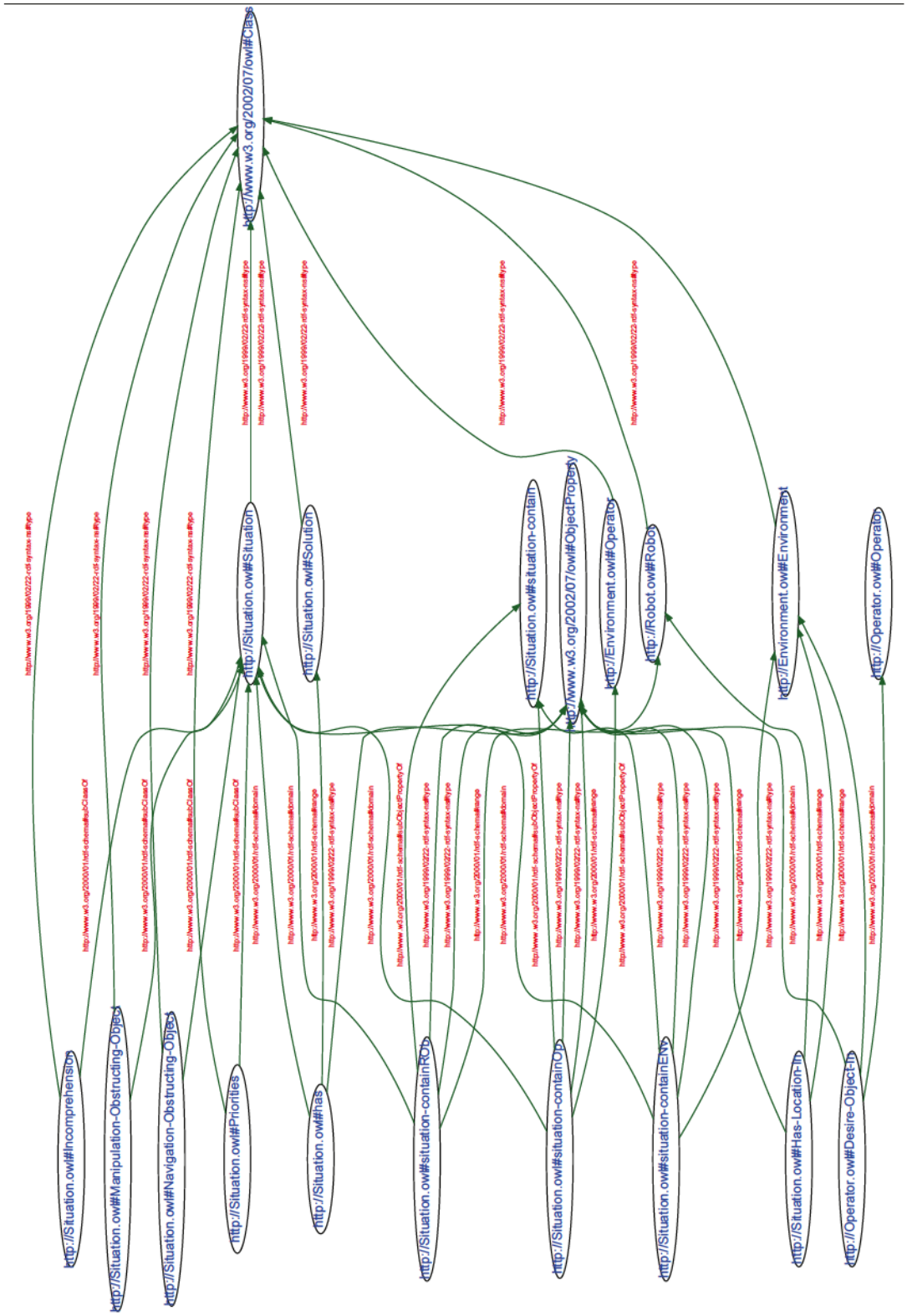


FIGURE 5.3 – Méta modèle de la situation (S-Ontology)

La situation contient l'environnement, ils sont liés par l'ObjectProperites : **situation-contain-ENv**. La situation contient le robot, ils sont liés par l'ObjectProperties : **situation-contain-ROb**. La situation contient l'opérateur, ils sont liés par l'ObjectProperties : **situation-contain-Op**. Les concepts Robot et Environnement sont aussi interreliés grâce à l'ObjectProperties **Has-Location-In**. L'environnement et l'opérateur sont liés par l'ObjectProperties **Desire-Object-in**. Cette interdépendance sert à considérer la situation comme une **unité de connaissances non dissociable**.
Chaque situation a une solution. Donc, il existe dans l'ontologie un concept solution lié au concept situation par l'ObjectProperties **Has**.
Après que nos travaux de recherche avait été menés sur les situations de blocage, nous avons créé dans S-Ontology une typologie selon les types des situations de blocage comme explicité dans la section 4.2.3.2.
Cette ontologie mémorise des instances de situations de blocage.

5.5.2 Niveau modèle

Dans cette section, nous présentons les ontologies du domaine représentant les modèles dans notre système.

5.5.2.1 Ontologie du robot (R-Ontology)

Comme vu dans dans la section 4.2.2.2, un robot est composé d'une partie mobile pour se déplacer, et d'un bras avec un préhenseur pour saisir les objets dans son environnement. La partie réactive de notre architecture retourne des valeurs représentant les états de chaque composant à l'instant où le système détecte une anomalie.
A partir cela, R-Ontology, l'ontologie représentant les connaissances du robot, a été définie.
Pour décrire R-Ontology, nous montrons d'abord les relations entre les principaux concepts, ensuite nous traitons chaque ensemble ayant comme concept père un des concepts principaux de R-Ontology. Ces concepts sont définis selon les composants d'un Robot à savoir Arm, Mobile-Base et Gripper.
Ce qui permet de définir les triplets **HasArm(Robot, Arm), HasMobile(Robot,Mobile-Base) et HasGripper(Robot,Gripper)**.
Chaque Robot a un nom ainsi qu'un type de la classification faite dans la section 2.3.3.1.
La figure 5.4 montre les triplets distingués avec les principaux concepts de R-Ontology.
L'état du bras est représenté par les degrés de liberté du bras.
Le concept **"Arm"** est lié à la position du bras **"ArmPosition"**. Cette position est constitué par les différents dataproperties qui représentent les degrés de liberté du bras, en langue anglaise Degree Of Freedom (hasDOF1, hasDOF2, hasDOF3, hasDOF4, hasDOF5, hasDOF6). Des chaînes de caractères peuvent être associées à la position de bras. La figure 5.5 montre les principaux triplets issu du concept "Arm" dans R-Ontology.

La position du préhenseur (Gripper) est l'angle de chaque doigt par rapport à l'axe central sortant du préhenseur. Le concept "Gripper" est lié aux dataproperties qui représentent les angles de chaque doigt par rapport à l'axe central de la pince Posfinger1, Posfinger2, Posfinger3, Posfinger4, Posfinger5. Des chaînes de caractères peuvent être associées à la position du préhenseur (**Close, Open**).
La figure 5.6 représente les principaux triplets issus du concept "Gripper" dans R-Ontology.
L'état de la partie mobile est représenté par la distance de la base mobile à sa destination hasDistance, la niveau de batterie hasBattery et la vitesse de navigation hasVelocity. Des chaînes de caractères peuvent être associées concernant le niveau de la batterie et/ou si le mobile est en mouvement ou arrivé à sa destination.
La figure représente les principaux triplets issus du concept "Mobile-Base" dans R-Ontology.

5.5.2.2 Ontologie de l'environnement (E-Ontology)

Comme décrit dans la section 4.2.2.2, l'environnement du robot est composé de la localisation du robot et des objets pouvant être rencontrés lors de son évolution dans l'environnement.
La localisation d'un robot peut être par exemple une chambre, une cuisine ou une autre pièce si le robot évolue dans un environnement indoor. Cette localisation est représentée par la position de la partie mobile dans l'environnement. Les objets sont répartis dans l'environnement et ceux que le robot peut distinguer existent déjà dans la base de ses connaissances. Donc, le robot ne peut percevoir que les objets qu'il connaît à l'avance. Les coordonnées d'un objet sont celles du rectangle qui l'englobe, son nom et la distance qui le sépare du préhenseur.
E-Ontology représente les connaissances de l'environnement. L'environnement du robot est composé de *Location* du robot et des objets dans l'environnement. La figure 5.8 montre les principaux concepts de E-Ontology. La *Localisation* du robot est caractérisée par la position de la base mobile dans l'environnement. Cette position est formalisée par le vecteur(hasPositionX, hasPositionY, hasPositionTheta) de la base mobile dans la carte de référence (2D) de l'environnement. La figure 5.9 représente les principaux triplets issus du concepts Location dans E-Ontology. Les *Objets* de la scène sont définis par leurs noms hasName, bounding box : (hasXMin, hasXMax, hasYMin, hasYMax) et leurs distances à la pince hasDistanceToRobot. Le concept représentant les objets de la scène est composé de quatre concepts : des objets manipulables, des objets non-manipulables, la géométrie des objets et leurs emplacements. Les attributs des objets correspondent aux entités defines pour les *Object*. Ces attributs représentent chaque entité des bounding box,de la distance object-Gripper, et chaque composant de la localisation de la base mobile dans la map de l'environnement. La Fig. 5.10 illustre les triplets issus du concept Objet de E-Ontologie.

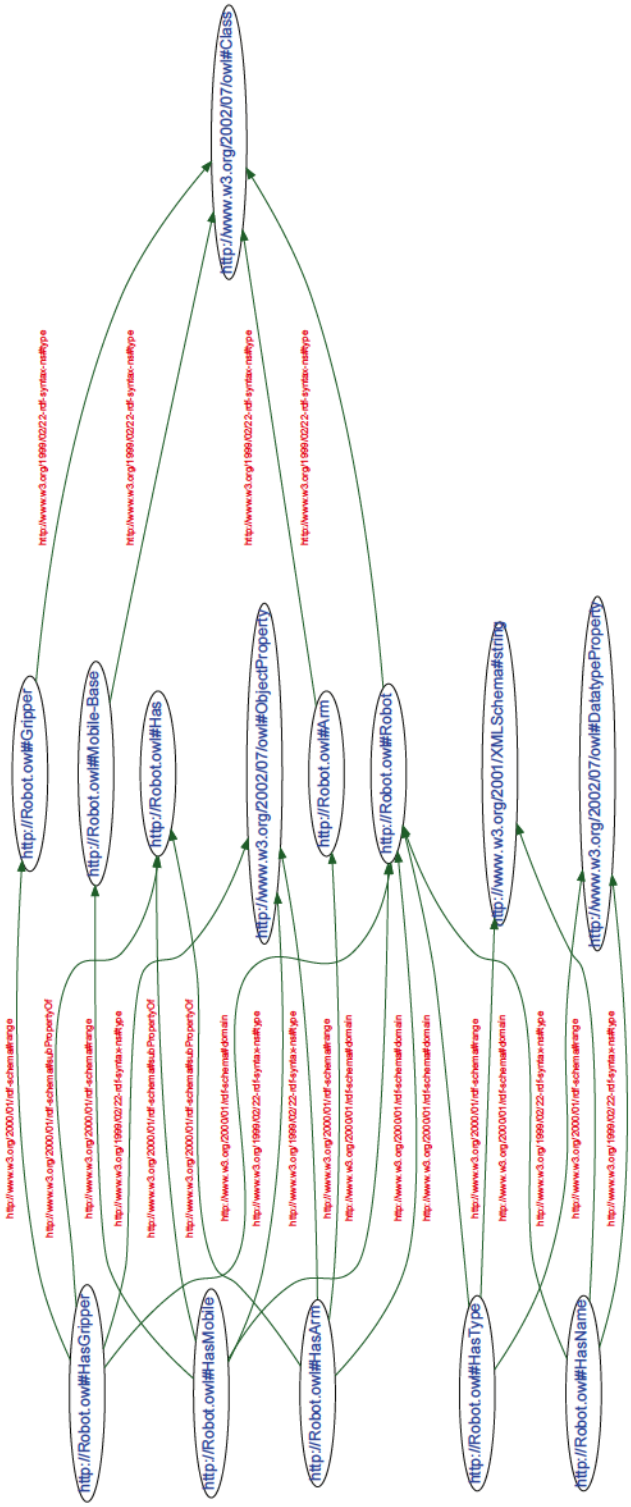


FIGURE 5.4 – Principaux concepts de R-Ontology

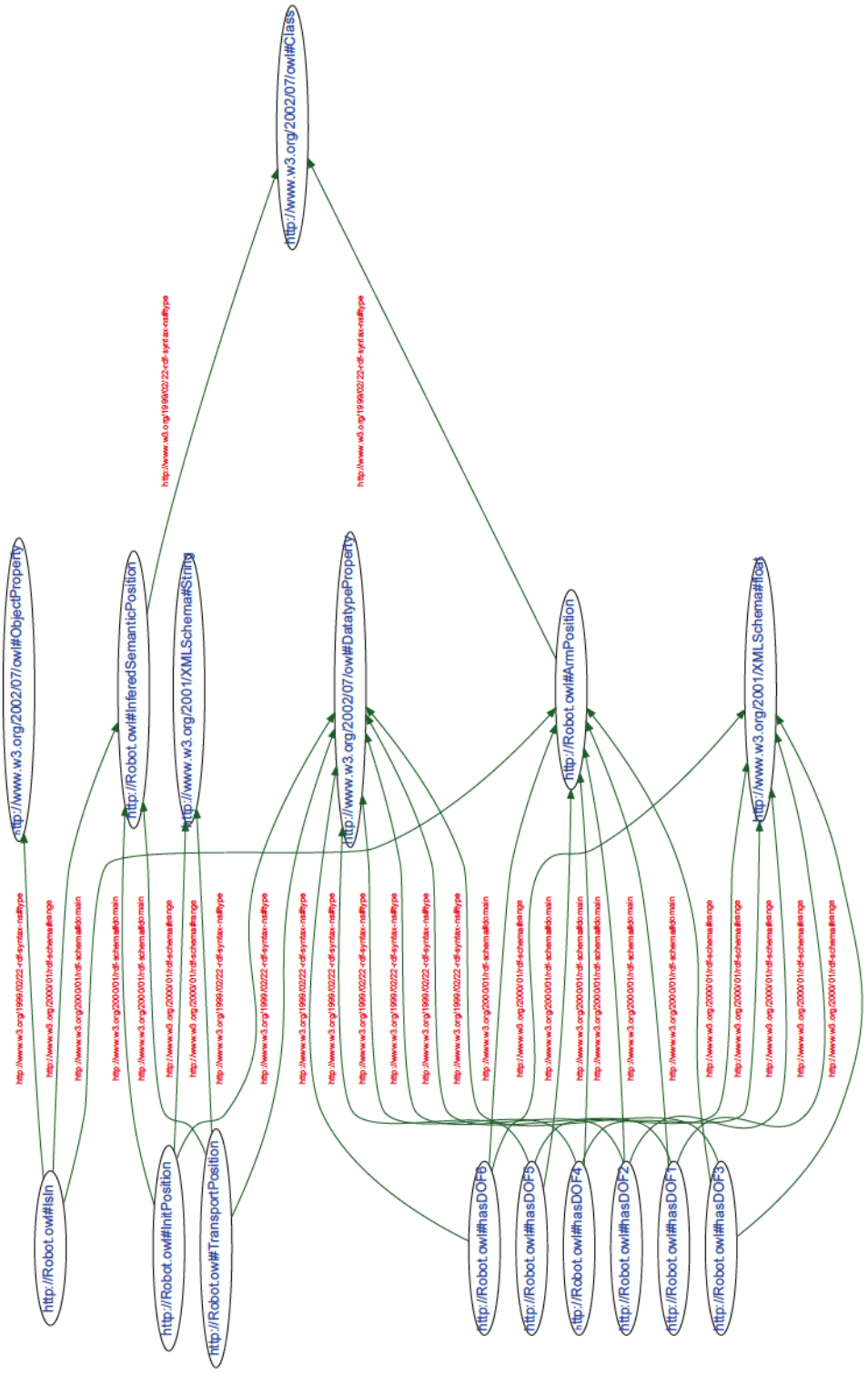


FIGURE 5.5 – Triplets issus du concept Arm dans R-Ontology

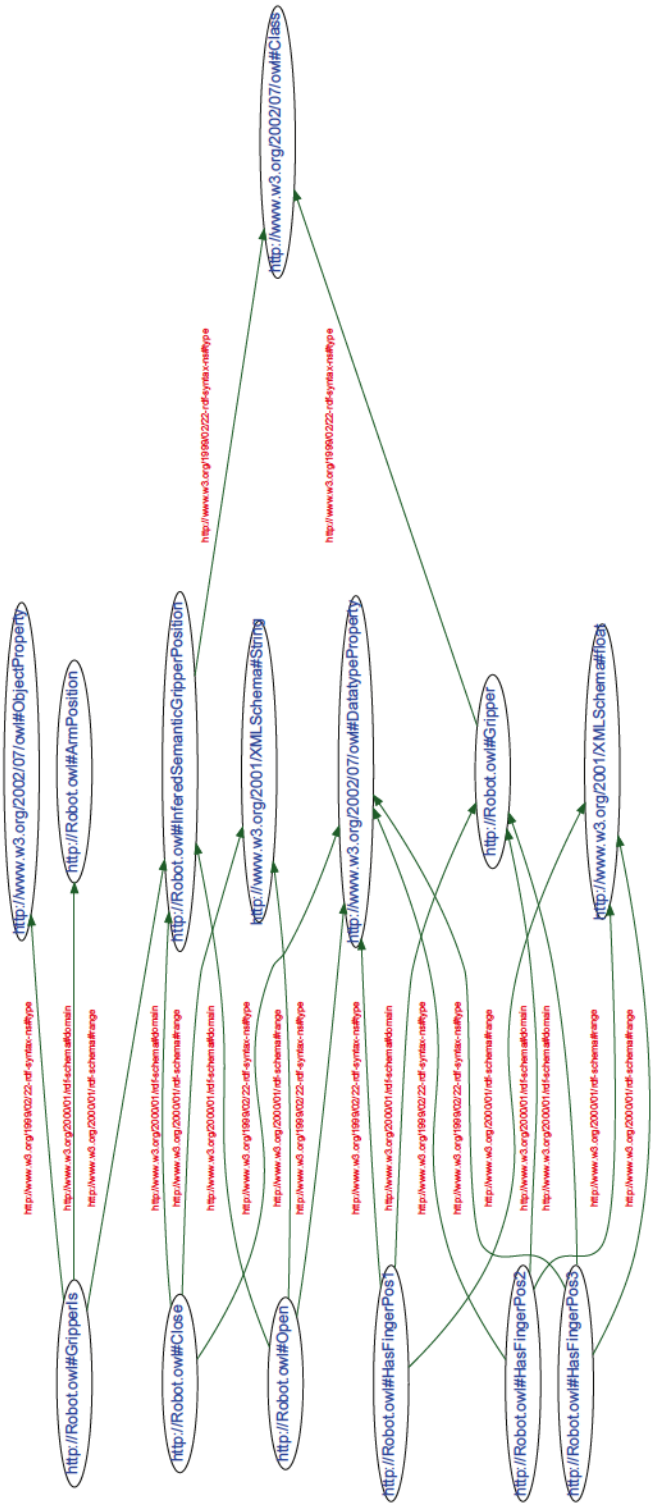


FIGURE 5.6 – Triplets issus du concept Gripper dans R-Ontology

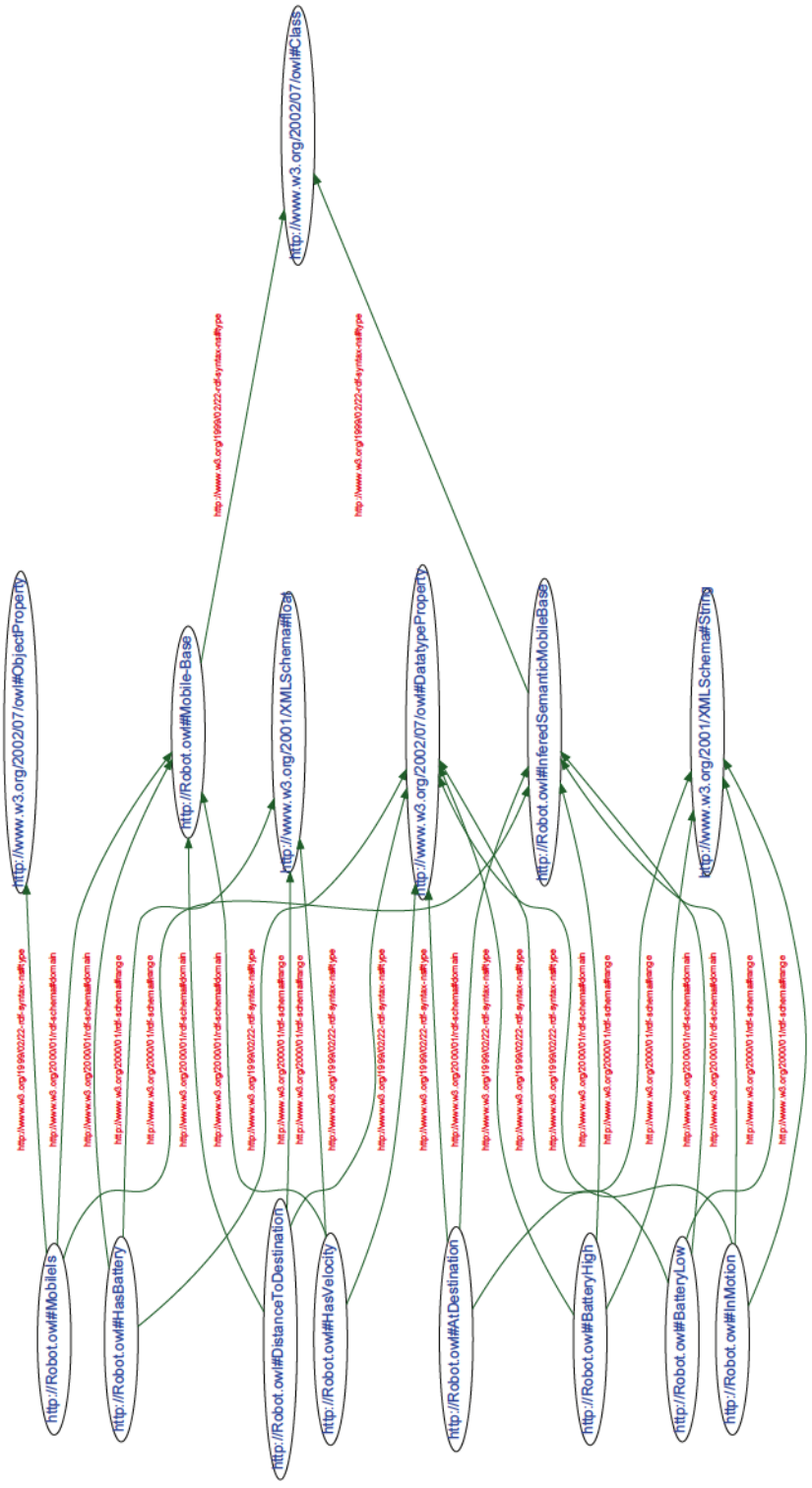


FIGURE 5.7 – Triplets issus du concept Mobile-Base dans R-Ontology

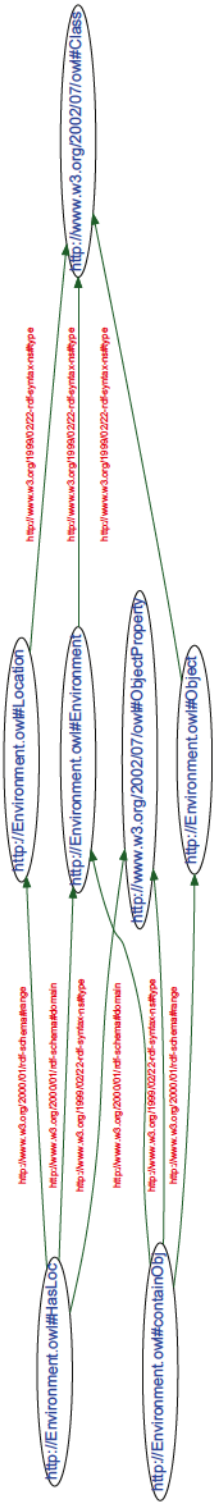


FIGURE 5.8 – Principaux concepts de E-Ontology

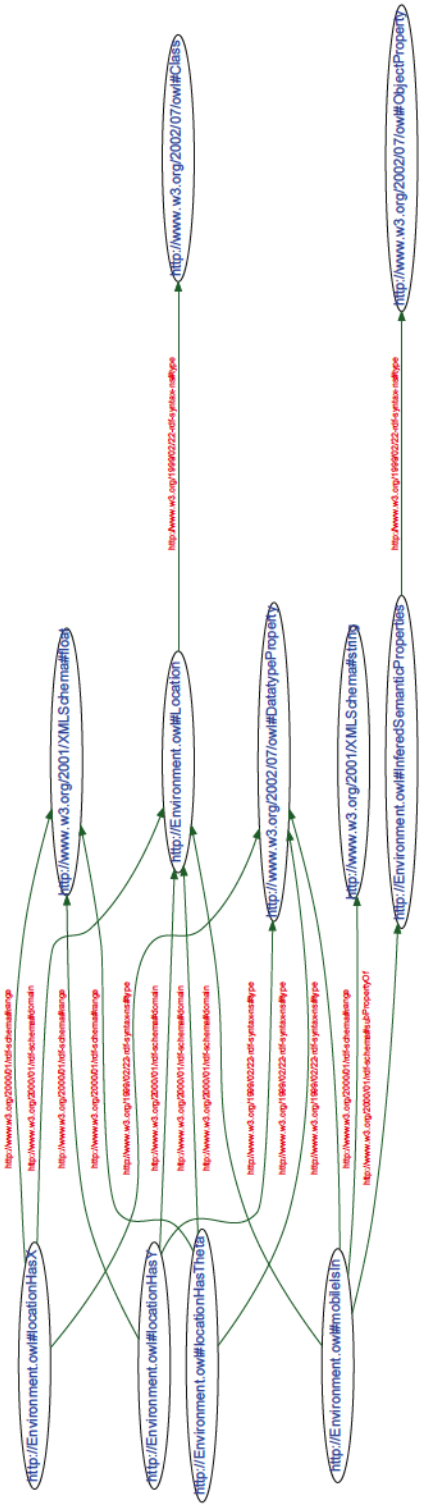


FIGURE 5.9 – Triplets issus du concept Location dans E-Ontology

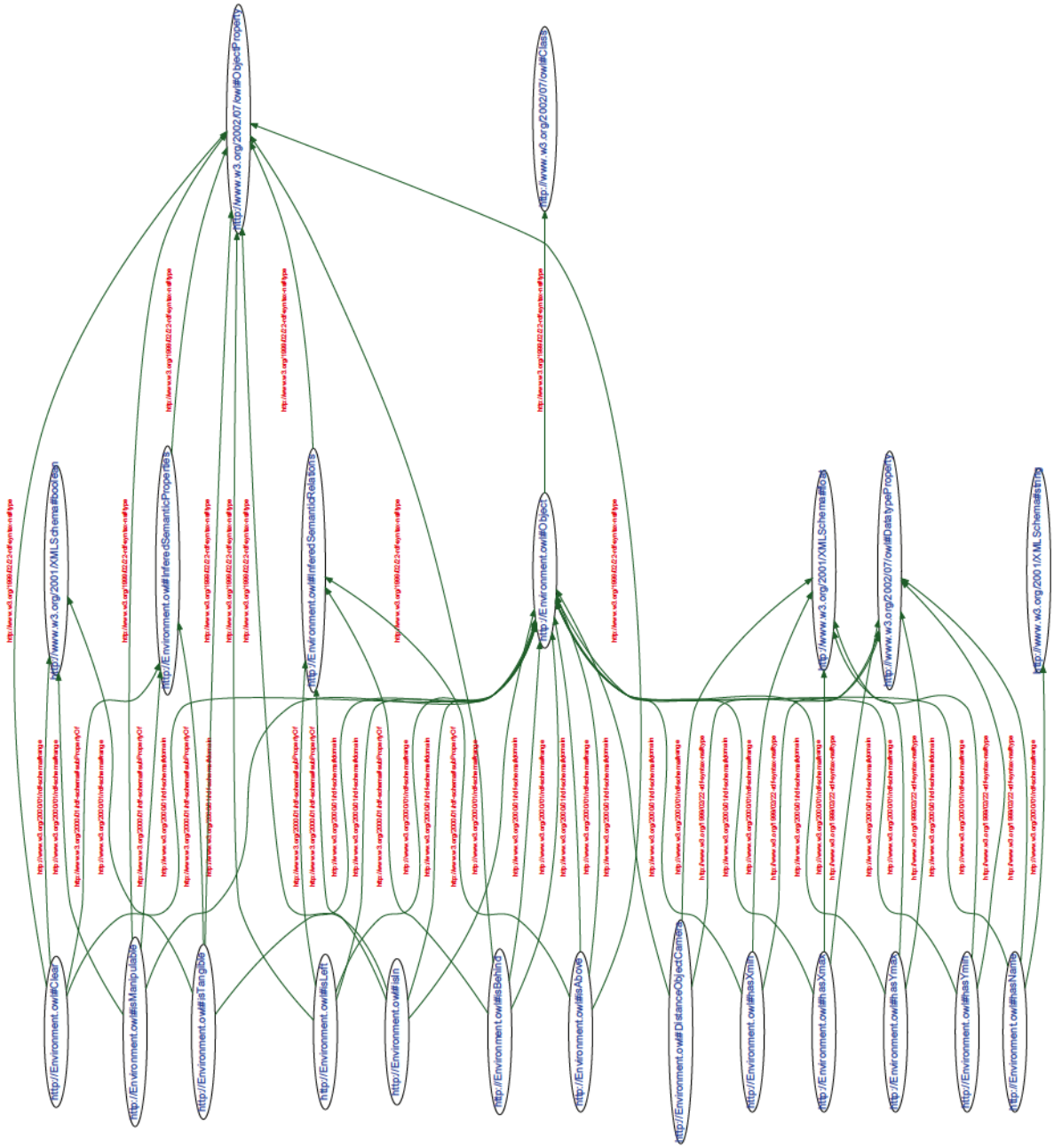


FIGURE 5.10 – Triplets issus du concept Object dans E-Ontology

5.5.2.3 Ontologie de l'opérateur (OP-Ontology)

OP-Ontology est l'ontologie qui représente les connaissances de l'*Operator*. Elle est structurée selon le profil qui est le niveau d'expertise de l'opérateur et le désir de ce dernier. Le désir de l'opérateur est généralement un objet de l'environnement (operatorHasDesir). Il est indiqué dans l'Interface Homme Machine par le ($Name_{desiredObject}$) et le nom de sa localisation dans l'environnement. La Fig. 5.11 représente la structure de l'ontologie de l'opérateur.

5.6 Conclusion

Ce chapitre a fait l'objet de notre deuxième contribution. Cette contribution concerne la représentation des connaissances adoptée dans RSAW pour favoriser la compréhension des situations par un robot. La structure d'ontologie est adoptée pour représenter les situations du fait qu'elles offrent dans un même cadre une puissante représentation des connaissances ainsi que le possibilité de raisonnement. Ces ontologies sont représentées avec le formalisme RDFS. L'organisation des ontologies suit l'Ingénierie dirigée par les modèles (IDM). Ces choix favorisent la généricité de notre approche et la facilité de son intégration dans un système robotique.
Ce chapitre a dressé, tout d'abord, les motivations et choix pour la représentation des connaissances, le formalisme de description et la méthodologie de modélisation adoptés. Ensuite, il a décrit la modélisation de la situation ainsi que l'organisation des connaissances dans RSAW. Enfin, il a présenté la modélisation des connaissances et les ontologies élaborées.
Dans le chapitre suivant, nous mettons l'accent sur l'activité de compréhension. Cette activité est fondée sur un raisonnement.

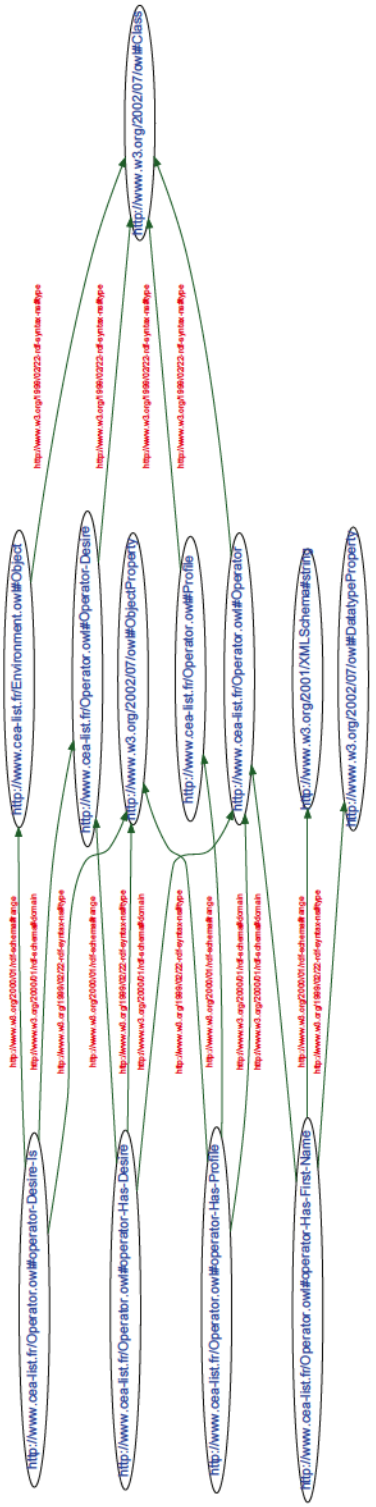


FIGURE 5.11 – Modèle représentant l'opérateur (OP-Ontology)

Chapitre 6

RAISONNEMENT POUR LA COMPRÉHENSION DES SITUATIONS DANS RSAW

Sommaire

6.1 Introduction

L'activité de compréhension décrite dans RSAW s'effectue en deux étapes permettant à un robot d'interpréter les données de la situation de blocage, d'une part, et d'autre part de préparer la prise de décision pour la surpasser en vue de répondre à son opérateur. Dans ces deux étapes le passage par un raisonnement s'impose puisque l'objectif est de les automatiser afin d'augmenter l'autonomie du robot. Différents mécanismes de raisonnement sont connus en Intelligence Artificielle (IA) applicables

dans différentes situations. Le choix d'un type de raisonnement répond, en général, à un besoin de fiabilité, de rapidité et de coût.
Dans ce chapitre, après une motivation des choix de raisonnement adoptés pour réaliser l'activité de compréhension, nous décrivons le raisonnement établi pour chacune des étapes de compréhension à savoir, l'interprétation et la préparation de la prise de décision.

6.2 Motivations et Choix

Dans la littérature, le raisonnement en IA est défini comme un processus cognitif permettant l'obtention de nouveaux résultats ou la vérification de la réalité d'un fait en appliquant différentes règles dans différents domaines d'application. Les mécanismes de raisonnement reconnus sont la déduction, l'abduction, l'induction et l'analogie[59].

- Un **raisonnement déductif** permet d'établir la déduction des connaissances à partir de connaissances déjà acquises. Ce processus est fiable du moment que les techniques de base sont empruntées à l'être humain qui utilise l'abstraction, la recombinaison et la connaissance
- Un **raisonnement inductif** permet d'aboutir à une généralisation à partir des observations effectuées
- Un **raisonnement abductif** suppose une réversibilité dans le sens où il considère que les résultats sont toujours impliqués aux causes et vice-versa.
- Un **raisonnement par analogie** est fondé sur un processus de recherche d'une situation passée semblable (ou voisine) à une nouvelle situation.

Certes ces mécanismes de raisonnement peuvent être appliqués à chaque situation rencontrée, néanmoins ceci se fera au dépend soit de la fiabilité soit du coût ou de la rapidité comme indiqué dans le Tableau 6.1 tiré de l'étude menée à l'Université de Lyon par Barga [59] en psychologie cognitive.
D'après cette étude, il apparait que l'être humain favorise plutôt la rapidité avec un

	Rapidité	Coût	Fiabilité
Analogie	++	-	+-
Abductif	-	+	+
Inductif	+	++	+
Déductif	-	+++	++

TABLE 6.1 – Caractéristiques des mécanismes de raisonnement [59]

effort réduit dans le raisonnement. Ceci implique que le type de raisonnement qu'il adopte se base le plus souvent sur le raisonnement par analogie qui s'avère le plus rapide mais aussi le moins coûteux, et le moins fiable puisqu'il compare des situations souvent formulées avec des symboles différents. De même l'étude stipule que le raisonnement déductif reste celui le plus utilisé dans une activité d'apprentissage du fait de sa fiabilité.

Dans l'application d'un raisonnement au niveau de la phase de compréhension de RSAW, nous cherchons à privilégier la *fiabilité et la rapidité tout en essayant de réduire le coût de calcul*. En effet, d'une part, la fiabilité du raisonnement est importante dans le sens où le robot autonome doit répondre le mieux possible aux attentes de l'opérateur. Ce qui signifie que la situation de blocage reconnue par le robot doit correspondre parfaitement à la situation réelle. D'autre part, la rapidité dans le raisonnement permet de répondre au désir de l'opérateur dans un temps raisonnable. Ce qui implique qu'il est souhaitable que le robot puisse aussi profiter de son expérience passée (dans le sens où la situation de blocage courante est similaire à une situation déjà rencontrée) afin de proposer à l'opérateur une solution « déjà vue » pour surpasser la situation de blocage courante. L'étude présentée plus haut qui a montré que la fiabilité est mieux perçue avec un raisonnement déductif alors que le raisonnement par analogie donne un résultat plus rapidement, nous a conduites à adopter un raisonnement dual, combinant un raisonnement déductif pour donner une description de la situation de blocage et un raisonnement par analogie pour trouver un plan d'action permettant de résoudre le problème. Les connaissances déduites en appliquant le raisonnement déductif sont exprimées dans le langage symbolique commun décrit dans le chapitre 5 pour être exploitées dans le raisonnement par analogie. Ceci contribuera à améliorer la fiabilité et la rapidité de la compréhension des situations de blocage par le système robotique.
Ce choix n'est pas contraignant puisque :

- D'un côté, avec les générateurs de plans d'action classiques en robotique utilisés dans la résolution des situations de blocage, il est possible de ne pas aboutir à un plan d'action pour résoudre une situation de blocage. Cela est dû soit à un manque de connaissances, soit à une incohérence des états décrivant le problème de planification.
  Le raisonnement déductif proposé dans RSAW est complémentaire et donne une description plus exhaustive puisqu'elle est dotée de la sémantique liant les différents concepts de la situation. En effet, ce raisonnement est utilisé en tant que transformation de modèles (l'ontologie de l'environnement, l'ontologie de l'opérateur et l'ontologie du robot) pour construire l'ontologie de la situation de blocage (instance du méta modèle). Cette instance de situation de blocage permet la description du problème de planification pour un générateur de plan d'action (tel que CPT).

- D'un autre côté, nous pensons qu'il serait judicieux de profiter des expériences passées du robot, puis qu'il est possible que les situations de blocage ressemblent à des situations déjà rencontrées. Cette réflexion nous a conduits à penser à l'utilisation d'un raisonnement par analogie pour optimiser l'élaboration des solutions aux situations de blocage.

Ainsi notre proposition dans RSAW utilise ces deux types de raisonnement dans sa phase de compréhension qui se déroule, comme présenté dans le chapitre 4, en deux étapes : (1) l'interprétation de la situation de blocage et (2) l'étape de préparation de la prise de décision pour l'élaboration d'une solution. Le principe de l'utilisation de ces raisonnements est le suivant :

1. Le raisonnement déductif (saturation de la base de connaissance en chainage avant) est utilisé dans l'étape de compréhension (1) pour l'interprétation des situations de blocage du robot. En effet, même si ce mécanisme réduit la rapidité, il s'adapte bien pour garantir la fiabilité de la constitution de la situation. Durant l'étude expérimentale de RSAW, nous avons expérimenté la faisabilité de la résolution des situations de blocage avec ce mécanisme de raisonnement. Ceci a été fait en développant un connecteur entre l'ontologie représentant une situation de blocage (instance du méta modèle) et l'entrée du générateur du plan d'action. Les résultats obtenus étaient insuffisants. Ceci est compréhensible car l'aspect sémantique créé par l'interprétation dans les ontologies est perdu, et donc pas exploité, lors du passage au générateur du plan d'action (manque de connaissance, etc..). L'amélioration que nous proposons pour y remédier consiste à appliquer un deuxième raisonnement sur les ontologies. Comme motivé plus haut, il s'agit du raisonnement par analogie.
2. Le raisonnement par analogie est utilisé par la suite dans la deuxième étape de compréhension (2) pour exploiter l'expérience acquise par le robot. Ce raisonnement peut donner un résultat approximatif en fonction du calcul de similarité adopté. Ceci a pour effet de réduire l'intervention de l'opérateur. En effet, ce dernier n'a plus qu'à valider le résultat obtenu (cf. chapitre 4). Ce type de raisonnement n'est pas couteux contrairement au précédent mais il est le moins fiable.

Pour résumer, notre proposition concernant le raisonnement dans la phase de compréhension consiste en l'adoption d'un langage symbolique pour la description des situations, l'application d'un raisonnement déductif pour interpréter les données de la situation de blocage et un raisonnement par analogie pour la préparation de la prise de décision. Dans ce qui suit nous présentons l'étape d'interprétation dans laquelle nous présentons le raisonnement déductif ainsi que la construction de la situation de blocage dans RSAW. Ensuite, nous présentons l'étape de préparation pour la prise de décision

dans laquelle nous posons le principe du raisonnement par analogie, le matching d'ontologies ainsi que l'application du raisonnement par analogie telle qu'élaborée dans RSAW.

6.3 Interprétation des données de la situation de blocage

Dans cette étape de la compréhension, l'interprétation des données joue un rôle important car elle traduit les valeurs perçues par le robot, au moyen de ses capteurs (proprioceptifs, extéroceptifs) et de l'IHM, lors de la phase de reconnaissance en vue d'obtenir une description homogène utilisant un langage symbolique commun établi dans le cadre de ces travaux de thèse. En fait, l'étape de reconnaissance alimente les ontologies correspondant au domaine du robot (R-Ontology), de l'environnement (E-Ontology) et de l'opérateur (OP-Ontology) telles que décrites dans le chapitre précédent. Durant cette phase, il s'agit de raisonner sur ces connaissances pour établir le modèle de la situation de blocage, en utilisant le langage symbolique commun. Comme précisé dans la section précédente, le mécanisme adopté dans cette étape est déductif. Nous nous appuyons sur les éléments collectés sur l'état du monde au moment du blocage et sur les connaissances déduites à l'aide des règles d'inférences contenues dans l'ontologie pour obtenir une description la plus complète possible du blocage en saturant la base de connaissance par chainage avant. Le raisonnement déductif correspond aux transformations à appliquer sur nos modèles au niveau architectural (cf. chapitre précédent), telles que décrites en Ingénierie Dirigée par les Modèles. Ces transformations sont basées sur des règles d'inférence établies dans le cadre de notre contexte.
Dans cette section, nous abordons l'application du raisonnement déductif afin d'assurer l'interprétation puis la constitution du modèle de la situation de blocage courante en se basant sur le langage symbolique commun ainsi décrit dans 5.2.

6.3.1 Raisonnement déductif

La déduction est un mécanisme de raisonnement qui s'applique du général au particulier. Ce mécanisme offre une méthode par laquelle les propositions sont élaborées en référence à des propositions déjà établies présentes dans l'ontologie. Il est défini par sa seule forme appelée règle de déduction [182] :

Si *propositions vraies* alors *conclusion vraie.*

Chacune des règles de déduction permet d'obtenir de nouvelles connaissances appelées (faits). La conclusion d'une règle n'est valide que si la proposition de départ est exacte. La déduction permet la conservation de la cohérence d'une théorie. Donc, si la théorie initiale (propositions) est cohérente, alors toute théorie qui en est une conséquence déductive (conclusion) reste cohérente [63]. C'est le propre du syllogisme.
Dans la littérature des systèmes logiques, les règles d'inférences sont les règles sur

lesquelles se fond le mécanisme de raisonnement déductif. Ce mode de raisonnement a fait ses preuves dans plusieurs domaines et notamment dans le web sémantique par la mise en œuvre des moteurs d'inférences. Un moteur d'inférence est un outil mettant en œuvre le raisonnement déductif à partir de bases de faits et de bases de règles. On peut citer les moteurs d'inférences les plus utilisés tels que JENA, Pellet, OWLIM, KAON, Hermit, FACT++ (Fast Classification of Terminologies), Racer (Renamed ABox and Concept Expression Reasoner) [60]. L'inférence dans le Web sémantique est l'un des outils de choix pour améliorer la qualité de l'intégration de données sur le Web, en découvrant de nouvelles relations, en analysant automatiquement le contenu des données, et/ou la gestion des connaissances sur le Web.
Les ontologies qui se concentrent sur les méthodes de classification et la façon dont les ressources peuvent être associées aux concepts profitent des règles d'inférences pour se concentrer sur la définition d'un mécanisme général sur la découverte et la création de nouvelles relations fondées sur celles qui existent déjà. Les techniques basées sur l'inférence sont également importantes dans la découverte d'éventuelles incohérences dans les données.
Nous utilisons, dans nos travaux, le moteur d'inférence offert par JENA pour effectuer l'interprétation des données perçues. Dans cette étape, un ensemble de règles énoncées dans l'annexe A.2 a été élaboré en vue de construire les situations de blocage courantes. Nous présentons dans ce qui suit la construction de la situation de blocage.

6.3.2 Construction de la situation de blocage courante

La construction de la situation courante consiste à interpréter les données perçues lors de la phase de reconnaissance et mémorisées dans les ontologies correspondantes :

- les données de l'état des composantes du robot sont mémorisées dans R-Ontology ;
- les données de l'état de l'environnement sont mémorisées dans E-Ontology
- les données de l'état de l'opérateur sont mémorisées dans OP-Ontology

L'interprétation repose sur l'application des règles d'inférences présentées plus haut sur les données perçues en vue de constituer la situation de blocage courante. Pour ce faire et conformément à notre choix d'utiliser le formalisme RDFS pour représenter les ontologies citées ci-dessus (cf. section 5.3.2), nous construisons, par application des règles, les données déduites écrites dans le langage symbolique commu relatives aux situations. Lorsque ces nouvelles données sont créées et à l'aide de requête DESCRIBE, les graphes RDFS que nous appelons graphes de compréhension sont construits. En fait, nous définissons deux graphes de compréhension. Le premier est le graphe R-RDFS qui contient les données (déduites) sémantiques associées au robot. Le second est le graphe E&OP-RDFS contenant les données (déduites) sémantiques associées à l'objet désiré par l'opérateur et sa position dans l'environnement. Ces deux graphes coopèrent pour caractériser la sémantique de la situation du robot.
Les données sémantiques déduites sont une combinaison de mots (par exemple, le

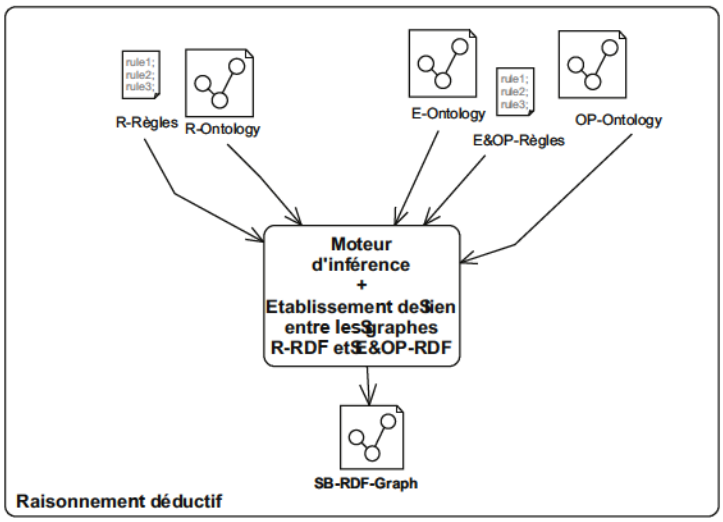


FIGURE 6.1 – Application des règles d'inférences

mot "Init_Position" indique que le bras est prêt à manipuler). Cette connaissance est utilisée pour créer (i) le graphe R-RDFS du robot et (ii) le graphe (E&OP-RDFS) représentant l'objet désiré par l'opérateur et sa localisation dans l'environnement. Ces règles d'inférences, (R-Règles) permettant la création des graphes R-RDFS et (E&OP-Règles) permettant la création de E&OP-RDFS, sont explicitées dans l'annexe A.2. Les règles d'inférences sont appliquées sur les instances des ontologies correspondantes comme suit :

- R-Règles A.2.1 sont appliquées sur les instances de R-Ontology, pour la création du graphe R-RDFS (Graphe de compréhension de robot) afin de représenter la sémantique de l'état du robot conformément au langage utilisé.
- E&OP-RèglesA.2.2 sont appliquées sur les instances de E-Ontology et OP-Ontology, pour créer le graphe E&OP-RDFS (Graphe de Compréhension de l'environnement et du désir de l'opérateur) afin de représenter la sémantique des objets, de l'objet désiré, de la relation entre eux et de la localisation du robot dans l'environnement.

Des exemples de règles concernant le bras, la main et l'environnement sont donnés dans le Tableau 6.2. Le graphe RDFS de la situation de blocage actuelle (SB-RDFS-Graph) est alors construit conformément au métamodèle S-Ontology 5.5.1. La création de ces graphes est obtenue par des requêtes SPARQL. SB-RDFS-Graph est créé en établissant le lien entre les deux graphes de compréhension R-RDFS et E&OP-RDFS sur la base de **la localisation actuelle** du robot comme le montre la Fig. 6.1. La Fig. 6.2 illustre un exemple de SB-RDFS-Graph. Dans cet exemple, un lien sémantique entre le nœud représentant la base mobile (Mobile-Base) et le nœud représentant la

Prémisses (propositions)	Déductions (conclusions)
equal(?initPositionDOF1value, ?jacoPositionDOF1value) equal(?initPositionDOF2value, ?jacoPositionDOF2value) equal(?initPositionDOF3value, ?jacoPositionDOF3value) equal(?initPositionDOF4value, ?jacoPositionDOF4value) equal(?initPositionDOF5value, ?jacoPositionDOF5value) equal(?initPositionDOF6value, ?jacoPositionDOF6value)	(jaco arm-Is init-position)
equal(?HasFingerPos1, 50) equal(?HasFingerPos2, 50) equal(?HasFingerPos3, 50) equal(?HasFingerPos4, 50) equal(?HasFingerPos5, 50)	(gripperofjaco GripperIs open)
lessThan(?batteryValue, 50.0)	(robosoft HasBattery low)
notEqual(?o1, ?o2) equal(?DistanceCamerao1 ?DistanceCamerao2) lessThan(?yMaxo1, ? yMino2)	(?o1 isLeft ?o2) (?o1 isClear "true") (?o2 isRight ?o1) (?o2 isClear "true")

TABLE 6.2 – Quelques exemples de règles d'inférences

localisation actuelle est établi (Kitchen). Aussi il existe un lien entre Operator-Desire et l'objet désiré (can).
La situation ainsi créée est sous forme de graphe RDFS.
Dans ce qui suit, nous présentons la deuxième étape de la phase de compréhension qui est la préparation de la prise de décision. Cette étape consiste en un raisonnement par analogie pour connaitre la situation la plus proche à la situation de blocage courante.

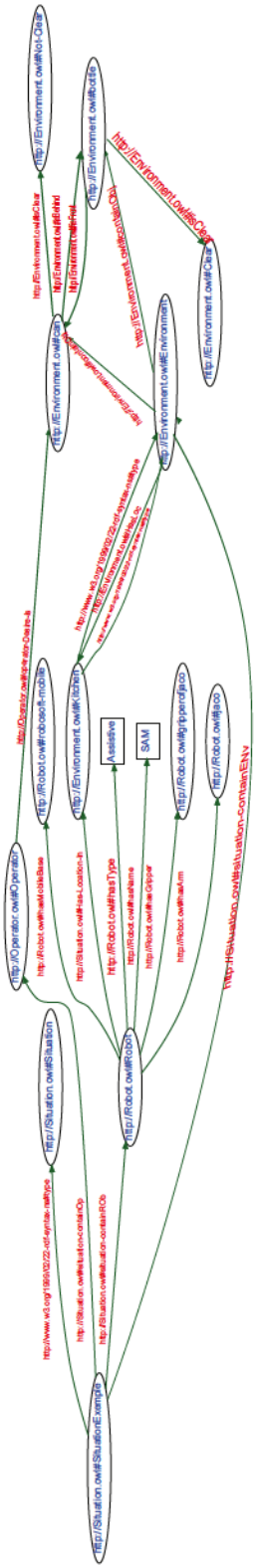


FIGURE 6.2 – Graphe RDFS de la situation (SB-RDFS-Graph)

6.4 Préparation de la prise de décision

Nous avons fait le choix d'appliquer un raisonnement par analogie pour effectuer la préparation de la prise de décision (voir section 6.2), notre base de cas (contenant les situations de blocage déjà apprises) étant l'ontologie des situations S-Ontology.
En fait pour une situation de blocage (SB) représentée elle-même par un graphe RDFS, il s'agit de chercher dans S-Ontology une situation (une instance contenue) la plus proche de SB. Ainsi notre problème revient à appliquer un appariement d'ontologies fondé sur un calcul de similarité à choisir.
Dans cette section, nous présentons, tout d'abord, le principe de base du mécanisme de raisonnement par analogie. Ensuite, nous abordons l'appariement d'ontologies et le calcul de similarité. Enfin, nous posons notre proposition telle que établie dans RSAW.

6.4.1 Principe de raisonnement par analogie

Les méthodes de raisonnement en intelligence artificielle (IA) et notamment celles utilisées en robotique, ont fait l'objet de la section 3.4 du chapitre de l'état de l'art 3 de cette thèse. Cependant, pour répondre à cet objectif, les méthodes de raisonnement existantes dans la littérature sont plutôt de type raisonnement à partir d'exemples, à partir de cas, à partir d'instances ou celui basé sur la mémoire. Ces méthodes ne sont autres que des variétés de l'approche très connue et sollicitée en IA, à savoir le raisonnement par analogie [143][252][177][110][225].
Le mécanisme de raisonnement par analogie qui répond à la problématique posée par cette étape est un processus général par lequel un système remarque une similitude entre deux choses [1].

Le principe de ce raisonnement est décrit ainsi : Pour résoudre un problème donné (problème cible), si on retrouve, dans une base de cas, un problème qui lui est **fortement similaire** (problème source) et ayant une solution, cette solution source sera fortement similaire à la solution attendue du problème cible.
Le raisonnement recherché par RSAW pour intégrer de l'intelligence dans un système robotique vise à résoudre les situations de blocage avec le même principe. En effet, le problème source correspond dans RSAW, à la situation de blocage courante (rencontrée par le robot) et le problème similaire correspond à une situation similaire mémorisée dans l'ontologie des situations de RSAW. Ceci est schématisé, dans la Fig. 6.3, qui représente le carré d'analogie tel qu'il est appelé dans la littérature.
Ce carré montre des relations de dépendance (β) entre les problèmes (source respectivement cible) et leurs solutions (source respectivement cible) ainsi que l'existence d'une similarité (α) entre les problèmes (source et cible) et entre les solutions correspondantes (source et cible) calculées au moyen d'une mesure de similarité.

1. http ://fr.wikipedia.org/wiki/Analogie#Informatique

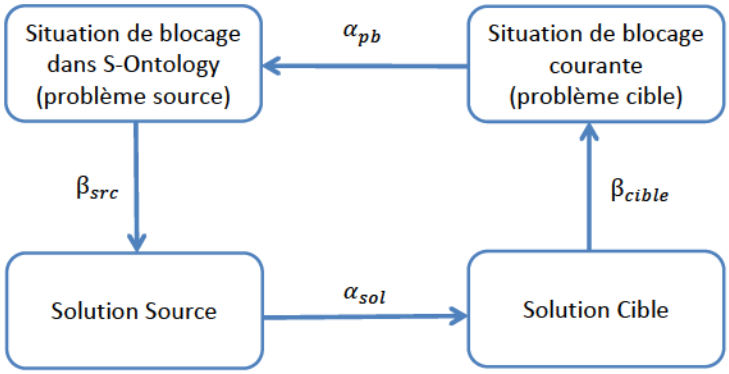


FIGURE 6.3 – Carré d'analogie

La représentation de connaissances conçue dans ce travail de thèse étant basée sur les ontologies (voir chapitre 5), notre approche de raisonnement sera donc appliquée sur une base de cas de type ontologie.

6.4.1.1 Méthode de recherche d'une situation semblable

Comme vu précédemment dans RSAW, chaque situation de blocage est représentée par un graphe RDFS et est mémorisée dans l'ontologie des situations (S-Ontology) qui sera la base de cas pour le raisonnement. Ainsi, cette base de cas contient des instances représentant les situations du passé (cf. Fig. 6.2). On se retrouve donc dans un contexte où la base de cas est une ontologie contenant des graphes RDFS représentant les situations de blocage connues (expérience du robot).
Pour faire la recherche de la situation la plus proche, dans S-Ontology, par rapport à la situation de blocage courante, nous avons commencé par effectuer une étude bibliographique sur la mise en correspondance d'ontologies. Dans la littérature, la méthode de mise en correspondance d'ontologies s'appelle appariement ou matching d'ontologies [70].
Nous adoptons cette méthode dans notre travail afin de faire le raisonnement par analogie et trouver la situation la plus similaire à la situation de blocage courante.

6.4.1.2 Méthode de calcul de similarité

D'après le principe de raisonnement adopté, la recherche de la situation la plus similaire se fait sur la base d'un calcul de similarité. Dans la littérature, différentes techniques de calcul existent et permettent de faire un calcul sur la base terminologique, topologique (structurelle) ou encore sémantique.

Ceci dit, le calcul de similarité basé sur la terminologie induit à une perte de la topologie (ou la structure) et de la sémantique du graphe. Ce qui veut dire, qu'avec ce type de similarité nous ne pouvons pas nous assurer de la comparaison des relations entre les différentes entités des situations. Aussi, nous perdons de l'information contenue dans les graphes, vu que la similarité terminologique ne s'assure que de la syntaxe des mots. Dans notre cas, nous souhaitons profiter de l'information, au sens de la théorie d'information, introduite par les graphes de situations. De ce fait, nous nous imposons l'utilisation d'une mesure de similarité qui garantit les deux hypothèses suivante :

- Hypothèse 1 : Maintenir les relations entre les concepts des situations de blocage courantes (cf. chapitre 5)
- Hypothèse 2 : Utiliser la quantité d'information, énoncée dans la théorie d'information, apportée par les graphes représentant les situations de blocage courantes,

Dans ce qui suit, nous établissons nos choix pour satisfaire nos hypothèses et commençons par l'exploration des systèmes d'appariement d'ontologies existants.

6.4.2 Appariement d'ontologies

L'appariement des ontologies est un processus de détection des correspondances entre des éléments dans deux vocabulaires hiérarchisés [14]. Ces correspondances reposent sur l'existence de propriétés similaires : des relations d'équivalence, de subsomption entre entités, etc. Classiquement, les entités à comparer sont des classes, des propriétés et des individus. En effet, l'appariement des ontologies représente en soi un domaine de recherche en pleine expansion et qui ne cesse d'évoluer [189][231]. Dans la littérature, les travaux existants dans ce domaine sont classifiés conformément à cinq catégories à savoir :

1. les techniques d'appariement s'intéressant aux mesures de similarités ainsi qu'aux stratégies et méthodologies d'appariement,
2. les systèmes d'appariement introduisant de nouveaux algorithmes pour faire l'appariement,
3. les applications pratiques décrivant les solutions d'appariement appliquées à un problème dans un domaine réel,
4. les Framework s'occupant des opérations d'alignement d'ontologies comme la fusion d'ontologies ou la médiation et
5. l'évaluation des systèmes d'appariement [189][231].

Nous remarquons, qu'en général, ces travaux visent un objectif précis et peuvent être combinés pour obtenir des solutions à des problèmes plus complexes [189]. Ils sont fondés sur l'utilisation de technique d'appariement.
Dans notre problématique liée à la compréhension en robotique, nous visons la réalisation

d'un appariement entre le graphe de la situation de blocage courante et celui représentant une situation connue mémorisée dans la base de cas S-Ontology. Cette configuration, où nous sommes amenés à comparer une ontologie à n concepts avec une autre à m concepts utilisant le même langage (n $\geq$ 16 et m $\geq$ 16), nécessite une technique d'appariement permettant de comparer des cardinalités de type n :m. De même pour satisfaire nos hypothèses de travail cités précédemment, cette technique devra permettre de faire l'appariement tout en exploitant le contenu global des graphes représentant les ontologies des situations (concepts, relations, propriétés, etc...).
Dans la suite, nous présentons les choix que nous avons adoptés pour atteindre notre objectif puis notre approche de l'appariement.

6.4.2.1 Travaux existants dans l'appariement d'ontologies

Dans la littérature, se distinguent neuf types de techniques d'appariement d'ontologies comme le montre la figure 6.4 sur lesquelles sont généralement fondés les systèmes d'appariement d'ontologies tels que [81][245][104][105][124].Les systèmes d'appariement d'ontologies sont très nombreux, ils représentent 43,52% des publications dans ce domaine [189](choix établi selon la pertinence et la participation dans la communauté).
Dans les articles [189][231][48], des études de ces systèmes sont données selon des points de vue de maturité, de publications et d'évaluation utilisant les benchmarks de l'OAEI (Ontology Alignemnet Evaluation Initive) [2]. Ces types sont organisés selon le niveau dans lequel les ontologies sont appariées, selon le contenu ou selon le contexte utilisant des ressources externes. Etant donné que les techniques répondant à nos hypothèses sont celles basées sur les graphes, nous nous sommes concentrés sur les systèmes d'appariement supportant une technique basée sur les graphes. Parmi les systèmes d'appariement pertinents, trouvés dans les différentes revues de littératures récentes dans ce domaine, ceux qui utilisent les techniques basées sur les graphes ne sont pas nombreux. Nous considérons, alors, les systèmes suivants :

- Le système FalconAO [122] effectue un appariement d'ontologies combinant deux algorithmes. Le premier est un algorithme linguistique et le deuxième est un algorithme d'appariement de graphes appelé GMO [112].
- Le système OLA (OWL Lite Alignment) [78]effectue l'alignement d'ontologies entre deux ontologies représentées par des graphes. Ce système propose plusieurs caractéristiques pour manipuler le résultat de l'alignement.
- Le système SIGMa [197] effectue de l'alignement d'ontologie en se basant sur la recherche d'information structurelle entre les relations des graphes ainsi qu'entre les propriétés.
- Le système YAM++ [68] utilise une technique de Machine learning pour découvrir les correspondances entre deux ontologies. Ce système utilise une combinaison de techniques d'appariements d'ontologies (niveau structurel, niveau terminolo-

2. http ://oaei.ontologymatching.org/

gique). Au niveau structurel, ce système utilise l'algorithme de Similarity Flooding.

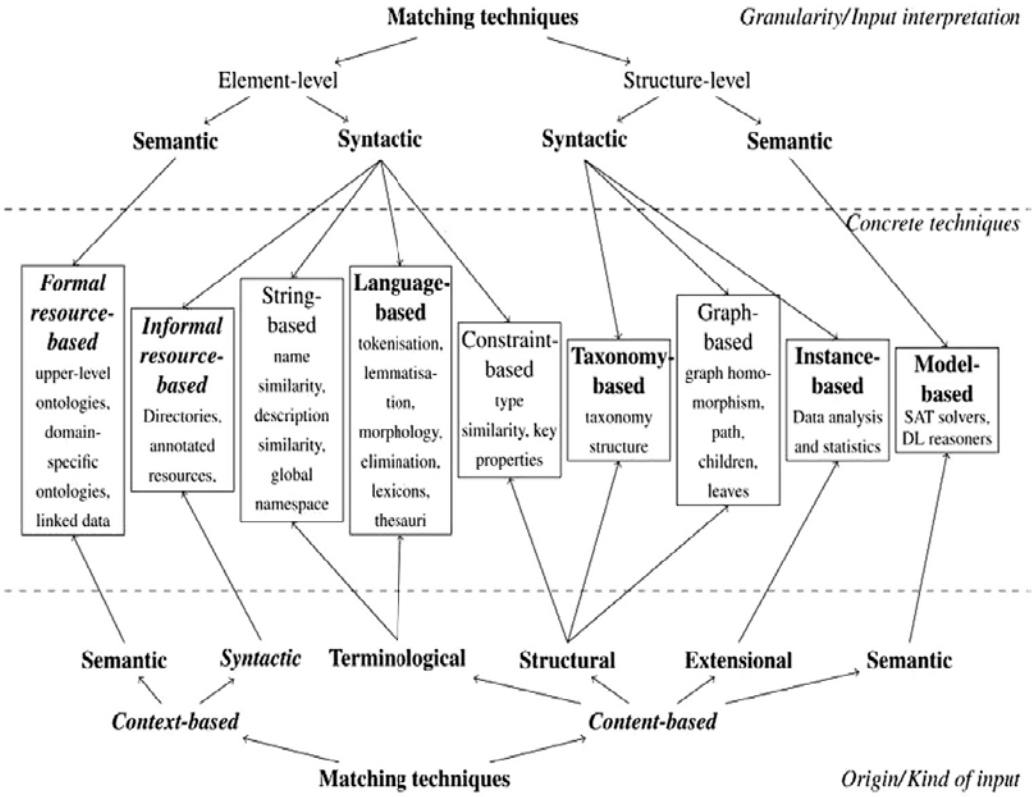


FIGURE 6.4 – Classification des techniques d'appariement d'ontologies [189]

Les systèmes FalconAO, OLA sont des systèmes utilisant les graphes bipartites et les matrices d'adjacence augmentant de ce fait la complexité du système (problème NP complet [190]). De plus, ces systèmes, ainsi que le système SIGMa, ne permettent pas d'effectuer un appariement prenant en compte les cardinalités avec lesquelles nos graphes de situations à comparer sont décrits, c'est-à-dire cardinalité (n :m), sans compter le fait qu'ils dépendent en grande partie des contenus des nœuds en négligeant leurs voisinages. Quant au système YAM++, il permet de faire des appariements sur des graphes utilisant des vocabulaires hétérogènes, exigeant de cette manière un prétraitement. En plus, ce système utilise la Similarity Flooding qui fait aussi un appariement avec des cardinalités (1 :1).

En fait, Similarity Flooding est une méthode impliquant un algorithme à deux phases : Une phase de construction d'un graphe nommé Pairewise Connectivity Graph (PCG) [245] et une phase de calcul de la propagation de similarité exigeant la cardinalité (1 :1). L'algorithme construisant le PCG répond parfaitement à nos besoins du point de vue du niveau structurel car il permet, dans le graphe construit, de garder la structure commune des graphes, d'écarter les parties communes mais liées à d'autres parties intermédiaires non communes, de ne pas dépendre du contenu des nœuds et de dépendre des types (surclasse) de nœuds. D'un autre côté, pour répondre à notre seconde hypothèse qui s'intéresse au contenu informationnel des graphes, parmi les mesures de similarité existantes telles que "Resnick" et "Jiang and Conrath", celle qui s'adapte le mieux à la problématique posée est celle de Lin [145]. Ce choix est dû au fait que d'une part, cette mesure offre une base théorique contrairement aux autres qui sont basées sur des résultats empiriques, et d'autre part, elle donne un résultat formel. Cette mesure est donc retenue pour faire l'appariement de graphes au niveau sémantique.

6.4.2.2 Principe de notre approche d'appariement de graphes

En général, l'appariement des graphes nécessite de mesurer la distance ou la similarité entre deux graphes. Cela se met en œuvre par la mise en correspondance des sommets et des arrêtes de graphes afin d'identifier leurs points communs et leurs différences.
Dans ce travail et afin de déterminer la situation représentée par un graphe RDFS la plus proche à la situation de blocage courante représentée également par un graphe RDFS, nous utilisons l'appariement de graphes. Pour ce faire, nous appliquons la mesure de similarité de Lin exigeant la connaissance des parties communes à ces dernières. Ceci nous a permis de combiner, sur cette base, les deux méthodes choisies pour répondre aux hypothèses. La méthode de calcul du Pairwise Connectivity Graph(PCG)[163] offrant un moindre cout de calcul permet de cerner les parties communes des graphes RDFS à comparer afin que ce résultat soit en entrée de la distance de Lin.
Dans ce qui suit, nous décrivons le raisonnement par analogie mis en place dans RSAW.

6.4.3 Application du raisonnement par analogie

Le principe de raisonnement par analogie dans RSAW suit le carré d'analogie (cf. figure 6.3). En effet, à partir de l'interprétation de la situation de blocage courante, le système recherche dans la base de situations une situation issue de l'expérience passée du robot et qui est similaire à la situation courante. La résolution de cette situation de blocage courante sera proche de la résolution de la situation similaire.

Pour appliquer le raisonnement par analogie dans RSAW d'une manière optimale, nous procédons comme suit :

- En premier lieu, une fonction de filtrage sur la base de la typologie conçue pour S-Ontology est effectuée. En effet, dans S-Ontology, les situations de blocage sont mémorisées selon le type de blocage rencontré par le robot (comme précisé dans le chapitre 5 Fig. 4.5) et récapitulé dans le Tableau 6.3. En effet, chaque type de situation est caractérisé par un triplet RDF représentant le blocage, c'est à dire que toutes les instances de la situation de blocage appartenant au même type de situation contiennent ce triplet RDF. Par exemple pour un type de situation de blocage au niveau de la ***Scène*** représentée par un objet bloquant la saisie et situé "devant" l'objet désiré par l'opérateur, le triplet RDF est **isFront(DesiredObject, Object)**.
- En deuxième lieu une fonction de sélection de la situation similaire est effectuée :
  - Extraction, à travers une requête SPARQL (DESCRIBE [208]), des situations non-filtrées est effectuée afin d'obtenir les graphes des situations.
  - Un calcul de similarité entre chaque situation non-filtrée et la situation de blocage courante (SB-RDFS-Graph) est effectué pour choisir la situation possédant la similarité la plus élevée avec la situation courante.
- En troisième lieu, une fonction d'aide à la prise de décision est alors effectuée dans laquelle un seuil est fixé en fonction duquel le choix est fait entre adapter le plan d'action initial en lui ajoutant la séquence de résolution de la situation ou régénérer un plan d'action.

Blocage	Type de blocage	Filtrage
Etat du robot	Base mobile	hasLevel(Battery, Low)
	Bras	Broken(Arm)
Environnement	Parcours	hasLocation(Mobile-Base, Object)
	Scène	isBehind(DesiredObject, Object) isLeft(DesiredObject, Object) isRight(DesiredObject, Object)....
Incompréhension	Incompréhension du désir de l'opérateur	Desir-Object-In (Operator, Environment) n'existe pas dans la situation
	Incompréhension des objets de la scène	hasObject(Environment,Object) N'existe pas dans la situation

TABLE 6.3 – Étape de filtrage dans le raisonnement dans RSAW

La Fig. 6.5 représente une illustration du principe du raisonnement par analogie

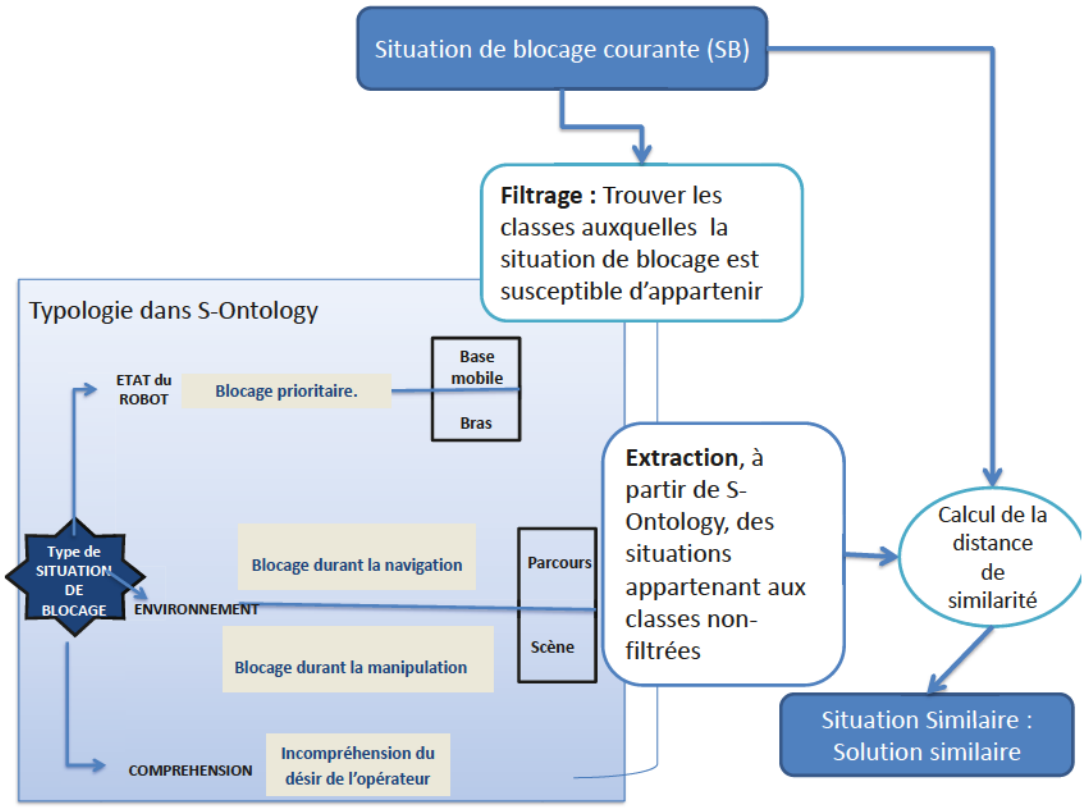


FIGURE 6.5 – Illustration du raisonnement par analogie dans RSAW

ainsi décrit dans RSAW. Cette figure montre la mise en œuvre du raisonnement par analogie sur les types des situations de blocage traités dans nos travaux et illustrés dans S-Ontology.
Dans la suite, nous présentons ces trois fonctions de filtrage, de sélection et d'aide à la prise de décision.

6.4.3.1 Fonction de filtrage

Le but de la fonction de filtrage est d'une part d'éviter d'avoir des incohérences dans les solutions que RSAW génère et d'autre part d'optimiser le temps de calcul en gardant uniquement les situations ayant le même type que la situation de blocage courante. Les types de situation sont tels que décrits dans la sous-section 4.2.3.2. Nous vérifions à quel type de situation la situation courante peut être liée. Pour ce faire, nous avons identifié une combinaison de triplet RDFS qui caractérise chaque type de situation.
Le Tableau 6.3 représente les types des situations de blocage traités dans cette thèse ainsi que les triplets RDF caractérisant chaque type de situations traité.

L'étape de filtrage se base sur cette correspondance pour déterminer l'appartenance de la situation de blocage courante.
A l'issue de l'étape de filtrage, les situations susceptibles d'être similaires à la situation de blocage courante sont extraites. Parmi ces situations, une sélection de la situation la plus similaire est effectuée. Cette sélection est fondée sur l'appariement de graphes.

6.4.3.2 Fonction de sélection

La fonction de sélection consiste à rechercher parmi les situations non filtrées la situation la plus proche. Puisque les situations sont des graphes, nous explicitons dans ce qui suit le principe de l'appariement de graphes appliqué dans RSAW.

a Appariement de graphes

Comme vu dans la sous-section 6.4.1.1, nous avons choisi la distance de Lin comme mesure de similarité en l'adaptant à la structure de nos graphes représentant les situations et en appliquant une technique d'appariement topologique par propagation de la similarité.
La Fig. 6.6 montre le principe de la mesure de similarité appliquée entre deux graphes

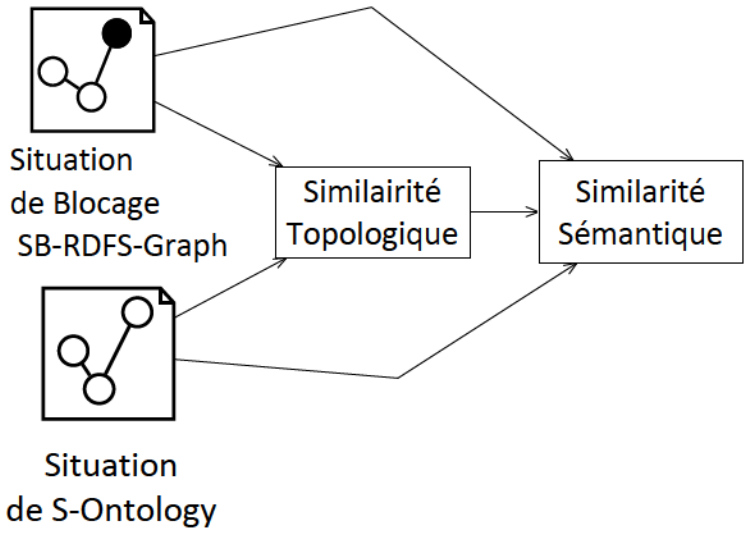


FIGURE 6.6 – Principe de l'appariement de graphes dans RSAW

dans nos travaux. Cette figure illustre la combinaison hybride des deux techniques de similarité utilisées, la première technique de similarité étant topologique par propagation de similarité et la seconde étant sémantique (distance de Lin).

b Calcul de similarité

Similarité sémantique La distance de Lin a été publiée en 1998 [145]. Cette distance est issue des définitions d'une mesure de similarité. En effet, une mesure de similarité est définie par :

- Définition 1 : La similarité entre deux concepts A et B est lié à l'intersection de ces concepts. Plus l'intersection est importante, plus ils sont similaires ;
- Définition 2 : La similarité entre deux concepts A et B est lié à la différence entre eux. Plus la différence est importante, moins ils sont similaires ;
- Définition 3 : La similarité entre deux concepts A et B est maximale lorsque A et B sont identiques, peu importe leur intersection.

D'après la définition 1 la distance de Lin dépend alors de la quantité d'information, notée I, apportée par l'intersection des deux concepts sur lesquels la mesure de similarité est calculée. $Lin(A, B)$ est proportionnel à $I(common(A, B))$. La fonction $common(A, B)$ est l'ensemble d'intersection entre les deux concepts. De la deuxième définition on déduit que $Lin(A, B)$ est inversement proportionnelle à $I(union(A, B)) - I(common(A, B))$. La fonction $union(A, B)$ représente l'union de A et B. Donc,

$$Lin(A, B) = f(I(union(A, B)), I(common(A, B)))$$

La quantité d'information, selon le théorème de Shannon, est mesurée par le logarithme (négatif) de la probabilité de l'insertion.
La formule est : $I(A) = -Log(P(A))$.
D'après la démonstration de Lin [145], la mesure de similarité est calculée suivant cette formule à :

$$Lin(A, B) = \frac{Log(P(common(A, B)))}{Log(P(union(A, B)))} = \frac{2 * Log(P(common(A, B)))}{Log(P(A)) + Log(P(B))}$$

La formule de Lin a été définie d'une façon générale quelques soient les concepts à comparer. Nous proposons dans RSAW de la spécialiser pour les graphes RDFS qui définissent les situations de blocage. Dans RSAW, nous calculons la distance de Lin entre deux graphes de situations (G_1, G_2). Donc, la distance de Lin pour ces graphes est définie par :

$$Lin(G_1, G_2) = \frac{2 * Log(P(common(G_1, G_2)))}{Log(P(G_1)) + Log(P(G_2))}$$

Pour ce faire, nous avons associé à un graphe RDFS une probabilité. Nous avons défini cette probabilité comme étant la somme pondérée des probabilités d'apparition de chaque triplet du graphe RDFS.
Nous avons conçu dans RSAW la possibilité d'affecter des pondérations pour privilégier des triplets par rapport à d'autres. Ces pondérations sont d'une majeure importance dans la pérennité de RSAW car elles permettent aux développeurs qui

utiliseront RSAW de faire des compromis entre les trois domaines (Robot, Environnement et Opérateur) selon l'application et le contexte d'utilisation.

$$P(GraphRDF) = \frac{\sum_{triplet=0}^{nbretriplet}(\alpha_{triplet} * P_{triplet})}{\sum_{triplet=0}^{nbretriplet}(\alpha_{triplet})}$$

Le but maintenant est de calculer $common(G_1, G_2)$, l'intersection entre deux ontologies. Comme annoncé dans la la section 6.4.2.1, nous nous basons sur une similarité topologique, en l'occurrence le Pairewise Connectivity Graph (PCG), pour calculer la partie commune des graphes.

Similarité topologique Afin de construire le PCG de deux graphes RDFS, une paire de nœuds (concepts) des deux graphes est fusionnée en un seul nœud lorsque cette paire de concepts a le même prédicat avec les nœuds voisins, et à son tour, leurs concepts voisins correspondants sont de même fusionnés. La Fig.6.7 montre un exemple de construction du PCG.
Puisque les situations manipulées sont sous le formalisme RDFS, donc le PCG est de même. Un nœud du PCG est noté nœud d'appariement "pairwise node". Afin de finaliser notre similarité topologique, nous comparons les termes dans le pairwise node. Si pour les deux nœuds constituant le triplet, les termes ou leurs concepts associés sont égaux, nous gardons le triplet. Comme décrit plus haut, afin de calculer la distance de Lin nous avons eu recours au calcul du PCG. La définition de la similarité dans RSAW est :

$$D_{RSAW}(G_1, G_2) = \frac{2 * Log(P(PCG(G_1, G_2)))}{Log(P(G_1)) + Log(P(G_2))}$$

où

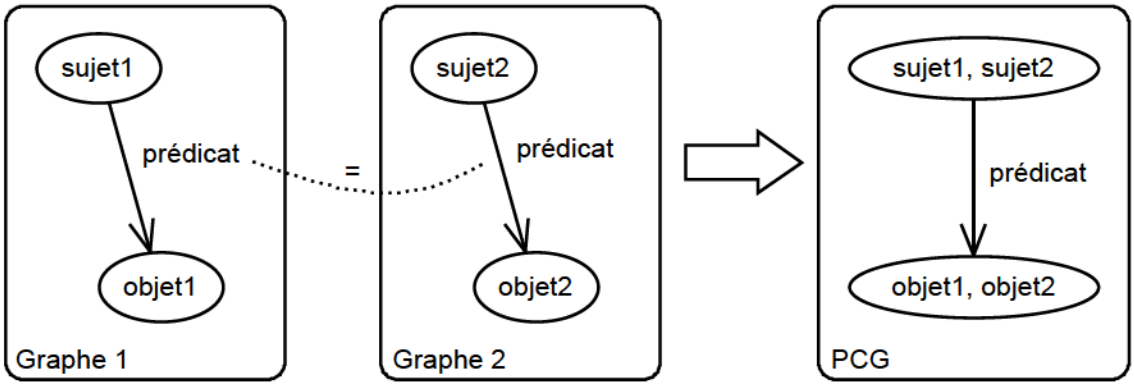
$$P(G_1) = \frac{\sum_{triplet=0}^{nbretripletG1}(\alpha_{tripletG1} * P_{tripletG1})}{\sum_{triplet=0}^{nbretripletG1}(\alpha_{tripletG1})}$$

$$P(G_2) = \frac{\sum_{triplet=0}^{nbretripletG2}(\alpha_{tripletG2} * P_{tripletG2})}{\sum_{triplet=0}^{nbretripletG2}(\alpha_{tripletG2})}$$

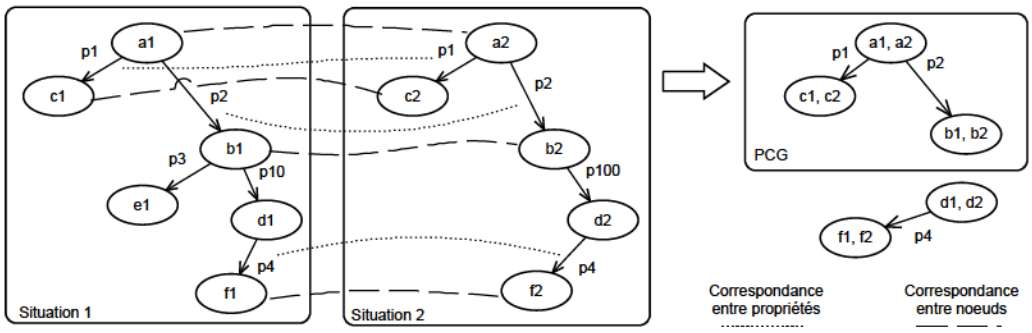
$$P(PCG(G_1, G_2)) = \frac{\sum_{triplet=0}^{nbretripletPCG}(\alpha_{tripletPCG} * P_{tripletPCG})}{\sum_{triplet=0}^{nbretripletPCG}(\alpha_{tripletPCG})}$$

6.4.3.3 Fonction d'aide à la prise de décision

Le résultat attendu de cette étape est la valeur de similarité entre la situation de blocage courante et la situation la plus semblable se trouvant dans S-Ontology. La situation la plus semble est la situation qui donne la plus grande distance de similarité avec la situation de blocage.



(a) Principe du Pairewise Connectivity Graph



(b) Exemple illustrant le principe du Pairewise Connectivity Graph

FIGURE 6.7 – Pairewise Connectivity Graph : Le principe

Nous avons défini dans la phase de projection un seuil de tolérance (T) déterminant si la situation la plus semblable à la situation de blocage courante est suffisamment proche et que RSAW peut tolérer la différence entre les deux situations pour utiliser la solution de la situation similaire dans la résolution de la situation de blocage courante. En fonction de cette distance de similarité calculée la plus élevée (DS_{sup}), le système robotique peut (1) lorsque ($DS_{sup} = 1$) utiliser directement la solution de la situation similaire, (2) lorsque ($DS_{sup} >= T$), utiliser la solution de la situation similaire après validation de l'opérateur (3) lorsque ($DS_{sup} < T$), générer un nouveau problème de planification lui permettant d'obtenir un nouveau plan d'action à partir le générateur de plan d'action.
Donc, suivant la valeur de la similarité, la phase de projection prendra la décision

d'adapter le plan d'action initial en lui ajoutant la séquence des actions nécessaires au déblocage du robot ou générer un nouveau plan d'action initial.
La détermination du seuil est très importante car il représente une estimation de l'erreur acceptable entre deux situations. Comme vu dans la section de motivation et choix de ce chapitre, le raisonnement par analogie n'est pas le raisonnement le plus fiable car il est sensible aux erreurs et justement, même chez les humains nous n'arrivons pas toujours à faire la bonne estimation de seuil. C'est ainsi, que la présence de l'opérateur devient une obligation afin de valider les solutions proposées par le robot. Dans cette approche, nous avons fixé le seuil d'une manière empirique, ceci fait l'objet d'une section dans la partie des évaluations (cf. 8.4.3).

6.5 Conclusion

Ce chapitre a été l'objet de notre troisième contribution. Cette contribution concerne le raisonnement appliqué dans RSAW pour la compréhension des situations de blocage qui se déroule en deux étapes, l'interprétation des situations de blocage perçues puis la préparation à la prise de décision, tout en permettant au robot de profiter de son expérience passée.
En premier lieu, un raisonnement déductif est utilisé pour l'interprétation des données perçues ayant permis au robot de construire la situation de blocage dans laquelle il se trouve. Ces connaissances déduites sont exprimées dans un langage symbolique commun élaboré pour homogénéiser la description des situations. En deuxième lieu, un raisonnement par analogie est appliqué pour retrouver une situation de blocage similaire dans la base des situations passées (base de cas) S-Ontology. Cette similarité est trouvée par un appariement de graphes combinant la technique structurelle (PCG) et une technique basée sémantique (LIN). Les résultats obtenus suite à cette phase de compréhension préparent la phase suivante de RSAW, phase de projection, où le robot retiendra le comportement adéquat avec l'aide de l'opérateur. Un nouveau comportement retenu sera mémorisé dans la base de cas S-Ontology pour son enrichissement.

Dans cette partie, nous avons présenté nos contributions pour rendre un robot plus autonome. Il s'agit d'une approche, RSAW, générique et facilement intégrable dans un système robotique, visant à rendre un robot conscient de la situation. Cette approche propose un processus à quatre phases où la compréhension constitue la partie intelligente censée augmenter l'autonomie du robot puisqu'elle est réalisée sans l'intervention de l'opérateur. Elle se base sur une représentation générique des connaissances conçue selon la méthodologie de l'IDM et opérationnalisée par un raisonnement fondé sur un appariement de graphes.

Dans la partie suivante, nous proposons de décrire la mise en œuvre et l'intégration de RSAW dans un système robotique ainsi que son expérimentation.

Troisième partie

ÉTUDE EXPÉRIMENTALE DE L'AIDE À L'OPÉRATEUR À PARTIR DU MODULE LOGICIEL DE COMPRÉHENSION DYNAMIQUE DU CONTEXTE

Chapitre 7

MISE EN OEUVRE DE RSAW

Sommaire

7.1 Introduction

La problématique de notre thèse est de rendre un robot conscient de la situation de blocage dans laquelle il se trouve. Plus précisément, lorsqu'un robot exécute un plan d'action correspondant à un objectif de l'opérateur et qu'une situation de blocage (une situation qui incite le robot à "réfléchir") est constatée, ceci exige de la part du robot une compréhension et une prise de décision pour lui permettre de réagir au blocage et d'atteindre son objectif.
La deuxième partie de ce rapport a été consacrée à la présentation de la recherche menée dans le cadre de cette thèse, à savoir l'approche RSAW, complète et générique, qui assure à un robot la capacité de comprendre et faire face aux situations de blocage rencontrées. Nos principales contributions dans RSAW ont porté sur la conception d'un cadre sémantique intégrant la capacité de compréhension, fondé sur une représentation des connaissances génériques et donnant la possibilité d'appliquer certaines techniques de raisonnement.

Dans cette troisième partie l'attention est portée à la mise en œuvre et l'expérimentation de RSAW. La mise en œuvre de RSAW a nécessité, durant nos recherches, la mise en place d'un système logiciel permettant de doter un robot « d'intelligence » en le rendant capable de comprendre et résoudre les situations de blocage qu'il peut rencontrer au fur et à mesure qu'il évolue dans son environnement dynamique.
Dans ce chapitre, l'intérêt est porté à la conception et la mise en œuvre du système supportant notre approche RSAW, qui se doit d'être facilement intégrable dans un système robotique. Nous donnons à ce système le même nom, RSAW, que l'approche. Dans la première section, nous présentons la conception du système RSAW où nous décrivons son architecture logicielle ainsi que les modèles conceptuels et les algorithmes qui ont précédé son implémentation. Dans la dernière section, nous présentons la mise en œuvre de l'intégration de RSAW.

7.2 Conception de RSAW

Cette section présente l'architecture logicielle ainsi que la modélisation du système RSAW développé.

7.2.1 Architecture logicielle

Comme vu dans la section 4.4, la principale exigence pour concevoir le système RSAW est qu'il doit être facilement intégrable dans un système robotique. La conception d'une architecture logicielle pour RSAW est donc d'une importance capitale. Elle doit respecter, en plus de la modularité, certains critères de qualité dont principalement les critères de :

- Réutilisabilité : la capacité à rendre génériques des composants vis-à-vis de l'application et à construire des boites noires autonomes susceptibles de fonctionner dans différents environnements.
- Maintenabilité : la capacité de modifier et de maintenir en production une application sur une période de vie assez longue. La prévision de l'intégration d'extensions nécessaires à l'architecture et l'amélioration des composants existants sont primordiales.
- Performance : caractérisée par le temps mis par une application à répondre à une requête donnée.

La conception générale de la principale composante de RSAW, à savoir la conscience de la situation, est schématisée en UML sur Fig. 7.1. Cette conception tient compte des critères précédemment énoncés et présente les packages conçus, ainsi que leurs relations. Elle est basée sur une architecture orientée services, le composant RSAW joue le rôle de service orchestrateur.
La figure 7.1 illustre les packages relatifs à la phase de perception, la phase de compréhension (raisonnement déductif et raisonnement par analogie), la phase de

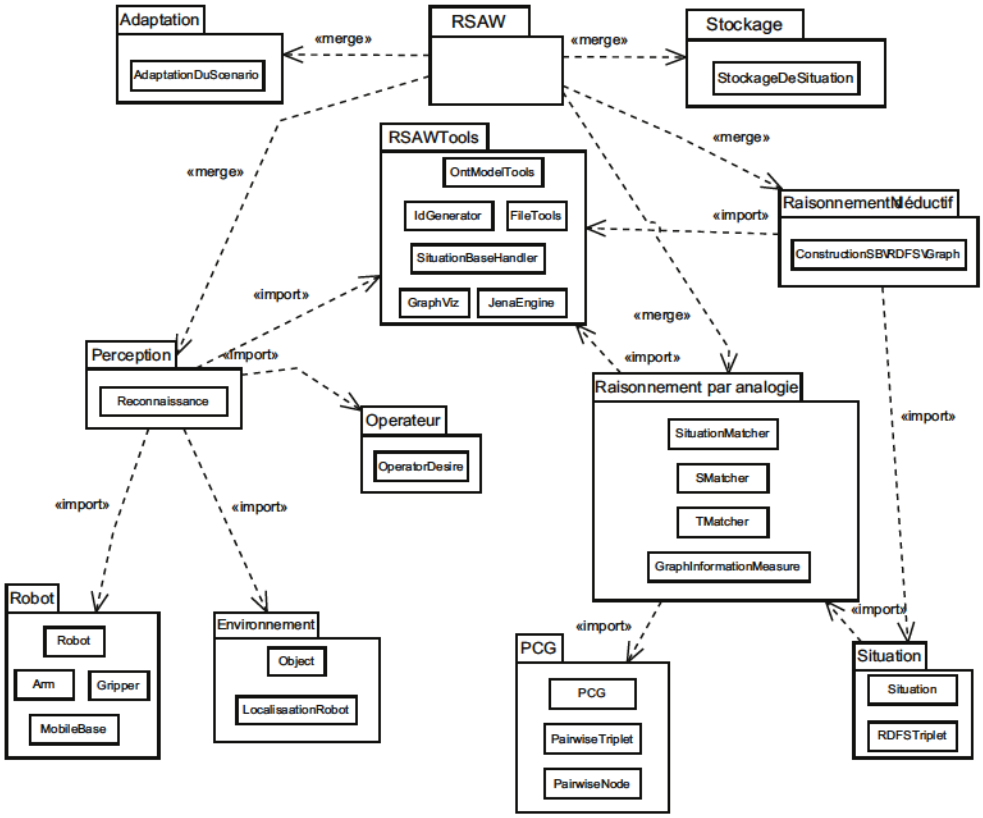


FIGURE 7.1 – Conception générale de RSAW

projection (stockage de la situation et adaptation du plan d'action initial). A titre d'exemple, si l'utilisation d'une autre technique de calcul dans le raisonnement par analogie est souhaitée pour améliorer la performance de RSAW, il suffit de remplacer ou modifier le package de raisonnement par analogie.

Le système RSAW gère aussi les connaissances représentant l'état du robot, de l'environnement dans lequel évolue le robot et de l'opérateur qui, comme vu dans le chapitre 5, contribuent à la compréhension de la situation.

L'ontologie représentant la situation S-Ontology est conçue comme un Méta Modèle auquel sont conformes les ontologies du robot (R-Ontology), de l'environnement (E-Ontology) et de l'opérateur (OP-Ontology).

La Fig .7.2 illustre l'architecture du système de RSAW. Elle montre, d'une part, la partie de représentation de connaissances suivant l'Ingénierie Dirigée par les Modèles et d'autre part le processus d'activité de RSAW opérationnalisant les ontologies de la représentation des connaissances en utilisant les différents raisonnement (déductif et par analogie).

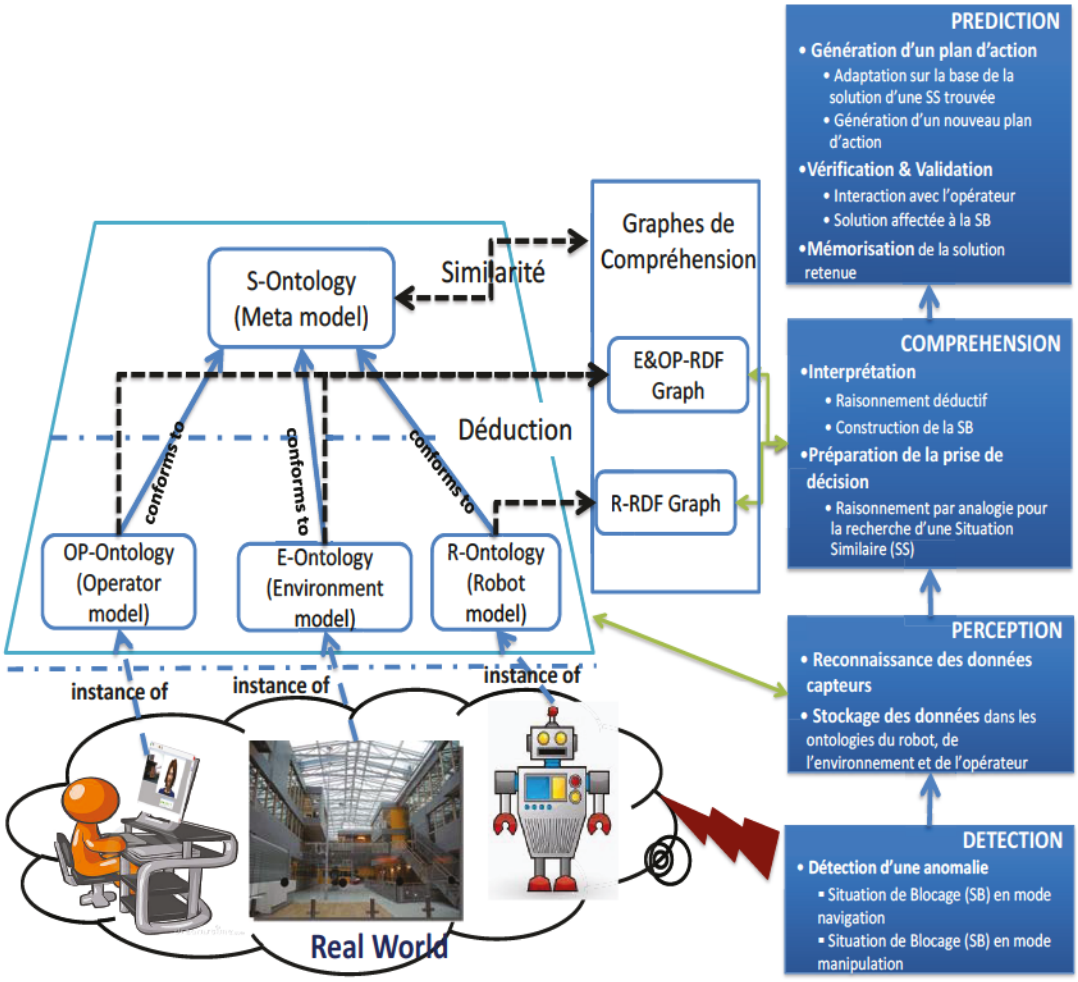


FIGURE 7.2 – Système RSAW

Le raisonnement déductif est utilisé pour créer les graphes de compréhension et le raisonnement par analogie est utilisé obtenir la solution de la situation de blocage courante.

7.2.2 Modélisation du système

Dans cette section, nous abordons les modèles conceptuels et les algorithmes pour les différentes phases de RSAW. Rappelons que RSAW propose un processus de fonctionnement qui devra être exécuté par le robot pour répondre au besoin de l'opérateur. Ce processus est en fait le moyen utilisé pour permettre d'intégrer les fonctions de compréhension pour que l'autonomie du robot puisse augmenter et évoluer grâce à la faculté de raisonnement et d'apprentissage. En effet, le modèle de processus suivi étant orienté activité [113], ceci permet d'assurer son automatisation à condition de

disposer d'une bonne représentation des connaissances.

7.2.2.1 Phase de détection

Lors de cette phase préalable, RSAW reste à l'écoute pour détecter les situations dans lesquelles le robot se trouve bloqué. Le robot peut se trouver bloqué lorsqu'il se déplace (en mobilité) ou pendant la saisie d'un objet (en manipulation). La détection des situations de blocage se fait conformément aux algorithmes ci-dessous.
Pour détecter l'anomalie en cours de mobilité, nous proposons un algorithme DETECT-MobDS (Alg. 1) basé sur les caractéristiques du mouvement, c'est-à-dire la trajectoire et la vitesse en fonction du temps (cf. section 4.2.2.2). Nous notons *TP (t)* la trajectoire prévue en fonction du temps, *TR(t)* la trajectoire réelle en fonction du temps, la notation de *V(t)* pour la vitesse à l'instant t, *V_limit* la vitesse limite acceptable (seuil) et *diverge (TR (t), TP (t))* la fonction qui détecte si la trajectoire réelle ne suit plus la trajectoire prévue.
L'événement d'anomalie durant la mobilité du robot est déclenché si *diverge (TR(t), TP(t))* renvoie vrai ou si la vitesse instantanée ($V(t)$ est inférieure à la valeur seuil (V_limit).

Alg. 1 DETECT-MobDS

Input: TP(t), TR(t), V(t), V_limit ;
Output: Event ;

```
Event ← Not_Anomaly_Event;
While Event = Not_Anomaly_Event Do
    If (diverge(TR(t),TP(t))Ou(V(t) < V_limit)) Then
        Event ← Anomaly_Event; {Le robot est dans une situation de blocage}
        Perception()
    End If
End While
```

Pour détecter la situation de blocage lors de la manipulation, nous proposons l'algorithme DETECT-ManDS (Alg. 2) qui vérifie le bon fonctionnement des fonctions de contrôle. Par exemple, lorsque le logiciel de suivi de l'objet tombe en panne, RSAW détecte un événement d'anomalie et la perception est donc déclenchée. Dans l'Alg. 2, nous notons *TrackingObject(t)* le retour visuel de la méthode de suivi de l'objet à l'instant t. Initialement, l'objet à saisir fait partie du retour visuel des caméras du robot. Lorsqu'au cours du mouvement, l'objet désiré n'est plus dans le retour visuel du robot, cela génère un événement d'anomalie. Cette condition limite est notée par LostObject.

Alg. 2 DETECT-NanDS

Input: *TrackingObject(t)*
Output: Event
`Event` ← `Not_Anomaly_Event`
While *Event* = *Not_Anomaly_Event* **Do**
If (*TrackingObject*(*t*) = *LostObject*) **Then**
`Event` ← `Anomaly_Event` {Le robot est dans une situation de blocage}
Perception()
End If
End While

La détection d'un blocage (Event=Anomaly_Event) est un événement qui déclenche la phase de perception.

7.2.2.2 Phase de perception

Il s'agit de la perception ou la reconnaissance des données à partir des capteurs afin d'instancier les modèles du Robot, de l'environnement et de l'opérateur. Ceci permet d'associer des valeurs aux concepts décrits dans le chapitre 5 qui leur correspondent, en vue de préparer la phase de compréhension de la situation de blocage (cf. 4.2).
Ainsi, les données récupérées à partir des capteurs proprioceptifs couvrent l'état du système robotique (l'état de chaque structure par exemple le bras du robot ; la pince ; la partie mobile). Celles récupérées à partir des capteurs extéroceptifs couvrent l'état de l'environnement. En effet, les capteurs ultrasons de la base mobile permettent de retrouver la localisation du robot dans la carte de l'environnement (Map). Aussi, l'image de la scène (ScenePicture) prise par la caméra du robot permet de reconnaitre les objets se trouvant devant le robot. Enfin les données récupérées à partir de l'Interface Homme-Machine (IHM) couvrent l'état de l'opérateur.
Les données récupérées par la fonction de reconnaissance permettent d'alimenter les différentes ontologies (R-Ontology, E-Ontology et OP-Ontology) avec les valeurs associées à la situation de blocage courante. La Fig 7.3 montre le diagramme de classe relatif à cette phase. Dans ce diagramme, nous trouvons les classes correspondant au robot, à l'environnement et à l'opérateur. Chacune de ces classes est composée de concepts la représentant tels que décrit dans la section 4.2.2.2 "Concepts d'une situation".
L'algorithme (Alg. 3) décrit la récupération des données capteurs et l'instanciation des différentes ontologies (R-Ontology, E-Ontology et OP-Ontology) pour préparer la phase de compréhension. C'est ainsi qu'à son tour, à la fin de l'exécution de cet algorithme, la phase de compréhension est déclenchée afin d'interpréter et raisonner sur ces valeurs de la situation de blocage courante.

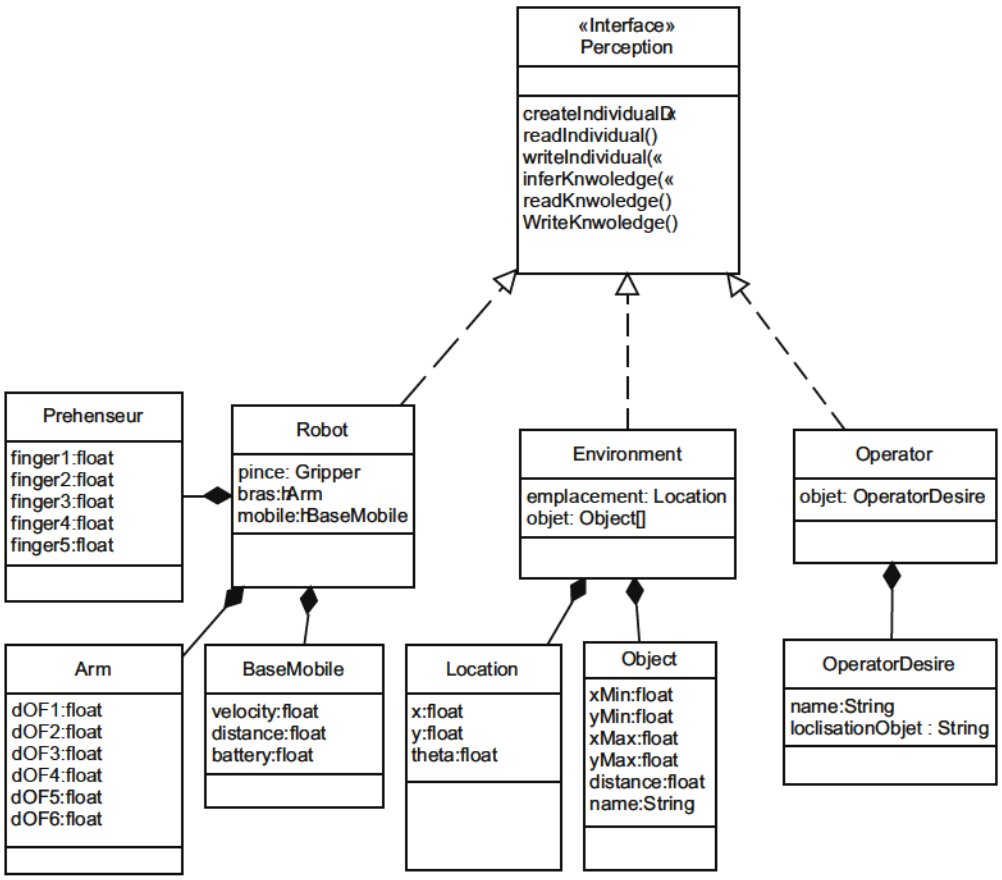


FIGURE 7.3 – Diagramme de classe de la phase de perception dans RSAW

7.2.2.3 Phase de compréhension

La phase de compréhension dans RSAW se déroule en deux étapes : l'interprétation et la préparation de la prise de décision. L'interprétation sert à déduire, à partir des connaissances mémorisées précédemment dans les trois ontologies (du robot (R-Ontology), de l'environnement (E-Ontology) et de l'opérateur (OP-Ontology), le graphe RDFS représentant la situation de blocage courante noté par (SB_RDFS_Graph). Elle est assurée au moyen d'un raisonnement déductif. Ce graphe construit est utilisé pour chercher une situation, la plus analogue à la situation courante (définie dans (SB_RDFS_Graph)), se trouvant dans la base des cas (S-Ontology) contenant les situations de blocage passées avec les solutions adoptées pour les dépasser. Ceci est fait dans l'étape de la préparation de la prise de décision au moyen d'un raisonnement par analogie. Ainsi, comme vu dans le chapitre 6, la phase de compréhension se déroule

Alg. 3 Perception

Input: Map, ScenePicture, IHM
Output: R-Ontology, E-Ontology, OP-Ontology

{Récupération des données liées à la localisation du robot, aux objets de l'environnement et création des instances de E-Ontology}
`MobileLocation ← Lire_MobileLocation (Map, X, Y, Theta)`
`Objets ← Reconnaissance_Objets (ScenePicture, BoundingBox, Nom, Distance)`
`Enrichissement-E (MobileLocation, Objets, E-Ontology)`

{Récupération du nom de l'objet désiré par l'opérateur et création des instances de OP-Ontology}
`objetDesir = LireObjetOperator (IHM, NomObjet)`
`Enrichissement-OP (objetDesir, OP-Ontology)`

{Récupération des données liées au robot et création des instances de R-Ontology}
`EtatMobile ← Lire_MobileEtat (DistanceToDestination, Battery, Velocity)`
`EtatBras ← Lire_BrasEtat (DOF, GripperPos)`
`Enrichissement-R (EtatMobile, EtatBras, R-Ontology)`

Alg. 4 Comprehension

Input: R-Ontology, E-Ontology, OP-Ontology, R_Rules, EOP_Rules S-Ontology ;
Output: SituationSimilaire ;

{Construction du graphe RDFS de la situation de blocage par application du raisonnement déductif}
`R_RDFS_Graph ← InferKnowledge (R-Ontology, R_Rules)`
`EOP_RDFS_Graph ← InferKnowledge (E-Ontology, OP-Ontology, EOP_Rules)`
`SB_RDFS_Graph ← ConstructSB (R_RDFS, EOP_RDFS_Graph)`

{Recherche d'une situation de blocage similaire dans S-Ontology par application du raisonnement par analogie}
`SituationSimilaire ← RSAW_RA (S-Ontology,SB_RDFS_Graph)`
return `SituationSimilaire`

en enchaînant en série un raisonnement déductif à base de règles d'inférences et un raisonnement par analogie servant à trouver la situation la plus analogue. L'Alg. 4 illustre l'appel aux règles d'inférence pour la constitution de la situation de blocage

courante (SB_RDFS_Graph). Une fois SB_RDFS_Graph construit, un raisonnement par analogie permet de trouver la situation la plus proche afin d'utiliser sa solution. L'Alg 5 permet de trouver la situation la plus similaire à la situation de blocage courante depuis S-Ontology.
La méthode CalculSimilarity entre deux situations dépend de la méthode choisie dans l'appariement de graphe. Dans nos recherches, comme précisé dans le chapitre 6, l'ap-

Alg. 5 RSAW-RA

Input: S-Ontology, SB_RDFS_Graph
Output: SituationSimilaire

{Réduire l'espace de recherche aux situations mémorisées de même type que la situation de blocage courante}
`SetOfSameType` ← `Filtre(S-Ontology, SB_RDFS_Graph)`

{Sélection d'une éventuelle situation similaire par application d'un calcul de similarité}
{Sim : Distance entre deux situations et Sim ∈ [0, 1]}
$HighSimilarity \leftarrow 0$;
For each $S \in SetOfSameType$ **Do**
  $Sim \leftarrow CalculSimilarity(SB\_RDFS\_Graph, S)$
  If $HighSimilarity \leq Sim$ **Then**
    $HighSimilarity \leftarrow Sim$
    $SituationSimilaire \leftarrow S$
  End If
End For
return `SituationSimilaire`

pariement de graphes est mis en œuvre par une combinaison hybride d'une technique topologique consistant à créer le Pairwise Connectivity Graph (PCG) et d'une technique sémantique consistant en l'utilisation de la distance de Lin.
La construction du PCG est décrite dans l'annexe 10. Le diagramme de classe de cet appariement est décrit dans la Fig. 7.4.

7.2.2.4 Phase de projection

Dans la phase de projection, la décision relative au comportement du robot est prise. Elle permet la génération du plan d'action qu'utilisera le robot. Aussi et pour mémoriser l'expérience du robot (apprentissage), cette phase assure le stockage de la situation de blocage courante avec sa solution adoptée et validée par l'opérateur dans la base des cas, à savoir l'ontologie des situations (S-Ontology). La génération d'un plan d'action pour le robot consiste soit en l'adaptation du plan d'action initial, soit en la planification d'un nouveau plan d'action en se basant sur la situation courante comme "état initial" et l'objectif de l'opérateur comme "état final".

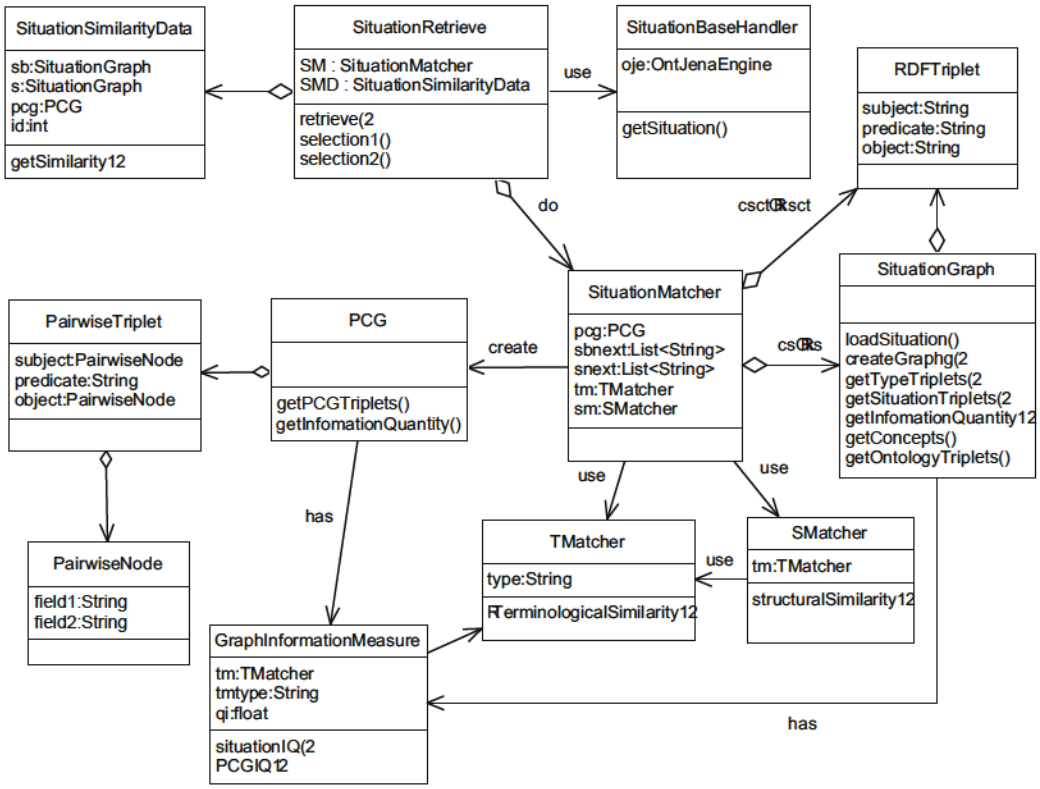


FIGURE 7.4 – Diagramme de classe du système d'appariement de graphes

En effet, comme vu dans la section 4.3.2, trois cas de figures peuvent se présenter selon le seuil de tolérance T (T dans Alg. 6)défini dans 6.4.3.3. Dans nos travaux, ce seuil a été fixé suite aux expérimentations, menées et décrites dans le prochain chapitre (chapitre 8, section...).

Dans **le premier cas**, la valeur de la similarité calculée, dans Alg. 6, est maximale (= 1). La décision est alors prise automatiquement et le système génère la solution de SB_RDFS_Graph à partir de la solution de la situation la plus similaire. Le diagramme de classe de la génération du nouveau scénario est illustré dans la fig 7.5.

Dans **le deuxième cas**, le calcul de similarité entre SB_RDFS_Graph et les situations se trouvant dans S-Ontology est supérieur au seuil de tolérance T. Dans ce cas, le système génère la solution comme dans le premier cas mais la vérification de cette solution par l'opérateur est alors nécessaire.

Dans **le troisième cas**, la similarité est inférieure au seuil de tolérance T, l'appel au générateur de plan d'action (CPT [249]), est alors nécessaire et l'intervention de

Alg. 6 PriseDecision

Input: Similarity, SituationSimilaire, PDDLDomain, InitActplan ;
Output: NewActplan ;

{Le seuil de tolérance T∈ [0, 1]}

If *Similarity* = 1 **Then** {Solution adoptée telle qu'elle}
`NewActplan ← Concat(Sol(SituationSimilaire), InitActplan)`

Else
If *Similarity* ≥ *T* **Then**
`NewActplan ← Concat(Sol(SituationSimilaire), InitActplan)`

Else
{Similarity < T (seuil de tolérance) : Génération d'un nouveau plan d'action avec CPT}
`Generate (PDDLProblem)`
`NewActplan ← CPT (PDDLDomain, PDDLProblem);`
End If

{Solution adoptée si elle est validée par l'opérateur}
{Sinon l'opérateur aura la charge de créer un nouveau plan d'action}
If *ValidateByOp*(*NewActplan*) **Then**
`Sol(SB-RDF-Graph) ← Sol(SituationSimilaire)`
`Store(SB-RDF-Graph, S-Ontology)`
End If
End If
return *NewActplan*

l'opérateur est obligatoire pour la prise de décision.
Pour faire appel au générateur de plan d'action une nouvelle description du domaine et du problème PDDL est obligatoire. Ceci est fait comme suit : Dans une même application d'un scénario la description du domaine (en PDDL) reste inchangée, en revanche la description en PDDL du problème (PDDL du problème) change en fonction du la situation de blocage. Le PDDL du problème est généré à partir de la situation de blocage courante (SB_RDFS_Graph). En effet, les nœuds du graphe représentent les objets dans le PDDL du problème. Le sous graphe comportant le désir de l'opérateur représente l'état final et l'autre sous graphe représente l'état initial. La sémantique des relations entre les nœuds de SB_RDFS_Graph doit exister dans le PDDL domaine pour obtenir un PDDL cohérent.
Le PDDL du problème est en fait un fichier contenant des informations, relatives aux objets du monde, à l'état initial et à l'état final du problème. Ces informations sont

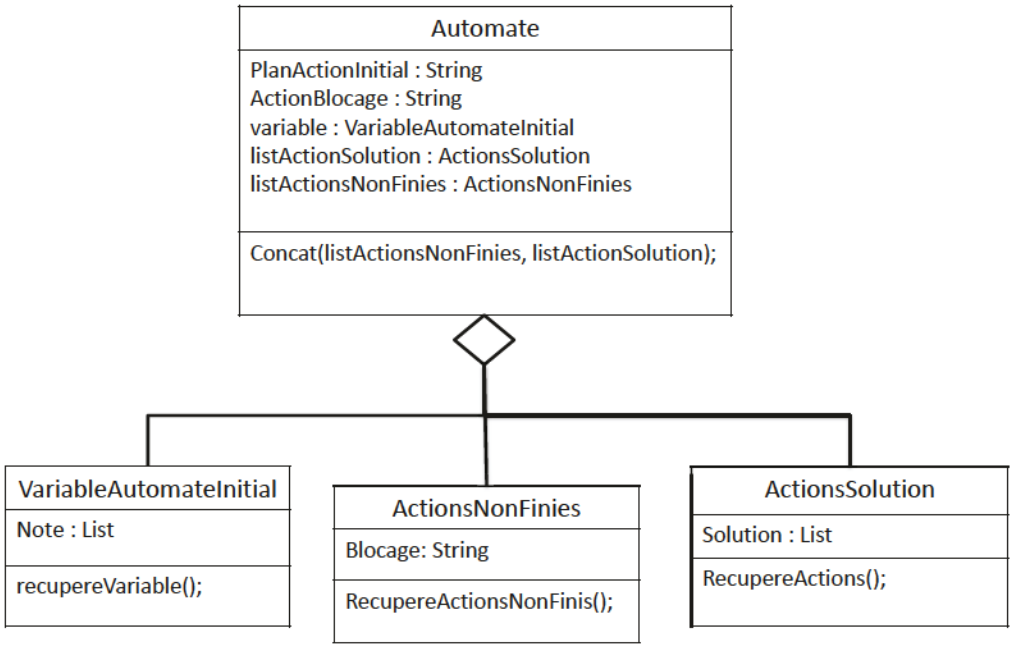


FIGURE 7.5 – Diagramme de classe du système d'adaptation du plan d'action initial

tirées à partir du graphe RDFS de la situation de blocage courante (SB_RDFS_Graph). L'algorithme de génération du PDDL problème est illustré dans Alg. 7.

La génération du plan d'action à partir de la solution de la situation la plus semblable se fait par l'intégration, dans le plan d'action initial, des actions à effectuer pour que le robot résolve la situation de blocage.

Comme le montre la Fig. 7.6, à partir du scenario écrit en ISEN XML et les actions de la solution adoptée, correspondant à la solution de la situation de blocage similaire retenue dans la base des cas S-Ontology, ce générateur concatène les actions à effectuer avec la séquence d'actions qui reste à faire pour accomplir le désir de l'opérateur. L'algorithme 8 représente le pseudo code d'adaptation du plan d'action initial par l'ajout de la solution de la situation de blocage courante.

7.3 Exigences d'intégration dans un système robotique

Comme vu dans le chapitre "Approche RSAW pour la compréhension du contexte pour la robotique" dans la sous-section 4.4.1, le système supportant l'approche RSAW

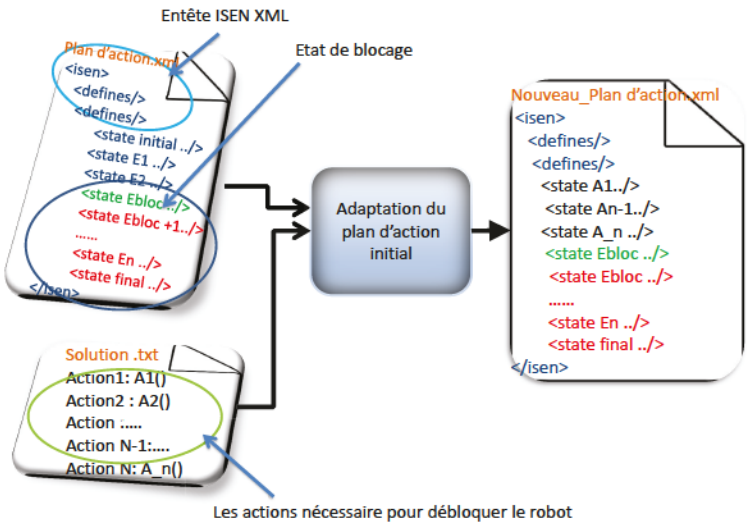


FIGURE 7.6 – Génération du nouveau plan d'action

a besoin de deux composants logiciels pour s'intégrer dans le logiciel du robot. Ces composants sont le générateur de plan d'action et le séquenceur.

Dans cette section, nous présentons le générateur de plan d'action et le séquenceur utilisés au CEA dans le Laboratoire de Robotique Interactive (LRI).
Le générateur de plan d'action est Constraint Planning Temporal (CPT) et le séquenceur est Interactive Scenarization ENgine (ISEN).

Alg. 8 Concat

Input: PlanActInit, EtatdeBlocage, SolutionDeBlocage ;
Output: new_PlanAction ;

{Récupérer les actions qui restent à faire dans le plan d'action initial}
`ActionNonFinie ← ExtractAct(EtatdeBlocage, PlanActInit)`

{Récupérer les actions à faire, extraites de la solution de la situation de blocage la plus analogue}
`ActionSol ← LireAct(SolutionDeBlocage)`

{Créer le nouveau plan d'action}
`new_PlanAction ← Enchainer(ActionSol, ActionNonFinie)`
return `new_PlanAction`

Alg. 7 PDDLProblem-Generation
Input: SB_RDFS_Graph ;
Output: new_PDDL_Problem ;

{Définition des objets du monde extraits à partir des connaissances des noeuds de SB_RDFS_Graph}
`Objet ← ExtractNodes (SB_RDFS_Graph)`

{Définition de l'état initial extrait à partir des connaissances relatives au robot et à l'environnement dans SB_RDFS_Graph}
`InitState ← ExtractEnvironmentRobotGraph (SB_RDFS_Graph)`

{Définition de l'état final extrait à partir des connaissances relatives au désir de l'opérateur dans SB_RDFS_Graph}
`FinalState ← ExtractOperatorGraph (SB_RDFS_Graph)`

{Transformation des informations extraites au format PDDL}
`InitState ← transform (InitState)`
`FinalState ← transform (FinalState)`

{Vérification et génération du PDDL problème}
If (Verification (FinalState) et Verification(InitState)) **Then**
{Les prédicats sont valides et la syntaxe est correcte. La génération d'un nouveau plan d'action est possible}
`CreateNewFile(new_PDDL_Problem)`
`new_PDDL_Problem = Insert (Objet, InitState, FinalState)`
Else
`DisplayMsgToOperator()`
End If
return `new_PDDL_Problem`

7.3.1 Architecture d'Intégration

La Fig. 7.7 montre le fonctionnement de l'ensemble des composants implémentant les activités du processus de RSAW. Elle illustre également l'importance des ontologies ainsi que l'interaction avec le système de génération de plan d'action et du superviseur.

7.3.2 Composants de liaison

Dans cette sous-section, nous présentons les composants nécessaires à notre système pour s'intégrer dans une application robotique.

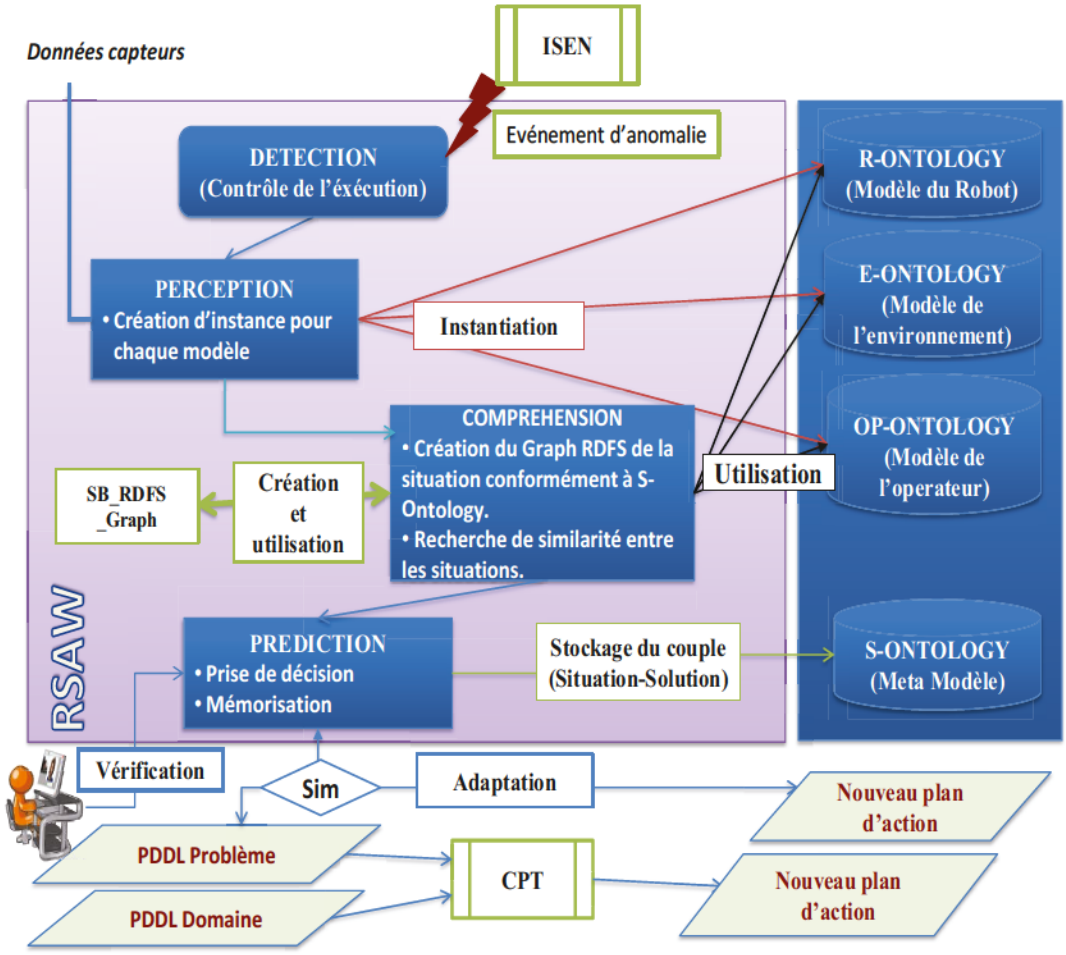


FIGURE 7.7 – Architecture fonctionnelle de RSAW

7.3.2.1 Générateur de plan d'action (CPT)

La génération de plan d'action a été mise en place dans le cadre du projet ARMEN [175]. CPT est un système de planification temporelle faisant suite à STRIPS (STanford Research Institute Problem Solver) développé par Nils Nilson et Richard Fikes [85]. Il s'appuie sur la programmation par contraintes. Il combine un système de branchement basé sur une planification d'ordre partiel (POCL) [248] et sur des règles puissantes mises en œuvre comme des contraintes.
Le CPT pallie à la limitation des approches heuristiques et la planification temporelle qui souffrent d'un facteur de ramification [20] et ont des difficultés à atteindre les performances des planificateurs construits sur des techniques de planification par satisfaction de bases de clauses (planification SAT) tels que SATPLAN [8]. Les planificateurs hiérarchiques, comme HTN (Hierarchical Task Network), considérant des

connaissances supplémentaires sur les tâches (par exemple, une séquence de sous-tâches de bas niveau décompose une tâche de haut niveau) pour réduire la complexité de la recherche du plan d'action. En revanche, le planificateur CPT est indépendant du domaine et assure de meilleures performances (rapidité de recherche de plan)[248]. En effet, CPT a remporté le prix de la performance distinguée à IPC'06 [1]

A l'entrée de CPT, nous trouvons comme dans tout système de planification, le do-

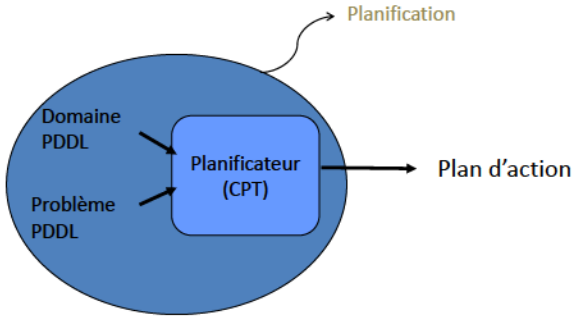


FIGURE 7.8 – CPT : Générateur de plan d'action

maine et le problème de planification. La mise en place a été effectuée par les fichiers PDDL (Planification domaine Definition Language). Le choix de PDDL, pour coder les tâches de planification avec toutes leurs composantes, est guidé par le fait qu'il s'agit d'un langage de planification standard en IA et par la contrainte de l'utilisation du planificateur (CPT) à notre disposition.

En entrée, CPT prend deux fichiers (illustrés dans 7.8), le fichier du PDDL du domaine et celui du PDDL problème :

- Le domaine PDDL comprend une liste de :
  - Prédicats : Les propriétés pouvant être appliqués aux objets.
  - Actions : Les actions à effectuer pour atteindre le but.
- Le problème PDDL est composé de :
  - Les objets du monde.
  - l'état initial.
  - La spécification du but.

Le domaine PDDL contient les actions et les prédicats. Ainsi, le robot doit avoir un PDDL complet et cohérent afin qu'il puisse prendre une décision appropriée. Pour sa part, le problème PDDL sert à décrire le problème à résoudre par le planificateur. En effet, il contient les objets du monde, l'état initial et l'état objectif. La syntaxe des prédicats est :

(Prédicat ?objet)

Les prédicats concernant :

- La déclaration d'un objet « a » comme étant un « arm » : (arm ?a).

1. International Planning Competition, http ://ipc.icaps-conference.org/

- La déclaration qu'un objet (a) est à l'état initial et à l'état de transport : (init-arm ?a) (transport-arm ?a)
- La déclaration d'une porte est fermée à la position pt : (door-closed ?pt)

Une action en PDDL est définie par les paramètres en jeu qui sont les objets du monde, les préconditions de l'action exprimer par un ensemble de prédicats et les effets de l'action exprimer aussi par un ensemble de prédicats. Ci-dessous l'exemple de l'action init-grasping pour mettre le bras qui n'est pas à l'état initial à l'état initial.
(:action init-grasping
:parameters (?a)
:precondition (and (arm ?a) (not-init-arm ?a))
:effect (and (init-arm ?a) (not (not-init-arm ?a))(not-transport-arm ?a)(not(transport-arm ?a))))
L'ensemble des actions utilisées dans nos travaux sont illustré en annexe B. Les objets du monde du PDDL problème est la syntaxe des objets qui sont utilisés pour la description de l'état initial et l'état finale. Un exemple de l'ensemble des objets du monde :

(:objects kitchen operator-position coca gripperjaco jaco) L'état initial est un ensemble de prédicats décrivant l'état dans lequel se trouvent le robot, l'environnement et l'opérateur initialement. Un exemple d'état initial : (:init
(arm jaco)
(transport-arm jaco)
(not-init-arm jaco)
(not-gripper-open)
(gripper gripperjaco)
(free gripperjaco)
(room kitchen)
(room operator-position)
(at-robot operator-position)
(object coca)
(clear coca)
(at coca kitchen)
)
L'état final est un ensemble de prédicats le désir de l'opérateur. Ci-dessous un exemple d'état final : (at coca operator-position)

7.3.2.2 Séquenceur (ISEN)

Nous avons utilisé dans nos travaux, le logiciel ISEN pour séquencer et déclencher les actions du robot. ISEN a été développé dans le laboratoire de Simulation Interactive au CEA. ISEN constitue le moteur pour appliquer les scénarios du système robotique. ISEN introduit la notion de scénario qui est la description du plan d'action (comportement) décrite dans un fichier XML. Il permet de décrire le comporte-

ment du robot grâce à un enchaînement d'actions gérées par l'envoi et la réception d'évènements. Les actions correspondent à des commandes envoyées à l'entité "Robot". ISEN est codé en C++ et permet de faire l'interface entre les scénarios et les composants du robot.
Ainsi, l'opérateur interagit avec le système via l'Interface Homme Machine. Les ordres donnés par l'utilisateur sont envoyés à ISEN qui constitue le cerveau de l'application. ISEN envoie alors les ordres au robot qui fournit en retour des informations sur son état.
Avec le logiciel ISEN, il est possible de créer un modèle d'automate hiérarchique (automate à pile) qui introduit donc la notion de sous-automate. Lorsque survient un événement prioritaire, on stocke l'état courant dans une pile, on lance le sous-automate qui traite l'événement, et quand celui-ci est terminé, on retourne à l'état que l'on a précédemment stocké au sommet de la pile. L'usage d'une pile permet d'empiler un nombre infini de traitements d'interruption de ce type, la machine est donc capable de gérer une combinatoire infinie des états élémentaires qui ont été décrits. La machine à état d'ISEN a été constituée à partir de cette notion d'automates à pile.
Ainsi, on trouve dans ISEN la notion de ressource permettant d'éviter l'exécution de deux actions appelées en même temps. La notion de priorité règle le problème de conflit dans le cas où deux actions requièrent la même ressource. L'état prioritaire détient la ressource puis la relâche lorsque l'on passe à l'état suivant. A ce moment-là, l'autre état qui détenait la ressource et qui était appelé en même temps, récupère la ressource et on passe alors dans cet état. L'utilisation d'ISEN permet donc d'aborder la modélisation de comportements complexes, réalisant des actions en parallèles si elles ne sont pas concurrentes.
Le passage entre le plan d'action et le scénario a été développé dans [176]. Un composant de traduction est développé pour générer un scénario ISEN à partir d'un plan générer par le composant CPT. Le principe consiste à correspondre chaque action du plan d'action généré par CPT à des états. Philippe Morignot le développeur de ce traducteur a établi un fichier nommé Handlers contenant toutes les variables, les constantes et les états associés aux actions nécessaires pour l'élaboration du fichier scénario ISEN.

7.4 Conclusion

Dans ce chapitre, l'intérêt a été porté sur la mise en œuvre d'un système informatique supportant l'approche RSAW et son intégration dans un système robotique. Après avoir présenté l'architecture logicielle de ce système, une conception UML de ses principaux composants et structures de données ainsi que les principaux algorithmes élaborés sont décrits. Ce système a été développé indépendamment de tout système robotique. Pour son intégration dans un système robotique, deux composants

de liaison sont développés, le séquenceur (ISEN) et le générateur de plans d'action (CPT), pour le système AVISO utilisé dans le robot Smart Autonomous Majordomo (SAM) acquis par le CEA-LIST.

Le chapitre suivant traite des expérimentations menées dans le cadre de ce travail de thèse. Ces expérimentations sont basées sur l'utilisation de SAM avec le système AVISO.

Chapitre 8

EXPÉRIMENTATION DE RSAW

Sommaire

8.1 Introduction

Notre solution a été testée et validée sur le robot d'assistance SAM développé dans le cadre du projet ANR ARMEN (Assistant Robotique pour le Maintien en Environnement Naturel) [96] dont l'objectif consiste à développer un robot fournissant des fonctions avancées pour aider à maintenir les personnes dépendantes (âgées ou handicapées) à domicile.
Le robot d'assistance "Smart Autonomous Majordomo" (SAM) utilisé dans ARMEN vise ainsi à compenser la disponibilité limitée de la personne chargée d'assister la personne dépendante (l'aidant). Ceci donne l'occasion aux aidants de restructurer leur calendrier pour se concentrer sur les besoins des personnes dépendantes dont ils ont la charge. L'un des principaux efforts d'ARMEN a été de faciliter l'utilisation d'un dispositif très technique (le robot SAM) par un opérateur non-spécialiste [134]. C'est ainsi que, dans un premier temps, nous avons participé au développement et à l'intégration, dans AVISO [139], de nouveaux composants logiciels assurant certaines fonctionnalités de base, à savoir les composants de contrôle du robot (base mobile et bras), le composant de reconnaissance d'objets, le séquenceur (ISEN), l'Interface Homme-Robot (IHM) et le générateur de plans d'action (CPT). L'IHM proposée permet aux opérateurs d'exprimer simplement son besoin au robot. Ensuite, nous avons développé le système supportant l'approche RSAW que nous avons intégré dans AVISO comme décrit dans le chapitre précédent 7 de la mise en œuvre de RSAW.
Pour l'expérimentation, nous avons mené les trois tests suivants :

- Tests du système robotique SAM avec AVISO dans un environnement réel : Ces tests ont permis d'expérimenter le robot SAM avec le système AVISO enrichi ,par les fonctions de navigation, de manipulation et de reconnaissance d'objets, dans les centres de réadaptation de Berck sur Mer et Cerbère [96], avec des personnes dépendantes. L'objectif principal de notre expérimentation était d'étudier l'utilisabilité et la pertinence des fonctionnalités de SAM auprès d'une population non spécialiste. Ceci nous a permis de valider l'efficacité des composants logiciels, développés et intégrés dans AVISO au moyen de la mise en place et le test de différent scénarios.

- Tests du système robotique SAM doté de RSAW en laboratoire (CEA-LIST) : Ces tests ont permis d'analyser l'intégration de RSAW dans AVISO avec des scénarios mis en œuvre sur le robot SAM. La version de RSAW utilisée appliquait un raisonnement par analogie basé sur un calcul de similarité syntaxique (Alg .9) [97]. Les résultats obtenus ont montré la faisabilité de rendre un robot conscient de la situation en réduisant l'intervention de l'opérateur de 66,7% avec un seuil de tolérance élevé (de l'ordre de 0,95).

- Tests de l'application RSAW développée : Ces tests ont permis d'évaluer le raisonnement par analogie fondé sur un appariement de graphes mettant en

œuvre une combinaison hybride de techniques structurelles avec un calcul de similarité sémantique (Alg. 6). Pour ce faire et étant donné le manque de Benchmarks pour notre problématique, nous avons décidé d'en créer un pour prouver la fiabilité de notre approche. Ainsi, nous avons construit un jeu de données décrivant 1000 situations (ces données représentent une simulation des informations pouvant être captées par le robot) conformément à la définition de la section 4.2.2.2. L'utilisation de l'échantillon construit a permis de suivre le comportement des algorithmes de notre approche RSAW face aux différentes situations. Les résultats obtenus sont très prometteurs et montrent que l'intervention de l'opérateur est réduite de 66,6% à partir d'un seuil de tolérance raisonnable (de l'ordre de 0,7).

Dans ce chapitre, nous présentons les développements réalisés pour mettre en place le système robotique sur lequel nous avons intégré notre système supportant RSAW. Aussi, nous décrivons les expérimentations et la détermination du seuil de tolérance de l'approche RSAW.

8.2 Préparation du système robotique

La préparation du système robotique a été faite dans le cadre du projet ARMEN. Dans cette section, nous présentons le contexte de travail ainsi que le système robotique SAM (le robot SAM avec le système Aviso).

8.2.1 Contexte de travail

Le projet ANR ARMEN lancé en février 2010 a pris fin en juillet 2013. Afin d'offrir à un robot la capacité de réponde aux attentes des utilisateurs et des aidants (opérateurs), ARMEN a mis l'accent sur les questions suivantes :

- La fiabilité de la mobilité du robot dans un environnement intérieur (indoor),
- La facilité d'utilisation et l'intuitivité du dialogue entre le robot et l'opérateur en utilisant entre autre l'analyse sémantique des images et la conception de divers comportements du robot,
- Le développement de plusieurs fonctions automatiques de manipulation mobile pour aider l'opérateur telles que "trouver un objet perdu", "ramener l'objet à la personne" ou "manipuler l'objet"
- L'expérimentation de SAM avec des patients sous contrôle médical,
- Le développement d'un prototype avec un potentiel de devenir un produit industriel.

8.2.2 SAM, le robot expérimental

SAM, pour "Smart Autonomous Majordomo", est une plateforme robotisée multiservices et de faible encombrement, destinée à recevoir des applications variées dans

les domaines des services, de la communication, de l'éducation, de la santé et de la recherche. La compacité du robot, sa capacité de charge jusqu'à 30 kg, ses capteurs dédiés à la navigation, sa maniabilité et sa facilité d'utilisation permettent d'imaginer un grand nombre d'utilisations, y compris par un public large et non initié à la robotique. Comme le montre la Fig. 8.1, la plateforme mobile est surmontée d'un bras

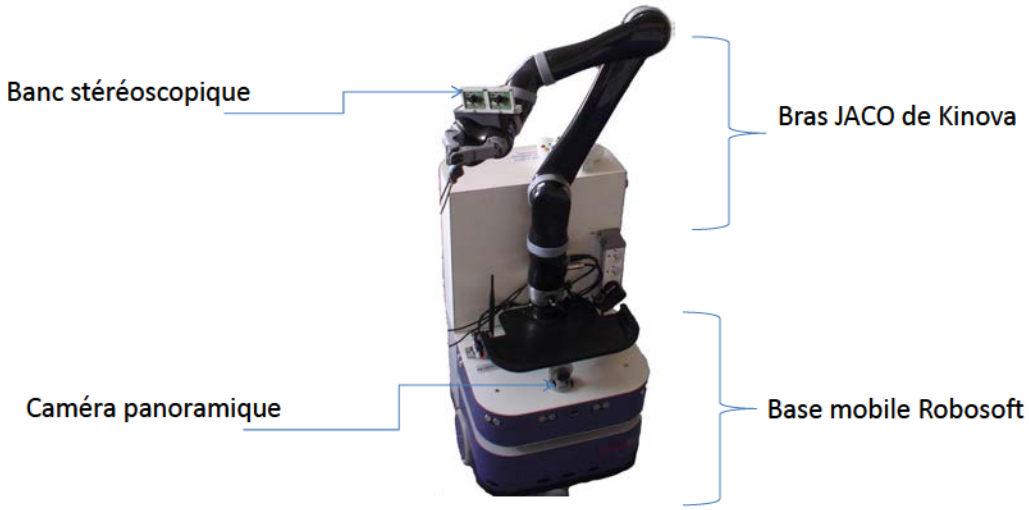


FIGURE 8.1 – Robot SAM

Jaco (société Kinova [148]). Le bras Jaco pèse 5,7 kg, il a une portée de 90 cm, il dispose de 6 degrés de liberté et il peut porter jusqu'à 1,5 kg. Il est alimenté en 24 Volts à partir des batteries du RobuLAB10 [66]. L'extrémité de ce bras est équipée d'une pince possédant trois doigts. Trois webcams sont intégrées, une panoramique et deux posées sur la pince afin de permettre d'avoir une vision stéréoscopique pour l'asservissement visuel sur les objets.

8.2.3 Système Aviso

Les principales fonctionnalités de SAM offertes dans le cadre d'ARMEN sont la navigation, la reconnaissance d'objets et la manipulation se matérialisant par la saisie d'objets.
L'architecture logicielle de SAM repose sur une architecture à couches utilisant DPWS (Device Profile for Web Service [57]) comme outil de communication inter niveau. Les fonctionnalités sont mises en œuvre sur un serveur et l'IHM (le superviseur) est un client connecté aux différentes fonctionnalités comme illustré sur la figure Fig. 8.2) [120]. Dans ce qui suit, nous développons les différents composants du système AVISO et les présentons conformément à l'architecture à niveaux présentée dans le chapitre 4. Rappelons que les niveaux étudiés sont les niveaux de supervision, fonctionnels et délibératifs. Les composants du système Aviso résident plutôt dans les niveaux

fonctionnels et de supervision. Le niveau délibératif est réservé à la compréhension et la génération de plan d'action.

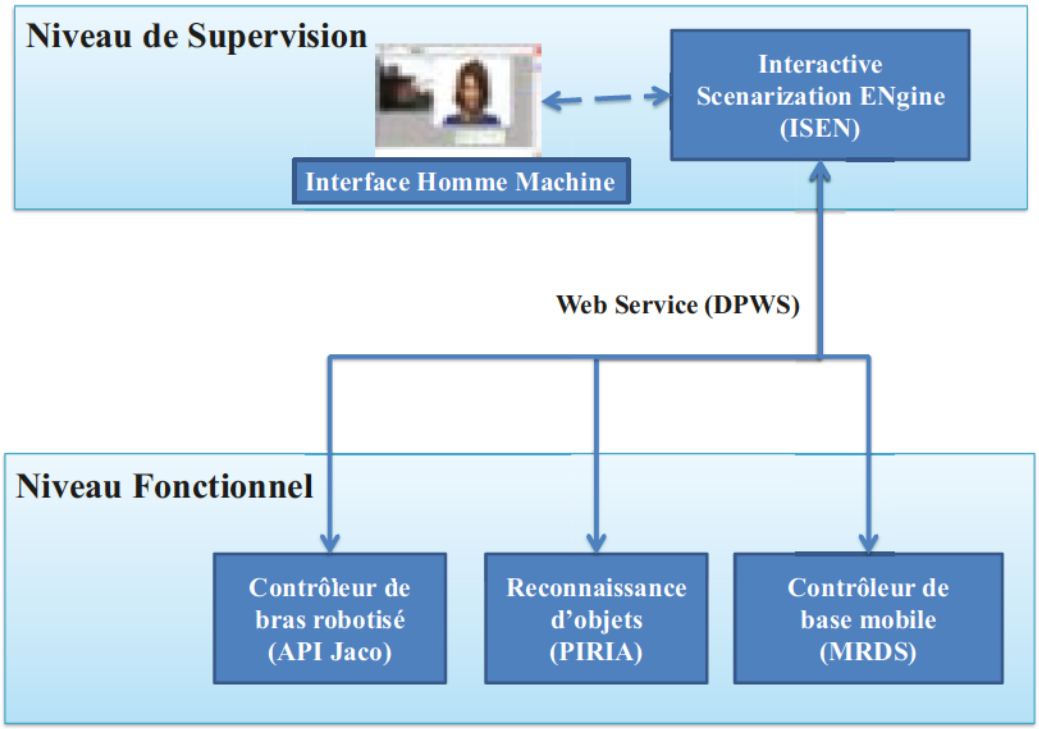


FIGURE 8.2 – Architecture à couches d'AVISO

8.2.3.1 Niveau supervision

Le niveau supervision, représentant la couche intermédiaire, contient l'IHM utilisée pour le dialogue entre l'opérateur et SAM. Ce niveau constitue donc le pilote du robot à travers les commandes lancées par l'opérateur. Afin de rendre l'utilisation de SAM facile, nous avons développé une IHM intuitive, aidés et conseillés par les ergothérapeutes qui ont pris part à ce processus [83]. Cette activité limitée est un facteur important pour l'acceptation du système de la part des personnes en situation de handicap et qui n'ont pas toujours des connaissances en informatique. Pour faciliter l'utilisation de l'IHM, nous avons conçu différents modes d'affichage, donnant accès à différents ensembles de boutons, qui permettent à l'utilisateur d'agir en fonction de la situation (par exemple, la sélection d'un objet dans une scène ou le choix d'un scénario ne peuvent pas être réalisée dans le même mode). L'opérateur peut facilement sélectionner un scénario dans la liste de boutons figurant dans l'IHM, de même un

nombre très limité de clics sont nécessaires pour accomplir une tâche. Le retour visuel des caméras embarquées est affiché pour permettre à l'utilisateur de localiser le robot dans l'appartement ou les objets dans une scène. Les messages codés en couleur sont affichés lors de l'exécution du scénario pour informer l'utilisateur de l'étape actuelle du scénario pour lui permettre de savoir si jamais un problème est survenu et ce qu'il faut faire.
Dans ce niveau de supervision, comme le montre la figure 8.2, l'IHM permet à l'opérateur de sélectionner les tâches à réaliser par le robot. Ces tâches consistent en des comportements exécutés et contrôlés à l'aide du moteur de scénarisation "ISEN" 7.3.2.2, contenu également dans ce niveau. Ce moteur ISEN met en œuvre un mécanisme de programmation évènementielle pour coordonner les tâches réalisées par le robot.

8.2.3.2 Niveau fonctionnel

Le niveau fonctionnel, représentant la couche réactive du système AVISO, contient les composants logiciels permettant le contrôle de l'accomplissement des fonctionnalités de navigation, de manipulation et de reconnaissance d'objets. Ces composants sont autonomes et plug-and-play sur le superviseur. Ainsi, une fonctionnalité liée au robot peut être facilement remplacée par une autre qui l'adapte ou l'améliore. Ceci se traduit par le remplacement du composant logiciel qui implémente la fonctionnalité, comme par exemple le logiciel de reconnaissance d'objets.
Dans la suite nous présentons les différents contrôleurs liés aux fonctionnalités susmentionnées, à savoir le contrôleur du bras robotisé nécessaire pour effectuer les tâches de manipulation, le contrôleur de la base mobile nécessaire pour effectuer les tâches de navigation et le composant de reconnaissance pour la recherche d'objet dans l'environnement.

a Interface avec le contrôleur du bras robotisé

L'interface avec le contrôleur du bras est générique vis à vis du robot utilisé facilitant un changement éventuel du bras robotique. Cette interface intègre les fonctionnalités de manipulation avec le bras (, l'asservissement visuel utilisé pour récupérer automatiquement les objets, saisie d'objets. Pour assurer les manipulations, l'interface fait appel, d'une part, à l'API (Jaco) fournie avec le bras, et d'autre part, à une fonction d'asservissement visuelle (visual servoing) développée au CEA et intégrée à AVISO. Le système de stéréovision est en mesure de calculer les coordonnées de cet objet dans l'espace et de commencer le processus de l'asservissement visuel.
L'opérateur peut également sélectionner à travers l'IHM des objets dans la scène en dessinant un rectangle autour de l'objet qu'il désire sans l'aide du module de reconnaissance d'objets. Le bras peut également être commandé manuellement à travers l'IHM grâce aux boutons directionnels cliquables. Cela permet la recherche d'objets dans la scène par l'utilisateur. L'universalité et l'interopérabilité du contrôleur du

bras sont garanties par sa capacité à travailler en association avec différents services ou par lui-même.

b Interface avec le contrôleur de la base mobile

L'interface avec le contrôleur de la base mobile est générique vis à vis du robot pour faciliter un changement de la partie mobile. Cette interface intègre les fonctionnalités de navigation de la base mobile (planification de trajectoire, évitement d'obstacle, etc...). Nous avons élaboré une communication entre le contrôleur de la base mobile et Microsoft Robotics Developer Studio (Microsoft RDS, MRDS) [41]. MRDS est un environnement Windows pour le contrôle du robot qui gère une grande variété de matériels des robots. MRDS est basé sur Microsoft .NET [256] pour la gestion des tâches parallèles asynchrones qui implique l'utilisation d'échanges de messages et une exécution orientée services. La communication entre le contrôleur de la base mobile et MRDS se fait à travers le service Lokarria [153] mis en œuvre dans MRDS. La localisation du robot se fait en mettant en correspondance des données fournies par un télémètre laser avec une carte 2D de l'environnement enregistrée au préalable (offline). Des noms d'emplacements clés tels que (kitchen, Bedroom) sont associés aux zones d'évolution. Les informations ont accessibles via MRDS.

c Reconnaissance d'objets

Le composant de reconnaissance d'objets développé dans AVISO permet un changement facile des caméras ou un remplacement de la fonction de reconnaissance elle-même. Il assure un contrôle de bas niveau et associe des informations sémantiques aux objets de l'environnement perçus avec les caméras. Il est l'un des éléments clés pour permettre au robot d'agir automatiquement lorsque des objets sont identifiés. Cette fonctionnalité s'appuie sur le logiciel PIRIA (Programme pour l'Indexation et la Recherche d'Images par Affinité) développé au CEA LIST. Il s'agit d'un moteur de recherche basé sur le contenu-Image. Cela signifie que le contenu de l'image (couleur, texture, formes, etc.) est analysé pour trouver des images similaires à l'image requête. Plusieurs descripteurs de couleur, de texture ou de forme caractéristiques existent pour créer plusieurs signatures. Pour le projet ARMEN, nous utilisons une version modifiée de la caractéristique SURF (Speeded Up RobustFeatures). Ce choix est motivé par la vitesse de l'analyse qui permet une utilisation interactive du logiciel.

8.2.4 Résultats des fonctionnalités du système robotique SAM

L'objectif principal de l'étude, que nous avons menée, a été d'analyser l'utilisabilité et la pertinence des fonctionnalités de SAM auprès d'une population non spécialiste. Aussi, nous avons expérimenté le déroulement des scénarios et des fonctions correspondantes afin de nous assurer de l'efficacité du système robotique sur lequel sont mises en œuvre nos contributions dans ce travail de recherche.

Dans ce qui suit, nous présentons les scénarios utilisés lors des expérimentations ainsi que les principaux résultats.

8.2.4.1 Scénarios d'évaluations

Les évaluations du robot SAM, se sont appuyées sur 3 scénarios de saisie d'objets par le robot SAM définis dans la description du projet ARMEN [96].

- Scénario 1 : Saisir un objet visible, situé dans la même pièce que l'opérateur. Aller saisir une canette de soda dans la même pièce à un endroit prédéterminé, et la déposer sur une table placée devant l'opérateur. Dans ce cas, l'objet est dans un espace péricorporel.
- Scénario 2 : Saisir un objet visible dans une autre pièce que l'opérateur. Prendre un objet à terre (la télécommande de TV) situé dans une autre pièce mais visible par l'opérateur et le porter jusqu'à hauteur d'une table (hauteur de 90 cm) et l'y déposer. Dans ce cas, l'objet est dans un espace extracorporel lointain qu'il faut rapprocher du participant.
- Scénario 3 : Saisir un objet non visible de l'opérateur, dans une autre pièce. Aller trouver un livre dans une autre pièce, non visible par l'opérateur et en position inconnue.

8.2.4.2 Évaluations

L'évaluation s'appuie sur une étude bi centrique réalisée dans le cadre du projet ARMEN (ANR TecSan) et sous l'égide de l'association APPROCHE (Association pour la promotion des nouvelles technologies au service des personnes en situation de handicap). Les opérateurs participant à l'étude ont été recrutés au sein de 2 établissements de Médecine Physique et de Réadaptation, situés à Berck sur Mer (62) et à Cerbère (66). Ce sont des personnes adultes ($> 18 ans et < 90 ans$), présentant une tétraplégie fonctionnelle engendrant une importante incapacité de préhension (maladies neuromusculaires, blessés médullaires,...), conservant la capacité à rester assis au minimum 2 heures en fauteuil roulant et à manier une interface de pointage et de validation pour ordinateur (l'interface Homme Machine utilisée n'ayant pas de fonction de défilement).
Nous avons exclu de l'étude les sujets présentant des troubles sensoriels visuels ou auditifs invalidants, des troubles des fonctions supérieures notables et pouvant entraver les capacités d'apprentissage et d'attention, une pathologie psychiatrique instable ou une pathologie aigue intercurrente. Le nombre de cette population (population cible) est de 17 personnes. Une population témoin de personnes valides a été associée. Les critères de non inclusion sont les mêmes que pour la population cible. Le nombre de cette population est de 20 personnes [96].
Les sujets ont bénéficié d'une information orale et écrite et ont donné leur accord par consentement écrit. L'étude a eu l'avis favorable du Comité de Protection des Personnes (CPP) en date du 09/04/2013. Les évaluations ont été faites selon le protocole

d'évaluation décrit dans le livrable du projet ARMEN "protocole d'évaluation". Dans ce protocole, il est noté que les évaluations se déroulent sous forme de 2 à 3 séances par participant, selon le degré de fatigue et de disponibilité :

- Visite V0 : note d'information, critères d'inclusion et signature du consentement
- Visite V1 : scénario d'apprentissage
- Visite V2 : 3 scénarios d'évaluation de l'utilisabilité, questionnaire sous l'échelle de Likert [1].

L'évaluation de l'utilisabilité du robot s'est appuyée sur les 3 scénarios précédemment cités. Chacun des scénarios a été reproduit à 3 reprises. La configuration des espaces utilisés a été identique pour les 2 centres et les distances séparant opérateurs et objets ont été harmonisées. Les réponses au questionnaire de satisfaction par la population cible ont été très positives. En fait, 10 personnes ont trouvé que le mode de pilotage de SAM ainsi que la facilité de son utilisation sont très satisfaisants et 7 personnes les ont trouvés satisfaisants. 16 personnes ont considéré que l'utilisation du système est peu ou pas du tout fatigante. Le degré de confiance du robot est élevé : 70,27% des personnes (76,47% de la population cible et 65% de la population de contrôle) qui ont utilisé le robot affirment qu'ils ont confiance, voire totalement confiance, en SAM. Il a également été constaté que SAM est vu comme une machine à utiliser au quotidien par les personnes en situation de handicap. L'utilité de SAM est ainsi confirmée, les objectifs initiaux du projet ARMEN ont été atteints [83].
Ces résultats ont pu être obtenus grâce aux développements de fonctions permettant une interaction de niveau sémantique avec l'opérateur et grâce à l'utilisation de fonctionnalités permettant de rendre le robot autonome dans un environnement dynamique où le robot ne peut se trouver dans une situation de blocage. Dans ce cadre, lorsque le robot se trouve dans une situation de blocage une intervention de l'opérateur est requise. L'intervention de l'opérateur dans de telles situations est de 100%.
Notre approche RSAW, développée dans le cadre de ces travaux vise à réduire l'intervention de l'opérateur en dotant le robot de plus d'autonomie. L'intégration de RSAW dans SAM fait l'objet de l'évaluation décrite dans la section suivante.

8.3 Intégration de RSAW dans le système robotique

Pour rendre SAM capable d'être conscient de la situation, nous avons intégré RSAW dans l'architecture d'AVISO. Nous avons donc créé, au-dessus du niveau réactif et du niveau de supervision, un troisième niveau correspondant à la couche délibérative qui contiendra, en plus du module de planification (CPT) décrit précédemment, les composants logiciels du système RSAW.
Comme décrit dans la section 4.4 relative à l'intégration de RSAW dans une architecture en couches, les composants de RSAW sont organisés de la manière suivante :

1. http ://fr.wikipedia.org/wiki/%C3%89chelle_de_Likert

- le composant de la détection est ajouté au niveau de supervision et interagit avec ISEN
- les composants automatisant la partie du processus de RSAW relative à la conscience de la situation (les phases de perception, de compréhension et de projection) sont ajoutés au niveau délibératif et interagissent avec le composant CPT.

Comme pour les interactions entre les niveaux fonctionnels et de supervision, la communication entre le niveau délibératif et le niveau de supervision, se fait à travers un service web (DPWS) basé sur le protocole SOAP.
La figure 8.3, montre comment est assurée l'intégration de RSAW (en rose) dans le système AVISO par l'agencement des composants et les échanges entre les couches.
Nous avons mis en œuvre le système « RSAW » supportant notre approche portant le même nom. RSAW est développé en architecture Client / Serveur, la phase de détection de RSAW est développé en C++, elle interagit directement avec le superviseur au sein du système AVISO. A la détection, les données nécessaires sont envoyées à travers un web service au serveur de RSAW qui est codé en Java. Le choix s'est porté sur le langage Java afin d'utiliser JENA qui est une API Java open source pour la manipulation des triplets RDF[157]. Aussi nous avons choisi PDDL4j pour la manipulation des PDDL développé dans [198], SIMMETRICS et SECONDSTRING pour les mesures de similarités développées [42], GRAPHVIZ pour la visualisation des graphes développé par [73]. RSAW est composé de 4 modules :

1. DETECTION, développé en C++ et intégré dans la couche supervision
2. PERCEPTION, développé en JAVA, utilise l'API JENA pour la manipulation des triplets RDF
3. COMPREHENSION, développé en Java, utilise le moteur d'inférence de JENA (pour le raisonnement déductif) et les outils SIMMETRICS et SECONDSTRING pour le calcul des distances de similarités (raisonnement par analogie)
4. PREDICTION, développé en Java, utilise PDDL4j d'une part pour la génération du problème PDDL, et d'autre part pour l'adaptation de la solution à la situation de blocage courante.

8.4 Expérimentations du système robotique avec RSAW

Les expérimentations du système robotique intégrant RSAW ont concerné, d'une part les effets introduits par RSAW sur le comportement du robot dans un contexte dynamique et, d'autre part, la capacité de compréhension mesurée en termes de pourcentage d'intervention de l'opérateur apportée par RSAW au robot lorsqu'il évolue dans son environnement. Les expérimentations se sont déroulées en trois stades.

- Le premier stade d'expérimentation a eu pour objectif de valider la résolution des situations de blocage par RSAW à travers l'application du raisonnement déductif, ainsi illustré dans 6.3.

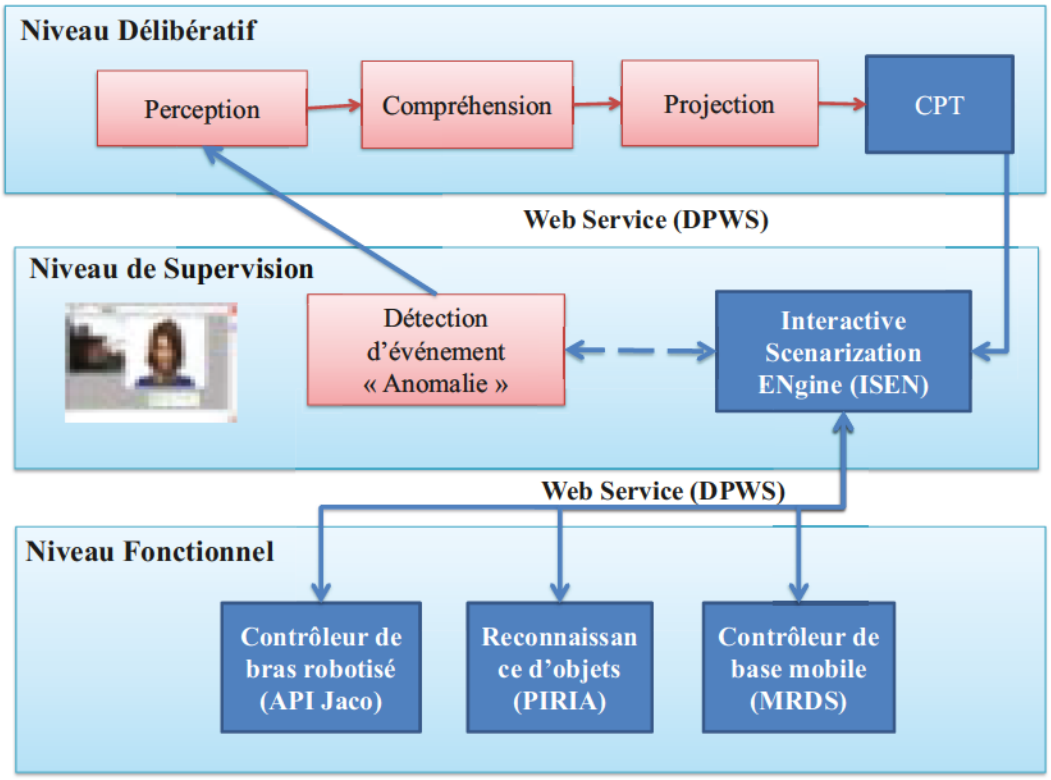


FIGURE 8.3 – Implémentation de RSAW dans le superviseur de SAM

- Le deuxième stade d'expérimentation a eu pour objectif de valider la résolution des situations de blocage par RSAW à travers l'application d'un raisonnement déductif et d'un raisonnement par analogie fondé sur une similarité syntaxique, ainsi illustré dans 6.3.
- Le troisième stade d'expérimentation a eu pour objectif de tester le raisonnement par analogie fondé sur la combinaison d'une mesure de similarité topologique et d'une mesure de similarité sémantique et de déterminer le seuil de tolérance, comme décrit dans la section 6.4. Ce seuil de tolérance est le seuil à partir duquel RSAW peut utiliser la solution de la situation la plus proche de la situation de blocage courante

Dans ce qui suit, nous présentons les expérimentations menées et les évaluations sous-jacentes des différents cas de figures annoncés dans le paragraphe précédent.

8.4.1 Raisonnement déductif pour la résolution des situations de blocage

Les expérimentations ont permis l'évaluation du raisonnement déductif afin de comprendre et de résoudre les situations de blocage. De ce fait, nous avons procédé à la provocation de situations de blocage lorsque le robot est en train d'accomplir un plan d'action préétabli pour répondre au désir de l'opérateur consistant à "Apporter à l'opérateur une canette de coca se trouvant dans la cuisine" (scénario 1 dans les expérimentations précédentes).La canette de coca se trouve sur un plancher (simulation d'une table de la cuisine), comme illustré dans la Fig. 8.4. La Fig. 8.5 montre un extrait d'une image d'une canette de coca à partir de l'Interface Homme Machine (IHM).
Le plan d'action suivi par le robot pour accomplir sa mission a été généré avec le générateur de plan d'action (CPT) à partir de deux fichiers PDDL :

- Le fichier PDDL domaine décrivant les prédicats et les actions utilisables dans le contexte du robot.
- Le fichier PDDL problème décrivant les objets du monde, l'état initial et l'état final pour répondre au désir de l'opérateur.

Les fichiers PDDL (domaine et problème) permettant de générer le plan d'action pour attraper la canette de coca se trouvent dans l'annexe B dans les sections respectives B.3 et B.4.
Afin de tester le raisonnement déductif de l'approche RSAW, nous avons provoqué

FIGURE 8.4 – Cannette de coca dans l'environnement réel

des situations de blocage du robot SAM dans les deux modes de manipulation et de navigation.
Le raisonnement déductif a permis d'interpréter les données perçues afin de construire les situations de blocage sous la forme de graphes RDFS conformément à la section 6.3. Par la suite le graphe RDFS construit est exploité pour créer un nouveau fichier PDDL du problème où les objets de ce fichiers sont les nœuds du graphe de la situation de blocage, l'état initial (devient l'état courant) est composé des triplets RDF représentant la situation de blocage courante décrits dans la syntaxe du PDDL et

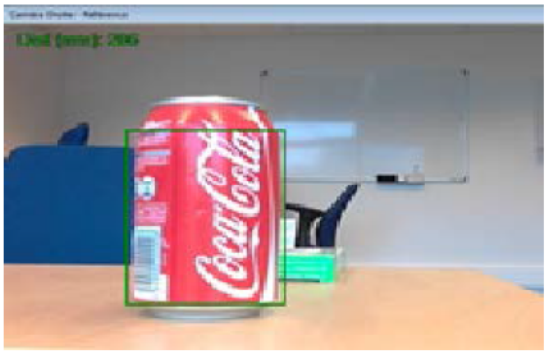


FIGURE 8.5 – Canette de coca captée par la caméra du robot (Image dans l'IHM)

l'état final qui reste le même que celui du problème PDDL du plan d'action initial. Le fichier PDDL du domaine reste inchangé. Les résultats montrent une amélioration dans le comportement de SAM que nous présentons en deux sous sections spécifiques aux deux modes de manipulation et de navigation.

8.4.1.1 Manipulation

En mode de manipulation, nous avons observé le comportement de SAM dans différents scènes, lorsque le robot est équipé ou pas de RSAW.

a Cadre de l'expérimentation

Nous avons effectué des tests avec six situations de blocage que nous avons provoquées et répété chaque test 20 fois sur chacune des situations. Les six situations de blocage sont résumées dans les figures Fig. 8.8, Fig. 8.9 et Fig. 8.10. Nous avons essayé de choisir ces situations en fonctions des situations que le robot peut rencontrer quotidiennement.
L'objectif de SAM dans chaque test était de saisir l'objet désiré par l'opérateur, dans notre cas, la canette de coca. Comme attendu, nous avons noté que lorsque le robot perd la trace de la canette (détection d'évènement d'anomalie (cf. 4.2.3.2)), RSAW prend le relai pour l'aider à prendre une décision appropriée afin de satisfaire le désir de l'opérateur.

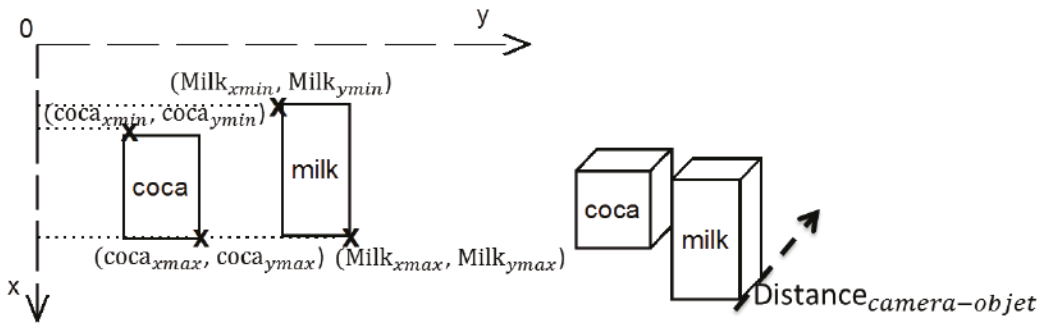
b Utilisation du raisonnement déductif

Lorsque RSAW est déclenché pour résoudre une situation de blocage, un raisonnement déductif est appliqué pour créer un nouveau problème de planification (PDDL problème) en passant par la construction d'un graphe RDFS représentant la situation de blocage. Dans ce paragraphe, nous présentons la construction du graphe RDFS

de la situation de blocage courante puis le passage d'un tel graphe de situation de blocage au problème PDDL qui sera fourni au générateur de plans d'actions pour une nouvelle planification.

Construction du graphe RDFS de la situation de blocage courante Dès que la phase de perception de RSAW associe des valeurs aux concepts (instanciation des concepts) de la situation conformément à ce qui est décrit dans 4.2.2.2, un ensemble de règles d'inférence (annexe A) sont alors appliquées pour établir la déduction de la connaissance symbolique nécessaire à la création du problème de planification. Le logiciel de reconnaissance d'objet PIRIA, utilisé dans AVISO pour le robot SAM, fournit le nom des objets, les boites englobant les objets $(X_{min}, X_{max}, Y_{min}, Y_{max})$ dans l'image ainsi que la distance entre la caméra et l'objet. Par application des règles d'inférence, RSAW déduit la relation spatiale des objets entre eux (devant, derrière, à droite, à gauche,...). Par la suite, RSAW génère le graphe RDFS correspondant à la situation de blocage à partir des connaissances déduites. Le graphe RDFS est construit conformément à l'Alg. 4 présenté dans le chapitre 7, section 7.2.
Nous illustrons ces propos sur la scène de la situation 4 de la Fig. 8.9 qui a été utilisée dans l'exemple de la vidéo [2]. Cet exemple consiste en un blocage causé par un objet se trouvant devant l'objet désiré et que cela n'est pas prévu par le plan d'action initial généré par CPT.
La Fig. 8.6 illustre la déduction des connaissances symboliques, par application des règles d'inférences (b), à partir d'une représentation géométrique de la situation de blocage (a). L'exemple illustré par (a) représente la disposition d'une bouteille de lait (Milk) et de la canette de coca dans l'espace. La partie de gauche de (a) montre la disposition sur le plan de l'image (X, Y) prise par la caméra posée sur l'effecteur du robot avec une origine située en haut à gauche de l'image. La partie de droite de l'exemple (a) montre la distance entre cette caméra et l'objet ce qui représente la dimension Z de l'espace. A partir des données de disposition des objets de la scène et par application des règles d'inférence dont un extrait est illustré dans le tableau représenté en (b), la déduction de la relation sémantique entre les objets est faite. La première ligne de l'exemple des règles (b) montre que si la valeur sur l'axe Z ($distance_{camera-object}$) d'un objet (objet1) est supérieure à celle d'un autre objet (objet1-objet2), cela signifie que l'(objet1) est derrière l'(objet 2), que le robot ne peut pas saisir l'(objet1) car il n'est pas visible (not-clear) alors qu'il peut saisir l'(objet2) qui est devant et donc (clear). La deuxième ligne de l'exemple des règles (b) montre qu'à une $distance_{camera-object}$ égale de deux objets , si Y_{max} de l'(objet1) est inférieur à l'Y_{min} de l'(objet2) alors l'(objet1) se trouve à gauche de l'(objet2). La méthode de raisonnement déductif a permis, grâce aux requêtes SPARQL, la construction des graphes RDFS des situations de blocage. Le graphe RDFS de cette situation de blocage est schématisé sur la Fig. 8.7. Dans le sous paragraphe suivant, nous présentons le passage entre le graphe

2. https ://www.youtube.com/watch ?v=l8qMu6GHTjg.



(a) Représentation géométrique de l'exemple de la situation 4

Prémisses (propositions)	→	Déductions (conclusions)
notEqual(?objet1, ?objet2) lessThan(?Distance$_{camera-objet2}$, ?Distance$_{camera-objet1}$) lessThan(?objet2$_{ymin}$,?objet1$_{ymin}$) lessThan(?objet1$_{ymin}$, ?objet2$_{ymax}$)		(?objet1 IsBehind ?objet2) (?objet1 IsClear not-clear) (?objet2 IsFront ?objet1) (?objet2 IsClear clear)
notEqual(?objet1, ?objet2) equal(?Distance$_{camera-objet1}$, ?Distance$_{camera-objet2}$) lessThan(?objet1$_{ymax}$,?objet2$_{ymin}$)		(?objet2 IsRight ?objet1) (?objet2 IsClear clear) (?objet1 IsLeft ?objet2) (?objet1 IsClear clear)

(b) Règles d'inférences nécessaires pour la déduction des connaissances symbolique à partir de la représentation géométrique

FIGURE 8.6 – Passage de la représentation géométrique à la connaissances symbolique

RDFS et le problème PDDL afin que le générateur de plan d'action (CPT) puisse générer un plan d'action.

Passage d'un graphe RDFS à la génération du problème PDDL Rappelons que le graphe RDFS d'une situation de blocage est un ensemble de triplet RDF (Subject, Predicate, Object)[3]. Les éléments du triplet peuvent être :

- des individus
- des ObjectProperties représentant des relations entre les individus
- des DataProperties représentant les propriétés des individus.

De son côté, un problème PDDL est constitué de trois parties :

3. http ://fr.wikipedia.org/wiki/RDF_Schema

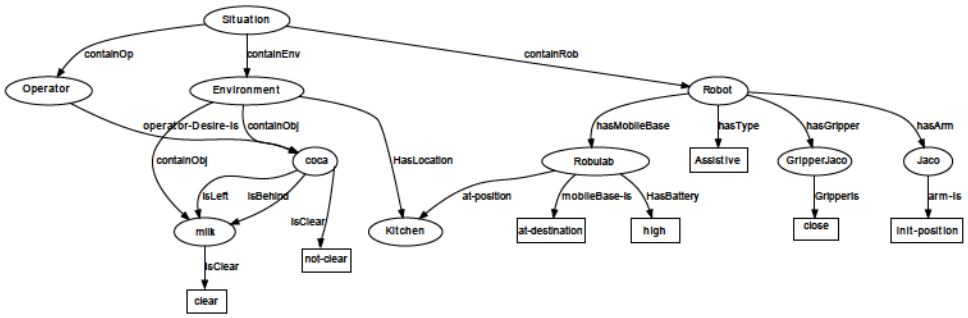


FIGURE 8.7 – Graphe RDFS de la situation de blocage 4 de la Fig. 8.9

- objects (les objets du monde) : Ces objets sont des mots (chaines de caractères)
- init (l'état initial) : défini par un ensemble de prédicats entre les objets conformément à la syntaxe PDDL (exemple (predicat objet1 objet2) montre une relation entre objet1 et objet2). Il décrit l'état initial du système dans le monde.
- goal (l'état final) : défini par un ensemble de prédicats entre les objets conformément à la syntaxe PDDL. Il décrit l'état du système à atteindre

La génération du nouveau problème PDDL est faite à partir du graphe RDFS de la situation de blocage et du problème PDDL initial de la manière suivante :

- Génération des objets du monde (Objects) :
  Les individus du graphe RDFS sont considérés comme objets du monde réel du robot. Ainsi les individus qui n'existent pas déjà dans le fichier du problème PDDL y sont ajoutés aux objets du problème PDDL initial.
- Génération de l'état initial (init) : Les triplets RDF du graphe RDFS de la situation de blocage sont considérés comme l'état initial du nouveau problème PDDL. Ces triplets sont sous la forme de (individu1, ObjectProperty, individu2) ou sous la forme (individu, DataProperty, Valeur). Ainsi la génération de l'état initial est établie de la manière suivante :
  - Les triplets RDF du type (individu1 ObjectProperty individu2) sont adaptés à la syntaxe du PDDL en (ObjectProperty individu1 individu2).
  - Les triplets RDF du type (individu DataProperty Valeur) sont adaptés à la syntaxe du PDDL (Valeur individu)

  Où les objectproperties et les dataproperties doivent être définis dans les prédicats du domaine PDDL.
- Génération de l'état final L'état final du nouveau problème PDDL reste le même que celui du problème PDDL initial.

Enfin, la génération d'un nouveau PDDL problème est assurée. Pour les six situations de blocage provoquées dans l'expérimentation, les problèmes PDDL générés,

à l'issue de ces tests réussis sont montrés dans les figures Fig. 8.8, Fig. 8.9 et Fig. 8.10.

c Résultats

RSAW permet au robot de surmonter certaines limites du niveau réactif tel que l'asservissement visuel pour la manipulation.
L'expérimentation de RSAW pour la manipulation est récapitulée dans la Fig. 8.15. La figure indique le nombre d'essais réussis pour chaque type de situation, lorsque SAM est doté ou non de RSAW. Dans la suite, nous présentons les résultats obtenus. La Fig. 8.8 montre deux exemples d'images de scènes causant un blocage du robot (situation 1 et situation 2) ainsi que les problèmes PDDL générés par le raisonnement déductif à partir de ces situations de blocage rencontrées.

Généralement, dans les situations 1 (a de la Fig 8.8) et 2 (b de la Fig 8.8) , RSAW ne détecte pas les évènements d'anomalie tant que l'asservissement visuel ne perd pas la trajectoire de l'objet. Durant ces tests, lorsque la luminosité dans la scène devient faible, il est arrivé que l'asservissement visuel perde l'objet durant son rapprochement à cause de la disparition des points d'intérêt relatifs à l'objet à saisir. Dans ce cas RSAW détecte un évènement d'anomalie conformément à l'Alg. 2.
La Fig. 8.9 montre deux exemples d'images de scènes causant un blocage du robot (situation 3 et situation 4) ainsi que les problèmes PDDL généré par le raisonnement déductif à partir de ces situations de blocage rencontrées.
Dans les situations 3 (a dans la Fig.8.9) et 4 (b dans la Fig.8.9), RSAW détecte le blocage en générant un plan d'action approprié. Le plan d'action consiste à saisir l'objet qui bloque l'objet désiré, le placer en position non gênante et enfin repositionner le bras pour la saisie de l'objet dégagé.
La Fig. 8.10 montre deux exemples d'images de scènes causant un blocage du robot (situation 5 et situation 6) ainsi que les problèmes PDDL généré par le raisonnement déductif à partir de ces situations de blocage rencontrées.
Pour la cinquième situation (a dans la Fig. 8.10), le robot est en situation de blocage car l'asservissement visuel ne permet pas au robot de s'approcher de l'objet désiré. Cela est dû à l'algorithme d'évitement d'obstacle implémenté d'AVISO [183]. RSAW se déclenche, prend le relai et génère un plan d'action approprié. Le plan d'action consiste à (1) prendre le premier objet d'obstruction pour le placer dans une position non gênante, (2) repositionner le bras pour prendre le deuxième objet d'obstruction et le mettre dans une position non gênante à la manipulation (3) et enfin de saisir l'objet désiré. Les positions non gênantes sont connues à l'avance comme des constantes au niveau du superviseur.

1	 Cause du blocage : L'objet désiré est devant tous les autres objets se trouvant sur la scène.	define (problem PickandPlace) (:domain PickandPlaceDomaine) (:objects kitchen operator-position milk coca gripperjaco jaco robulab) (:init (arm jaco) (init-arm jaco) (not-transport -arm jaco) (gripper gripperjaco) (close gripperjaco) (room kitchen) (room operator-position) (mobile-base robulab) (at-position kitchen robulab) (object coca) (object milk) (isBehind milk coca) (isFront coca milk) (not-clear milk) (clear coca) (at coca kitchen)) (:goal (and (at coca operator-position))))

(a) Situation 1

2	 Cause du blocage : L'objet désiré se trouve entre deux objets.	define (problem PickandPlace) (:domain PickandPlaceDomaine) (:objects kitchen operator-position nesquick milk coca gripperjaco jaco robulab) (:init (arm jaco) (init-arm jaco) (not-transport -arm jaco) (gripper gripperjaco) (close gripperjaco) (room kitchen) (room operator-position) (mobile-base robulab) (at-position kitchen robulab) (object coca) (object nesquick) (object milk) (isRight milk coca) (isLeft coca milk) (isRight coca nesquick) (isLeft nesquick coca) (isRight milk nesquick) (isLeft nesquick milk) (not-clear nesquick) (not-clear milk) (clear coca) (at coca kitchen)) (:goal (and (at coca operator-position))))

(b) Situation 2

FIGURE 8.8 – Exemple de situations (Situation1 ; Situation2)

3	 Cause de blocage : L'objet désiré se trouve derrière et à droite d'un objet bloquant.	define (problem PickandPlace) (:domain PickandPlaceDomaine) (:objects kitchen operator-position badoit coca gripperjaco jaco robulab) (:init (arm jaco) (init-arm jaco) (not-transport -arm jaco) (gripper gripperjaco) (close gripperjaco) (room kitchen) (room operator-position) (mobile-base robulab) (at-position kitchen robulab) (object coca) (object badoit) (isBehind coca badoit) (isFront badoit coca) (isRight coca badoit) (isLeft badoit coca) (clear badoit) (not-clear coca) (at coca kitchen)) (:goal (and (at coca operator-position))))

(a) Situation 3

4	 Cause du blocage : L'objet désiré se trouve derrière et à gauche d'un objet bloquant	define (problem PickandPlace) (:domain PickandPlaceDomaine) (:objects kitchen operator-position milk coca gripperjaco jaco robulab) (:init (arm jaco) (init-arm jaco) (not-transport -arm jaco) (gripper gripperjaco) (close gripperjaco) (room kitchen) (room operator-position) (mobile-base robulab) (at-position kitchen robulab) (object coca) (object milk) (isBehind coca milk) (isFront milk coca) (isLeft coca milk) (isRight milk coca) (clear milk) (not-clear coca) (at coca kitchen)) (:goal (and (at coca operator-position))))

(b) Situation 4

FIGURE 8.9 – Exemple de situations (Situation3 ; Situation4)

5	 L'objet désiré se trouve derrière deux objets bloquants.	define (problem PickandPlace) (:domain PickandPlaceDomaine) (:objects kitchen operator-position milk sch coca gripperjaco jaco robulab) (:init (arm jaco) (init-arm jaco) (not-transport -arm jaco) (gripper gripperjaco) (close gripperjaco) (room kitchen) (room operator-position) (mobile-base robulab) (at-position kitchen robulab) (object coca) (object milk) (object sch) (isBehind coca milk) (isBehind coca sch) (isFront milk coca) (isFront sch coca) (isLeft coca milk) (isRight milk coca) (isRight coca sch) (isLeft sch coca) (clear sch) (clear milk) (not-clear coca) (at coca kitchen)) (:goal (and (at coca operator-position))))

(a) Situation 5

6	 Cause de blocage : L'objet désiré se trouve en dessous d'un objet bloquant	define (problem PickandPlace) (:domain PickandPlaceDomaine) (:objects kitchen operator-position milk coca gripperjaco jaco robulab) (:init (arm jaco) (init-arm jaco) (not-transport -arm jaco) (gripper gripperjaco) (close gripperjaco) (room kitchen) (room operator-position) (mobile-base robulab) (at-position kitchen robulab) (object coca) (object milk) (isUnder coca milk) (isAbove milk coca) (clear milk) (not-clear coca) (at coca kitchen)) (:goal (and (at coca operator-position))))

(b) Situation 6

FIGURE 8.10 – Exemple de situations (Situation5 ; Situation6)

La dernière situation (b dans la Fig 8.10) dans laquelle une bouteille de lait se trouve au-dessus de l'objet à saisir (la canette de coca) est la plus difficile pour le robot. En effet, sans RSAW, le robot ne peut pas saisir l'objet désiré sans faire tomber l'objet situé au-dessus. Avec RSAW, les tests ne sont pas mieux parce que l'algorithme d'asservissement visuel n'échoue pas(conformément à l'Alg. 1). RSAW ne détecte pas une situation de blocage au moment de la saisie de l'objet désiré. Le robot réussit la saisie de la canette de coca mais il fait tomber la bouteille de lait. Parmi les 20 essais effectués sur cette situation de blocage (objet bloquant se trouvant au-dessus de l'objet à saisir), RSAW a réussi à générer un nouveau plan d'action prenant en compte cette situation dans deux essais. En fait, dans ces 20 essais, la détection d'un évènement d'anomalie causé par l'existence d'un objet au-dessus de l'objet à saisir n'est pas évidente dans la mise en œuvre que nous avons effectuée car l'algorithme d'asservissement visuel utilisé ne perd pas l'objet suivi s'il n'existe pas d'objets l'occultant. Cependant, la détection d'événement d'anomalie dans les deux essais qui ont réussi est causée par la mauvaise luminosité dans la cuisine qui conduit à perdre les points d'intérêt utilisé dans le suivi de l'objet à saisir.
Dans les situations 3, 4 et 5, il existe des cas où même lorsque RSAW se déclenche et que l'interprétation de la situation de blocage (construction du graphe RDFS) réussit, la génération des plans d'action avec CPT échoue. Cet échec est essentiellement causé par une mauvaise reconnaissance (la phase de perception) de l'état du monde et apparait par le fait qu'il y a un manque de triplets RDF (individus, dataproperties ou objectproperties). De ce fait, CPT n'arrive pas à générer un plan d'action qui couvre la situation de blocage. Ce qui se traduit techniquement par la mauvaise génération du problème PDDL qui est incomplet ou qui ne permet pas au générateur de plan d'action d'élaborer un plan d'action. Cela est causé soit par une défaillance lors du passage du graphe RDFS de la situation de blocage au problème PDDL, soit par un manque de connaissances dans le graphe RDFS.

8.4.1.2 Navigation

Pendant la navigation, nous avons observé le comportement de SAM à la rencontre d'une porte fermée, lorsqu'il est équipé ou non de RSAW.

a Cadre de l'expérimentation

Nous avons effectué les tests avec la situation de blocage "porte fermée" que nous avons provoquée. Dans cette expérimentation, comme dans celle décrite plus haut, l'objet désiré par l'opérateur (la canette de coca) se trouve à l'emplacement *Kitchen* comme montré sur la Fig. 8.11.
La Fig. 8.11, représente la vue de dessus de la carte de l'environnement du robot. La position de la base mobile représentée par le point vert, la portée de ses capteurs à ultrasons représentée par les lignes bleues ainsi que le trajet, représenté par la ligne rouge, poursuivi par le robot pour atteindre sa destination.

L'anomalie est détectée au moyen de l'Alg. 2 (comme indiqué dans 7.2.2.1), qui contrôle en particulier le trajet suivi et la vitesse du robot qui ne doit pas être en dessous de la vitesse limite inférieure (V_limit = 0,0065 m/s, vitesse en dessous de laquelle le robot s'arrête).
Comme attendu, nous avons noté que lorsque la porte est fermée, le robot tend vers l'arrêt ($V\_mobile < V\_limit$). D'après l'algorithme de détection (Alg. 1), un évènement d'anomalie est déclenché et une situation de blocage est détectée. De ce fait, RSAW prend le relai pour aider le robot à prendre une décision appropriée afin de satisfaire le désir de l'opérateur.

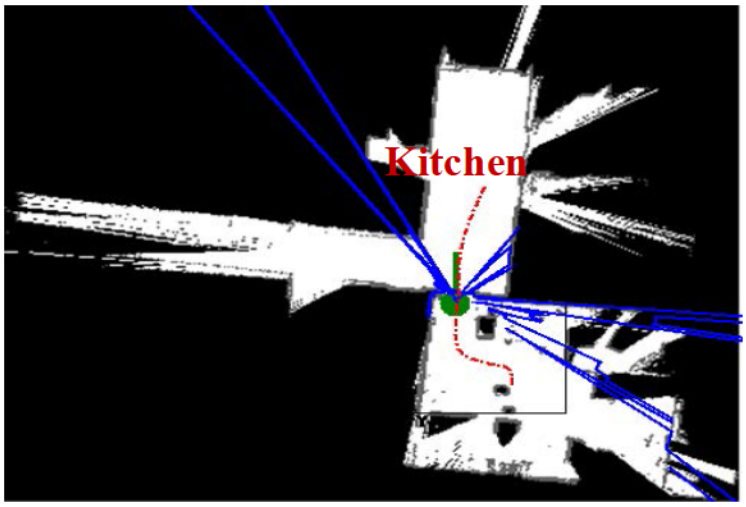


FIGURE 8.11 – Navigation dans SAM

b Utilisation du raisonnement déductif

Lorsque RSAW est déclenché pour résoudre la situation de blocage causée par la porte fermée, l'approche applique un raisonnement déductif pour créer un nouveau problème de planification (problème PDDL).
En effet, dès que RSAW perçoit les valeurs associées aux concepts de la situation conformément à ce qui est décrit dans 4.2.2.2, un ensemble de règles d'inférence (annexe A), sont appliquées pour établir la déduction de la connaissance symbolique nécessaire à la création du problème de planification.
Le contrôleur de la base mobile (MRDS), utilisé dans le robot SAM, fournit la localisation géométrique dans la carte de l'environnement représentée par les coordonnées du point vert dans le repère (X,Y) de la carte 8.11 ainsi que l'orientation du robot dans ce repère. Par application des règles d'inférence, RSAW déduit le symbole de la localisation (par exemple*kitchen, door, operator-position*...) du robot dans l'environnement.

Prémisses (propositions)	Déductions (Conclusions)
(?o1 locationHasX ?x) (?o1 locationHasY ?y) (?o1 locationHasTheta ?z) (?door locationHasX ?x1) (?door locationHasY ?y1) (?door locationHasTheta ?z1) equal(?x, ?x1) equal(?y, ?y1) equal(?z, ?z1)	(?o1 HasLocation ?door)

FIGURE 8.12 – Règle nécessaire pour déduire que la position du robot est la porte

La Fig. 8.12 présente quelques-unes des règles nécessaires à la déduction du symbole "door", à partir d'une représentation de la localisation du robot dans la carte de l'environnement 8.11. La méthode de raisonnement déductif a permis, grâce aux requêtes SPARQL, la construction des graphes RDFS de la situation de blocage dans laquelle la porte est fermée. Ce graphe RDFS est schématisé par la Fig. 8.13.
Pour la situation de blocage provoquée, le fichier problème PDDL est généré (8.14).

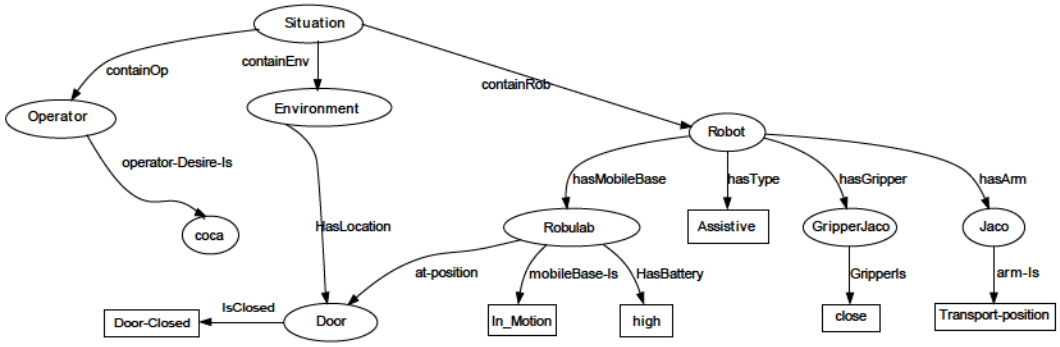


FIGURE 8.13 – Graphe RDFS de la situation de blocage dans la localisation "porte"

c Résultats

Dans cette expérimentation, le robot n'a pas d'autres moyens pour parvenir à la cuisine (*kitchen*) et se met en situation de blocage. Sans RSAW, le robot SAM ne sait pas comment atteindre la cuisine.

```
define (problem PickandPlace)
(:domain PickandPlaceDomaine)
(:objects door kitchen operator-position coca gripperjaco jaco robulab)
        (:init
                (arm jaco)
                (transport-arm jaco)
                (not-init-arm jaco)
                (not-gripper-open)
                (mobile-base robulab)
                (gripper gripperjaco)
                (free gripperjaco)
                (room kitchen)
                (room operator-position)
                (room door)
                (door-closed door)
                (at-position door robulab)
                (object coca)
                (clear coca)
                (at coca kitchen)
        )
(:goal (and
                (at coca operator-position)))
)
```

FIGURE 8.14 – PDDL problème correspondant à la porte fermée

Cependant, équipé de RSAW, SAM détecte ce blocage car sa vitesse devient presque nulle, donc inférieure à la limite fixée (événement d'anomalie), et est capable de le comprendre grâce au raisonnement déductif pour créer le plan d'action approprié. RSAW aide le système de génération de plan d'actions (CPT) à prendre en compte cette situation. La porte fermée fait maintenant partie du problème PDDL. Ainsi, la première tâche dans le nouveau plan est d'ouvrir la porte.
Seulement, ce scénario n'a pu être appliqué sur SAM à cause des risques d'endommager le bras du robot.

8.4.1.3 Discussion

Pour l'analyse de ces expériences, nous définissons le taux de réussite comme étant la moyenne des pourcentages de résolution des situations de blocage par le robot pour atteindre l'objectif indiqué par l'opérateur. D'après ces expérimentations pour évaluer le comportement du robot par application du raisonnement déductif lors des opérations de manipulation et de navigation, il s'avère que le taux de réussite de l'accomplissement du besoin de l'opérateur a été de 37,5% lorsque SAM n'est pas équipé de RSAW. Ce taux est nettement amélioré lorsque le robot est doté de RSAW pour atteindre 64,1% (Fig. 8.15).

Ceci montre que le raisonnement déductif a effectivement permis au robot, à travers

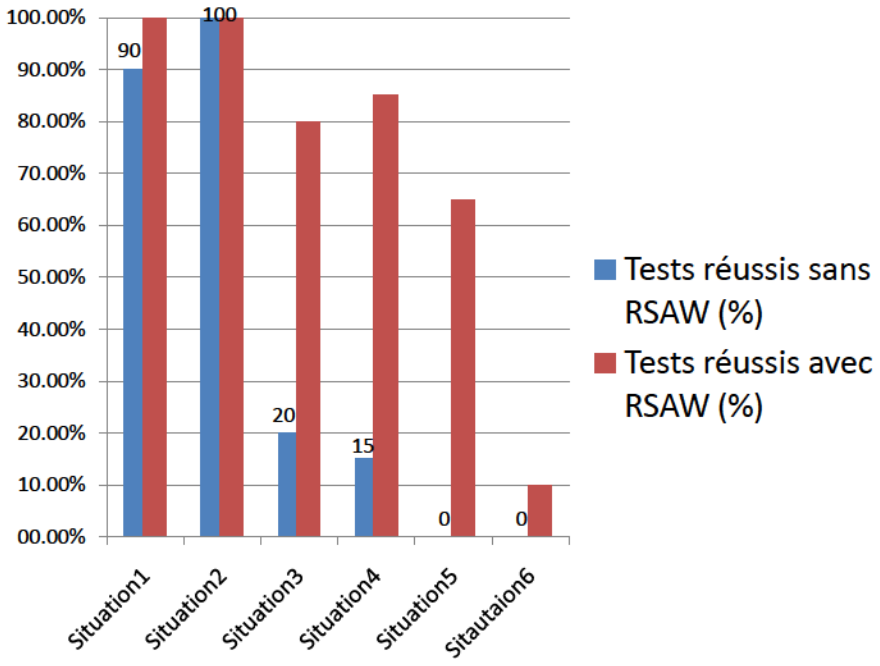


FIGURE 8.15 – Comparaison de la manipulation de SAM avec et sans RSAW

les règles d'inférence, de reconstituer le problème de planification (problème PDDL). Les 35,9% des cas d'échecs sont dus soit au non correspondance du problème à la réalité soit au manque de cohérence du problème (problème PDDL) avec le domaine (domaine PDDL). L'absence de correspondance du problème, posé au générateur de plan d'action (CPT), avec la réalité est due au fait que durant la phase de perception de RSAW (données reçues à partir de PIRIA, MRDS, API Jaco) le système reconnait uniquement une partie du monde (par exemple objets existants sur la scène et non reconnus par le logiciel de reconnaissance d'objets).
Le manque de cohérence entre le problème posé au générateur de plan d'action (CPT) et le domaine d'application (domaine PDDL) est dû au fait de l'utilisation d'une syntaxe, pour l'explicitation du problème, non conforme à la syntaxe utilisée dans le domaine. Nous pouvons citer, à titre d'exemples, le cas d'une classe d'un objet appartenant à la situation courante (au problème PDDL généré) et qui n'existe pas dans le domaine PDDL, ou bien le cas d'une relation entre deux objets, opérationnalisée dans le graphe RDFS de la situation de blocage, qui est insérée dans la description de l'état initial (du problème PDDL) alors qu'elle n'existe pas dans le domaine PDDL. En réalité, ceci se justifie par le fait que :

- Le domaine PDDL est statique car sa modification dynamique est complexe, voire "irréalisable", dans le sens où il ne peut être changé dynamiquement au

cours d'une application (navigation, manipulation, manipulation mobile, télé opération, etc..), et seul l'opérateur « expert du domaine » peut le modifier.
En effet, le domaine PDDL contient des actions qui sont interdépendantes. Cela veut dire qu'il faut s'assurer de la cohérence logique à chaque introduction d'une nouvelle action ou opération. Modifier un domaine PDDL, par exemple le passage d'un domaine « manipulation en robotique » à un domaine « manipulation mobile en robotique » présente le risque d'introduire de l'incohérence dans le jeu d'actions du domaine.
A titre d'exemple : Pour insérer une nouvelle action « *navigation* » dans un domaine PDDL existant et ne concernant que la manipulation, il faut insérer, dans le bloc des pré-conditions et des effets des actions liées à la manipulation, des prédicats tels que : « Le robot ne doit pas bouger lorsque le bras est en train de manipuler un objet et qu'il est dans une position permettant le déplacement du mobile » etc...
Ceci se traduit, comme illustré dans la Fig. 8.16 pour l'action « arm-transport » consistant à assigner le bras robotisé à la position de transport, par l'exigence de s'assurer que la base mobile ne change pas de localisation dans une application de manipulation mobile (colonne 2). Ce qui n'est pas le cas dans une application de manipulation seulement (colonne 1).
Par suite, insérer de nouvelles actions n'est pas sans conséquences sur les actions déjà existantes. Il faut s'assurer de la cohérence des actions entre elles.

(:action arm-transport ; Arm-Transport :parameters (?a) :precondition (and (arm ?a) (not-transport-position ?a)) :effect (and (transport-position ?a) (not (not-transport-position ?a))))	(:action arm-transport ; Arm-Transport :parameters (?a ?r ?b) :precondition (and (arm ?a) (room ?r)(mobile-base ?b)(not(transport-position ?a))(not-transport-position ?a)(at-position ?r ?b)) :effect (and (transport-position ?a) (not (not-transport-position ?a)) (at-position ?r ?b)))

FIGURE 8.16 – Action « arm-transport » en manipulation (colonne 1) et en manipulation mobile (colonne 2)

Le raisonnement par analogie que nous proposons ici ne présente pas cet inconvénient et offre une solution évolutive et dont nous pensons qu'elle représente une nouvelle méthode, une alternative au raisonnement déductif.

- Le problème PDDL est le seul fichier qui peut subir des modifications lorsqu'au cours d'un scénario, une situation de blocage est rencontrée.

Nous avons aussi remarqué, durant ces évaluations, que pour obtenir un plan d'action généré par CPT, il est nécessaire, d'une part, d'avoir un domaine PDDL exhaustif dans les actions et les prédicats, et d'autre part, d'avoir un problème PDDL sans erreurs syntaxiques (sans espace, etc..), dans lequel les états (initial et final) doivent être très bien explicités.
Les résultats obtenus restent optimistes car lorsque l'échantillon des situations devient important, il n'est pas sûr de maintenir le même taux de réussite, du moment que le générateur de plans d'action (CPT) peut ne pas générer de nouveau plan d'action, notamment à cause d'un manque de connaissances.
Lors du passage entre les graphes RDFS et le problème PDDL, l'information sémantique, créée et mémorisée dans les ontologies dans la phase d'interprétation de RSAW est perdue. Pour améliorer le taux de réussite obtenu en profitant de l'expérience du robot, nous avons conclu que deux situations de blocage différentes peuvent avoir le même plan d'action pour leur résolution (par exemple objet bloquant devant l'objet désiré et objet bloquant au-dessus de l'objet désiré). Nous avons alors constaté que les situations de blocage possibles devraient être connues par le robot ainsi que leur résolution générée par CPT. C'est ainsi que nous avons eu recours à une base de connaissances telle que décrite dans le chapitre 5 pour les mémoriser.
Ainsi, les situations de blocage, sous forme de graphes RDFS, sont stockées dans S-Ontology (Base de Connaissances). Cette base de connaissances stocke alors des situations de blocage pouvant avoir la même solution, impliquant ainsi la présence de redondances dans la base de connaissance S-Ontology.
De ce fait, nous avons réfléchi à une solution "fondée sur un raisonnement par analogie", innovante dans ce cadre, afin, d'une part, de minimiser les connaissances (minimiser le stockage des situations du passé), et d'autre part, de raisonner à partir de ces situations (expérience du robot) pour trouver la solution adéquate aux situations de blocage (chapitre 6).
Comme décrit dans le chapitre 6, notre approche pour la compréhension des situations de blocage associe le raisonnement déductif (dont l'expérimentation est décrite dans cette section) et un raisonnement par analogie.
Dans la section suivante, nous présentons les résultats obtenus en associant le raisonnement par analogie au raisonnement déductif. Ces résultats améliorent nettement ceux décrits dans cette section.

8.4.2 Dualité entre raisonnement déductif et par analogie pour la résolution des situations de blocage

Pour améliorer la capacité de résolution des situations de blocage rencontrées par le robot, nous avons procédé à la mise en place d'un raisonnement dual combinant le raisonnement déductif testé plus haut avec un raisonnement par analogie lui permettant de trouver une solution à la situation de blocage courante à partir de son expérience. Comme vu précédemment, le raisonnement déductif génère un graphe

RDFS de la situation de blocage courante. Le raisonnement par analogie se met en œuvre, systématiquement si le raisonnement déductif n'aboutit pas, pour la recherche de la situation la plus proche, à la situation de blocage courante, se trouvant dans la base de connaissances (S-Ontology). Cette recherche est effectuée moyennant une distance de similarité.

8.4.2.1 Utilisation de l'approche

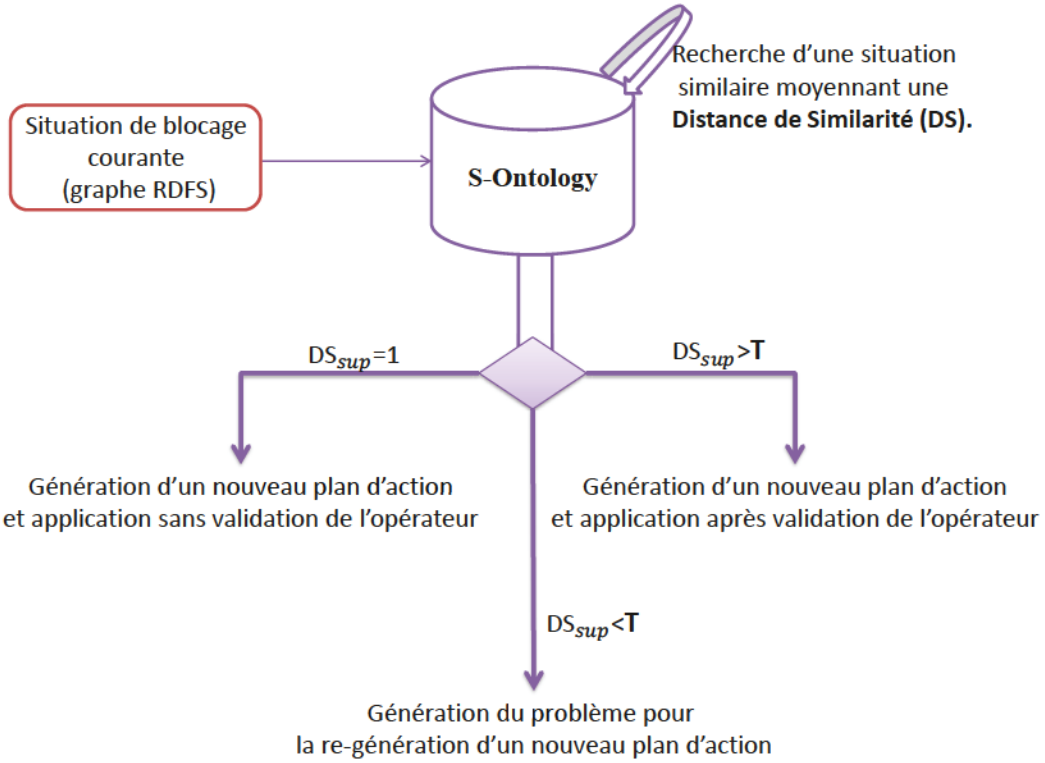


FIGURE 8.17 – Principe du raisonnement par analogie

Lorsque RSAW est déclenché pour résoudre une situation de blocage, nous appliquons un raisonnement déductif suivi d'un raisonnement par analogie pour créer un nouveau plan d'action.
Dès que RSAW perçoit les valeurs associées aux concepts de la situation (c.f. section 4.2.2.2), le raisonnement déductif est appliqué pour construire le graphe RDFS correspondant à la situation de blocage courante. Un raisonnement par analogie est ensuite appliqué Le principe en est rappelé dans la Fig. 8.17. Cette figure illustre la rechercher d'une situation similaire. La base contient l'expérience du robot (S-Ontology), un

calcul de la distance de similarité ($DS \in [0, 1]$) est établi entre le graphe RDFS de la situation de blocage courante et les situations mémorisées (stockées dans S-Ontology). La situation la plus proche est celle qui a la distance de similarité (cf. 6.4.1.1) la plus élevée (DS_{sup} dans la Fig. 8.17) avec la situation de blocage courante.
Selon la valeur de la distance DS_{sup} par rapport à un seuil de tolérance (T), le comportement de RSAW est décrit dans ce qui suit :

- Dans le cas où ($DS_{sup} = 1$) , le nouveau plan d'action est appliqué directement par le robot sans intervention de l'opérateur. Sinon, dans tous les autres cas présentés ci-dessous, l'intervention de l'opérateur est nécessaire.

- Dans le cas où ($DS_{sup} >= T$), ceci implique que la solution de cette situation de blocage sera équivalent à celle d'une situation existante dans S-Ontology. En effet, la solution d'une situation de blocage est contenue dans le fichier référencé par le lien attribué comme valeur de la propriété "HasSolution" de la situation. Le nouveau plan d'action généré, pour résoudre la situation de blocage, est élaboré à partir de la solution de la situation la plus proche et du plan d'action initial et a la même structure que celui généré par CPT. Il comporte tout d'abord les actions de la solution récupérée (celle de la situation similaire adoptée) suivies par les actions inachevées du plan d'action initial.
  Un exemple du fichier solution est montré dans la Fig. 8.18.
  Ce fichier contient les actions à réaliser pour résoudre un blocage causé par une bouteille de lait se trouvant devant l'objet désiré. Les actions sont : (1) mettre le bras à la position de saisie, (2) saisir la bouteille de lait, (3) placer cette bouteille dans une position non gênante (bac d'objets par exemple) (4) remettre le bras dans la position de saisie (5) saisir l'objet désiré (canette) et terminer le plan d'action initial.
  Dans ce cas de figure, le nouveau plan d'action généré doit être validé par l'opérateur.
- Dans le cas où ($DS_{sup} < T$), ceci implique qu'il n'existe pas de situation similaire dont la solution peut être utilisée pour résoudre cette situation de blocage. L'élaboration d'un nouveau problème de planification (PDDL problème) est réalisée. C'est ainsi que le fichier PDDL problème est mis à jour et vérifié avec l'aide de l'opérateur (l'expert du domaine).

Cette approche a été expérimentée en deux phases. D'abord, nous avons évalué le comportement du robot dans un environnement réel avec un raisonnement par analogie mis en œuvre avec un calcul de similarité syntaxique. Ceci a fait objet d'une contribution scientifique [97]. En deuxième lieu, nous avons procédé à l'évaluation du raisonnement par analogie, fondé sur une mesure de similarité hybride, appliquée entre deux graphes RDFS de situations, combinant des techniques d'appariement terminologique et sémantique (chapitre 6).

```
; ParsingTime
; NrActions
; MakeSpan 4
; TotalCost
; MetricValue
; PlanningTechnique
0: (init-grasping jaco) [1]
1: (pick milk jaco kitchen gripper) [1]
2: (place milk jaco safe-position gripper) [1]
3: (init-grasping jaco) [1]
```

FIGURE 8.18 – Exemple de solution de blocage du à une bouteille de lait

Dans la suite, nous présentons les résultats obtenus lors de ces deux expérimentations.

8.4.2.2 Expérimentation sur robot réel

L'expérimentation a été faite avec le robot SAM en se reposant sur un scénario consistant à saisir un objet (canette de coca cf. 8.2.4.1) ne se trouvant pas dans la même pièce que l'opérateur et le ramener à ce dernier.

a Cadre de l'expérimentation

L'application du raisonnement par analogie exige une base de cas et l'adoption d'une distance de similarité. De ce fait, nous avons défini un ensemble de situations de blocage comme expériences passées du robot et adopté la distance syntaxique de Levenshtein pour mesurer le rapprochement entre deux situations. Les situations de blocage créées pour représenter l'expérience du robot sont décrites ci-dessous :

- Une situation de blocage lors de la manipulation causée par la présence non prévue d'un objet devant l'objet désiré :
  - The arm is in manipulation-position ;
  - The gripper is open
  - The Battery of mobile is high ;
  - The mobile is in the kitchen ;
  - The kitchen contains two objects ;
  - The first is in front of the desired object
- Une situation de blocage durant la manipulation causée par la présence non prévue d'un objet à gauche de l'objet désiré :
  - The arm is in manipulation-position
  - The gripper is open
  - The Battery of mobile is high

  - The mobile is in the kitchen
  - The kitchen contains two objects
  - The first is on the left of the desire of operator
- Une situation de blocage durant la manipulation causée par la présence non prévue d'un objet à droite de l'objet désiré :
  - The arm is in manipulation-position
  - The gripper is open
  - The Battery of mobile is high
  - The mobile is in the kitchen
  - The kitchen contains two objects
  - The first is on the right of the desire of operator
- Une situation de blocage durant la navigation causée par la présence non prévue une porte fermée :
  - The arm is in transport-position
  - The gripper is closed
  - The Battery of mobile is high
  - The mobile is in the Door-location
- Une situation de blocage durant la navigation causée par le faible niveau de la batterie :
  - The arm is in init-position
  - The gripper is open
  - The Battery of mobile is low
  - The mobile is in the kitchen
  - The kitchen contains two objects
  - The first is on the right of the desire of operator

Ces situations de blocage ont été mémorisées dans S-Ontology avec la solution correspondante pour les résoudre.
Au cours des évaluations, nous avons provoqué 20 situations de blocage différentes pour étudier le comportement de SAM en appliquant le raisonnement par analogie. Chaque situation de blocage a été provoquée 5 fois pour contrôler la répétabilité de la résolution. Nous avons donc testé 100 situations de blocage.

b Exemple d'utilisation du raisonnement

L'application du raisonnement par analogie moyennant la mesure de similarité de Levenshtein est représenté dans la Fig. 8.19. Cette figure illustre le calcul de similarité entre le graphe RDFS de la situation de blocage courante généré par le raisonnement déductif et le graphe RDFS d'une situation mémorisée dans S-Ontology. Dans ce cadre, on considère le graphe RDFS comme une liste dont les éléments sont les triplets RDF.

Afin de respecter l'ordre d'enrichissement de la liste, nous utilisons un algorithme Depth First Search(DFS) [239]. La Fig. 8.20 illustre un exemple concret du pas-

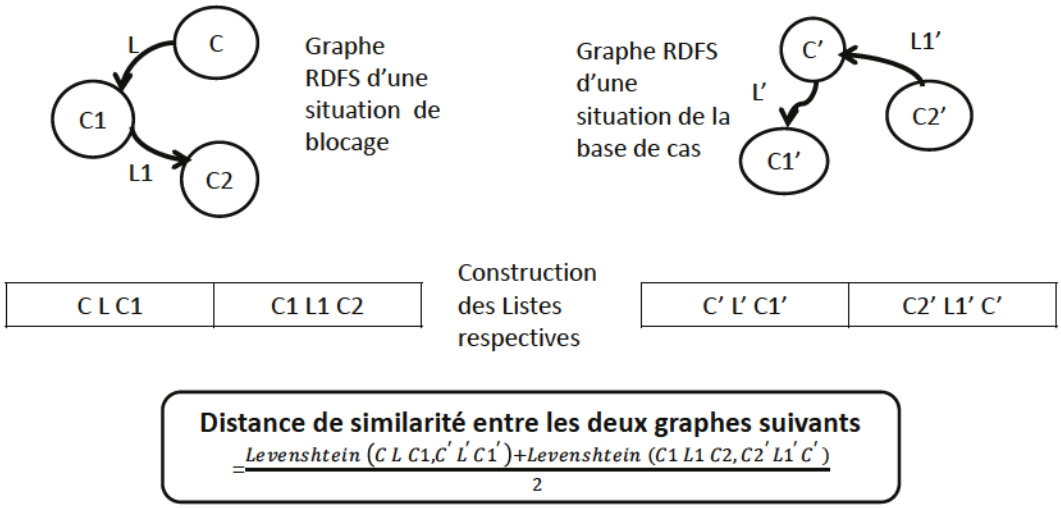


FIGURE 8.19 – Calcul de similarité entre deux graphes RDFS représentant les situations

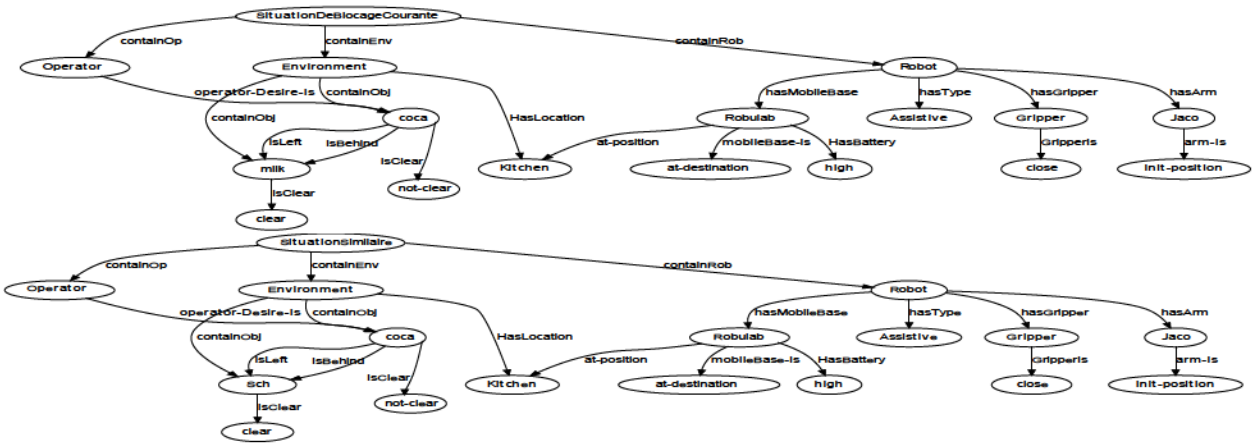
sage des graphes RDFS, d'une situation de blocage courante (issu du raisonnement déductif) et d'une situation se trouvant dans la base de cas (S-Ontology), aux listes représentatives pour élaborer le calcul de la distance de similarité (Alg. 9).

Une fois les listes construites, le calcul de similarité est élaboré. Enfin, un nouveau plan d'action est généré conformément à ce qui est mentionné dans la section précédente. Le seuil de tolérance choisi accepte un seul triplet RDF différent entre les deux situations comparées (une case différente de la liste). Ce qui signifie dans l'exemple traité que ce seuil est de l'ordre de 0.95.

c Résultats

Les résultats obtenus ont permis de classer les situations de blocage en trois ensembles, conformément à Alg. 6. Ils sont présentés ci-dessous :

- 20 situations résolvables, où théoriquement RSAW peut trouver une situation analogue avec une distance de similarité égale à 1 (situation existante dans S-Ontology). Pour ce premier ensemble, SAM réussit dans 16 situations (80%). Les 20% d'échec sont dus à une erreur dans la phase de perception de RSAW lors de l'instanciation des modèles et spécifiquement du modèle de l'environnement (mauvaise luminosité).
- 70 situations résolvables par similarité, où théoriquement RSAW peut trouver une situation analogue avec une distance de similarité $\geq$ au seuil de tolérance (situation n'existe pas dans S-Ontology mais la solution de la situation la plus



(a) Exemple d'un graphe RDFS d'une situation de blocage et d'un graphe RDFS de la situation la plus similaire

Liste de la situation de blocage courante
Operator operator-Desire-Is coca
Environment containObj coca
coca isClear not-clear
coca isBehind milk
milk isClear clear
milk isFront coca
coca isLeft milk
milk isRight coca
Environment containObj milk
Environment HasLocation kitchen
Robot hasMobileBase Robulab
Robulab hasBattery high
Robulab mobileBase-Is at-destination
Robulab at-position kitchen
Robot hasType Assistive
Robot hasGripper Gripper
Gripper GripperIs close
Robot hasArm Jaco
Jaco ArmIs init-position

Liste de la situation la plus similaire à la situation de blocage courante
Operator operator-Desire-Is coca
Environment containObj coca
coca isClear not-clear
coca isBehind Sch
Sch isClear clear
Sch isFront coca
coca isLeft Sch
Sch isRight coca
Environment containObj Sch
Environment HasLocation kitchen
Robot hasMobileBase Robulab
Robulab hasBattery high
Robulab mobileBase-Is at-destination
Robulab at-position kitchen
Robot hasType Assistive
Robot hasGripper Gripper
Gripper GripperIs close
Robot hasArm Jaco
Jaco ArmIs init-position

(b) Listes nécessaires au calcul de la distance de similarité

FIGURE 8.20 – Passage de graphes RDFS vers Listes

proche peut être utilisée pour sa résolution). Pour ce deuxième ensemble, la similarité est calculée et le robot réussit dans 45 cas (64,3 %). Mis à part l'échec de la perception des données provenant des caméras, le calcul donne une similarité acceptable alors que la solution correspondante n'est pas celle qui devrait être adoptée (Faux positif). Ceci confirme la nécessité de validation par l'opérateur.

- 10 situations non résolvables par similarité, où aucune situation proche n'a pu être trouvée et que l'intervention de l'opérateur (expert du domaine) est inévitable.
  Dans ce troisième ensemble, il n'y a pas de similarité avec les situations enregistrées dans S-Ontology. Donc, l'opérateur intervient pour la redéfinition des fichiers PDDL et met en marche le système de génération de plan d'action (CPT) pour générer un nouveau plan d'action. SAM réussit dans 6 cas (60 %). Ce résultat est dû à la difficulté d'obtenir un PDDL cohérent pour générer un plan d'action automatiquement.

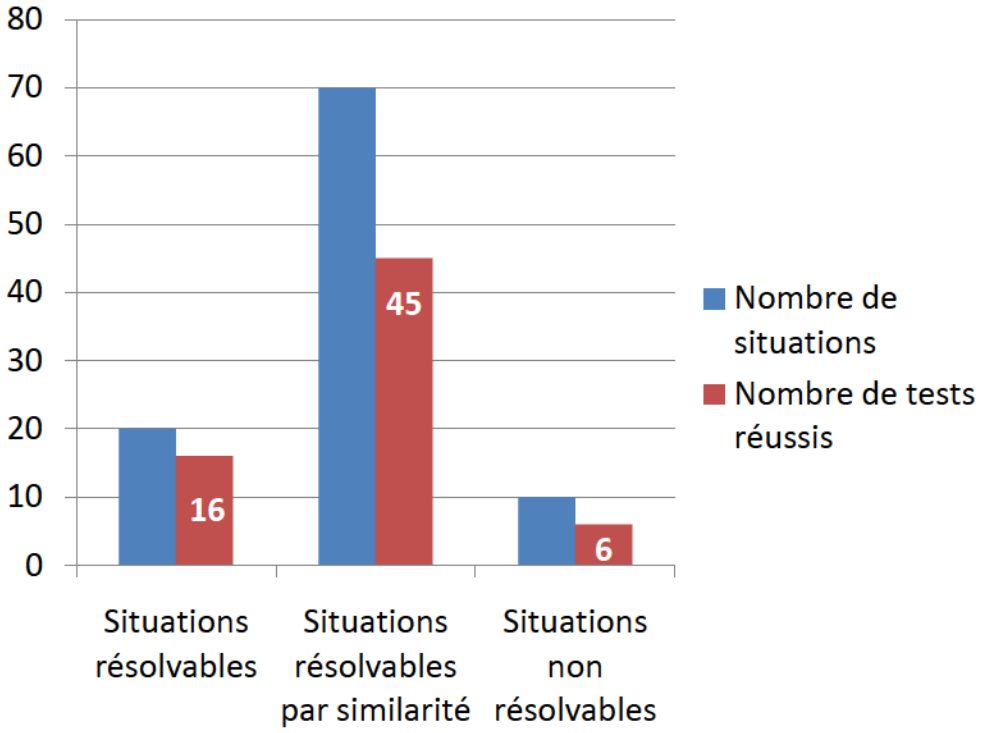


FIGURE 8.21 – Synthèse des résultats obtenus avec la distance de Levenshtein

Alors que SAM non doté de RSAW n'a pas réussi à dépasser les situations de blocage que nous avons choisies pour leur complexité, nous constatons que lorsqu'il est doté de RSAW, il réussit, dans 66,6% des situations, à satisfaire le désir de l'opérateur avec un seuil de tolérance entre situation de blocage et situation similaire assez élevé (de l'ordre 0.95). Nous considérons qu'il s'agit d'un résultat respectable parce que les situations de blocage ne peuvent pas être résolues avec uniquement le raisonnement déductif classique. RSAW a permis au robot de surpasser la majorité des situations de blocage et donc il y a satisfaction de l'opérateur.

d Limitations

La limitation de cette méthode est que la mesure de similarité choisie ne prend pas en compte ni la topologie, ni la sémantique apportée par les graphes représentant les situations. En plus, si le triplet indiquant le désir de l'opérateur n'est pas le même entre la situation de blocage courante et la situation la plus similaire, RSAW génère un plan d'action dans lequel le désir de l'opérateur est erroné. C'est pour cela que l'intervention de l'opérateur est nécessaire dans les cas où la distance de similarité avec la situation la plus proche est inférieure à 0.95.

Les situations de blocage recensées sont relatives à l'environnement dans lequel évolue SAM, c'est-à-dire dans le laboratoire du CEA. Passer à une échelle plus grande, pour trouver le seuil de tolérance choisi dans RSAW, avec des situations nouvelles et aléatoires ne peut se faire dans ce cadre (environnement restreint). C'est la raison pour laquelle nous basons la suite des expérimentations en simulation.

8.4.3 Seuil de tolérance dans RSAW

L'expérimentation simulée de RSAW permet, d'une part, de déterminer le seuil de tolérance à partir duquel l'approche RSAW peut utiliser la solution de la situation la plus proche à la situation de blocage pour la résoudre, et d'autre part, de vérifier et de valider ses algorithmes et notamment le raisonnement par analogie sur un échantillon consistant en des situations de blocage qui représentent les situations de blocage qu'un robot peut rencontrer durant l'exécution d'un plan d'actions.

8.4.3.1 Cadre de l'expérimentation

Pour obtenir des résultats simulés et vu le manque de Benchmark permettant de le faire, nous avons créé notre propre jeu de données pour évaluer le raisonnement par analogie. Nous avons testé la mise en œuvre et l'interprétation des données conduisant à la construction des situations ainsi que la préparation de la prise de décision. Le jeu de données est créé par un algorithme générant aléatoirement des valeurs des concepts 4.2.2.2 caractérisant une situation dans les limites de ce qui peuvent correspondre à la réalité. Nous avons affecté une solution à chaque situation de blocage d'une façon

manuelle.
Ainsi, nous avons construit un jeu de données contenant 1000 situations de blocage représentatives. Par la suite, nous avons procédé, au moyen du raisonnement déductif de RSAW, à l'interprétation de ces données et à la construction des instances de situations correspondantes dans S-Ontology.
Pour tester le raisonnement par analogie conformément au principe schématisé dans la FIg.8.17, nous avons créé 50 situations de blocage comme données de tests.
Dans la suite nous présentons la construction de la base de données contenant les valeurs associées aux concepts des situations de blocage. Puis, l'élaboration de la base des cas est faite. Ensuite une illustration en vidéo du déroulement de la résolution d'une situation de blocage (recherche de la situation la plus similaire à une situation de blocage). Enfin, nous déterminons le seuil de tolérance de la mesure de similarité à partir duquel RSAW génère un plan d'action pour la situation de blocage courante à partir de la solution de la situation la plus similaire.

8.4.3.2 Construction du jeu de données expérimentales

La construction du jeu de données a consisté à créer une table dont les colonnes correspondent aux données relatives aux situations telles que décrites dans la section 4.2 et les lignes contiennent les différentes instances de situations. Ces différentes instances sont générées aléatoirement, tout en assurant les contraintes du monde réel. Nous rappelons que les données relatives aux situations sont :

- Degrés de liberté du bras
- Positions des doigts
- Données sur la base mobile (Batterie du robot, Vitesse et Distance du robot à la destination)
- Localisation du robot
- Noms des objets de la scène
- Boites englobantes des objets
- Données sur l'opérateur (le nom de l'opérateur et le nom de l'objet désiré)

Les instances créées dans les lignes sont en structure JSON [56] parce qu'elle permet de représenter les données sans avoir des restrictions sur le nombre de celles-ci (plusieurs objets, plusieurs boites englobantes par exemple). 1000 instances (lignes) ont été générées dans cette table.

8.4.3.3 Construction de la base de cas S-Ontology

Nous avons opérationnalisé notre algorithme de raisonnement déductif sur le jeu de données décrit précédemment. Pour ce faire et en vue de structurer les données conformément aux modèles de représentation des connaissances de RSAW (cf. 5), nous avons développé un algorithme qui extrait les données de la table et instancie les modèles relatifs au Robot, à l'Environnement et à l'Opérateur. Après cela, la génération, au moyen du raisonnement déductif de RSAW, des graphes RDFS de

chacune des instances du jeu de données, s'est déroulée comme nous l'avons conçue dans la section 6.3. A l'issue de quoi S-Ontology a été construite par 1000 situations de blocage à l'issue de l'interprétation (étape de RSAW) de chaque ligne de la table construit à partir du jeu de données 8.4.3.2. Le contenu de cette ontologie correspond parfaitement aux situations contenues dans le jeu de données créé initialement.
Donc, d'une part, nous avons prouvé l'efficacité du raisonnement déductif puisqu'il

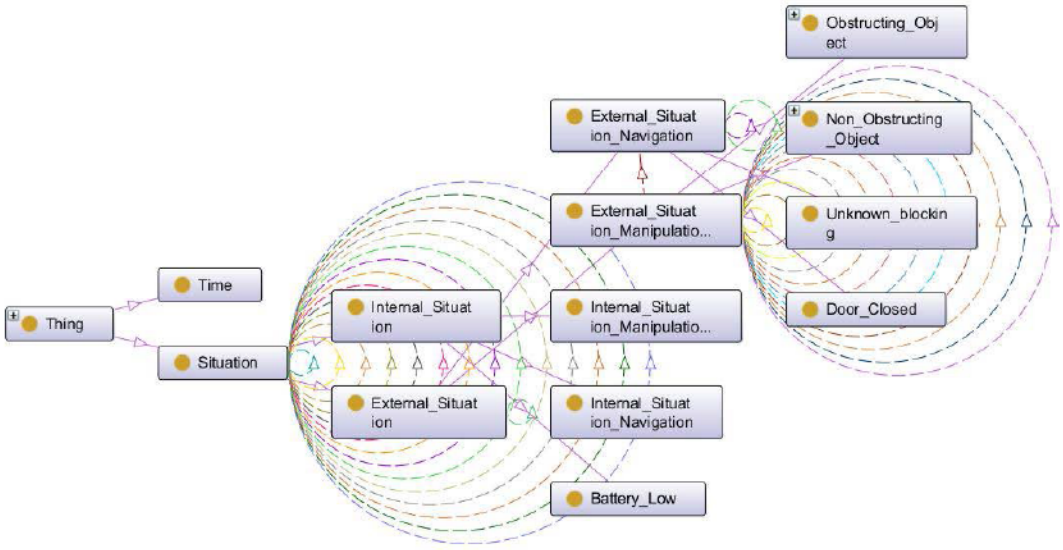


FIGURE 8.22 – Extrait de S-Ontology

a généré dans S-Ontology des situations correctes (correspondante à la réalité), et d'autre part, nous avons obtenu une base de cas (S-Ontology) prête à être exploitée par le raisonnement par analogie choisi dans nos travaux.

8.4.3.4 Illustration

Le déroulement d'un exemple de résolution d'une situation de blocage par le système RSAW que nous avons réalisé est donné dans la vidéo suivante [4] créée pour illustration.
Cette vidéo montre que lors de la détection de l'évènement d'anomalie, l'instanciation des modèles (Robot, Environnement et Opérateur) est établie par le système pour décrire la situation de blocage courante. Ensuite, par application du raisonnement déductif, RSAW construit le graphe RDFS de la situation de blocage courante qui est affiché à l'écran. Dès lors et pour cadrer la recherche des situations de blocage proches, l'algorithme de filtrage est mis en œuvre. Dans l'exemple illustré, RSAW trouve 54

4. https ://youtu.be/YsmJXgKqiPs

situations qui peuvent être similaires à la situation de blocage courante (appartenance à la même topologie). Après RSAW effectue le calcul de similarité comme expliqué dans le chapitre 6. Les différents PaireWiseConnectivity Graph entre la situation de blocage courante et les 54 situations retenues après filtrage (non filtrées) sont affichés à l'écran. Le calcul de la distance de Lin est alors élaboré et la comparaison est faite pour qu'à la fin le graphe de la situation la plus proche s'affiche.

8.4.3.5 Détermination du seuil de tolérance

De la même manière que pour la création des situations instances dans S-Ontology, nous avons créé 50 situations de blocage mais non instanciées au préalable dans S-Ontology afin d'appliquer notre raisonnement par analogie.
Au niveau conceptuel dans le chapitre sur le raisonnement, notre choix dans l'adoption d'une méthode de calcul de similarité a été de combiner deux métriques : une métrique topologique (PCG) et une métrique sémantique (Lin).
Une illustration de la construction du PCG entre les situations de la Fig. 8.20 est montrée dans la Fig. 8.23. Afin d'évaluer notre algorithme de raisonnement par ana-

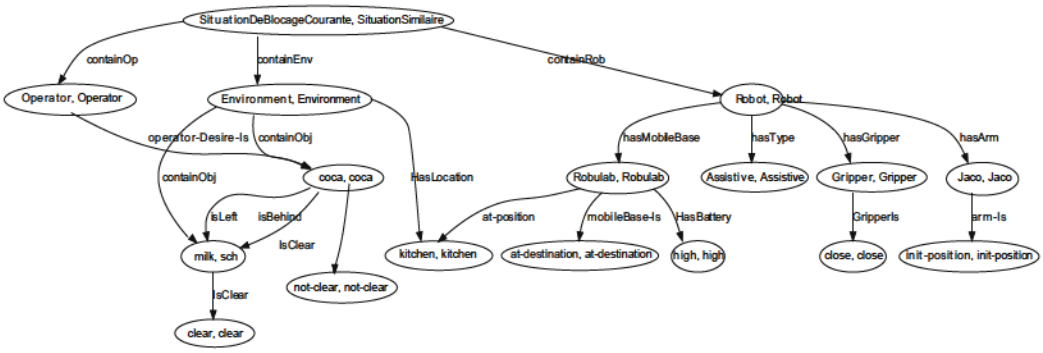


FIGURE 8.23 – Exemple d'un PCG

logie, nous avons procédé à une comparaison de différentes métriques sémantiques à base de contenu informationnel, à savoir Resnick [167] et Jiang and Conrath [241] appliquées à notre démarche. Les métriques sémantiques qui ont été utilisées sont celles de Lin, Resnick, Jiang et Conrath. La technique d'appariement topologique (PCG) adoptée dans la section D.1 pour calculer le graphe commun entre deux situations, a été combinée à ces métriques sémantiques pour les comparer. Pour mener une comparaison objective, nous avons établi un algorithme permettant, pour chaque situation de blocage (SB), de :

1. Chercher, dans la base de cas S-Ontology, la situation la plus semblable SS disposant du plus grand nombre de triplets communs avec SB

2. Calculer le pourcentage d'appartenance, dans SB, des triplets similaires entre SB et la situation retournée. Ce pourcentage est calculé, pour chaque situation similaire fournie par une métrique donnée et par rapport à la même SB, au moyen de la formule suivante :

$$\frac{NombreDeTripletsIdentiques}{NombreDeTripletsTotalDeLaSituationDeBlocage}$$

Le résultat calculé est par la suite comparé à SS pour confirmer notre choix des métriques adoptées ainsi que l'établissement d'un seuil de tolérance permettant d'orienter la prise de décision.
La Fig. 8.24 montre la comparaison du raisonnement par analogie avec les différentes métriques suscitées ainsi que le taux d'appartenance réelle de SB à la base de cas S-Ontology. La courbe bleue représente l'approche de raisonnement par analogie avec le calcul de similarité de Lin adopté dans RSAW pour les 50 situations respectivement avec S-Ontology.
La courbe verte représente l'approche de raisonnement par analogie avec la similarité de Resnick pour les 50 situations en fonction du pourcentage d'appartenance avec la situation la plus semblable de S-Ontology trouvée par cette méthode. La courbe rose représente l'approche de raisonnement par analogie avec la similarité de Jiang et Conrath pour les 50 situations en fonction du pourcentage d'appartenance avec la situation la plus semblable de S-Ontology trouvée par cette méthode.
La courbe en noire est celle du taux d'appartenance réelle des 50 situations courantes avec la vraie situation semblable (SS) dans S-Ontology. Nous remarquons qu'au-delà de 70% (0,7 en similarité) les courbes de Lin et celle du taux d'appartenance réelle sont pratiquement confondues dans 66,6% des cas.
Donc, afin d'assurer la bonne prise de décision, le seuil de tolérance dans la phase de projection est fixé à 0,7.
Ces résultats ont permis d'une part, de nous assurer que l'approche d'appariement de graphes choisie assure la fiabilité de RSAW et d'autre part, ils nous ont permis de fixer un seuil de tolérance permettant de réduire l'intervention de l'opérateur à un taux de 66,6%.

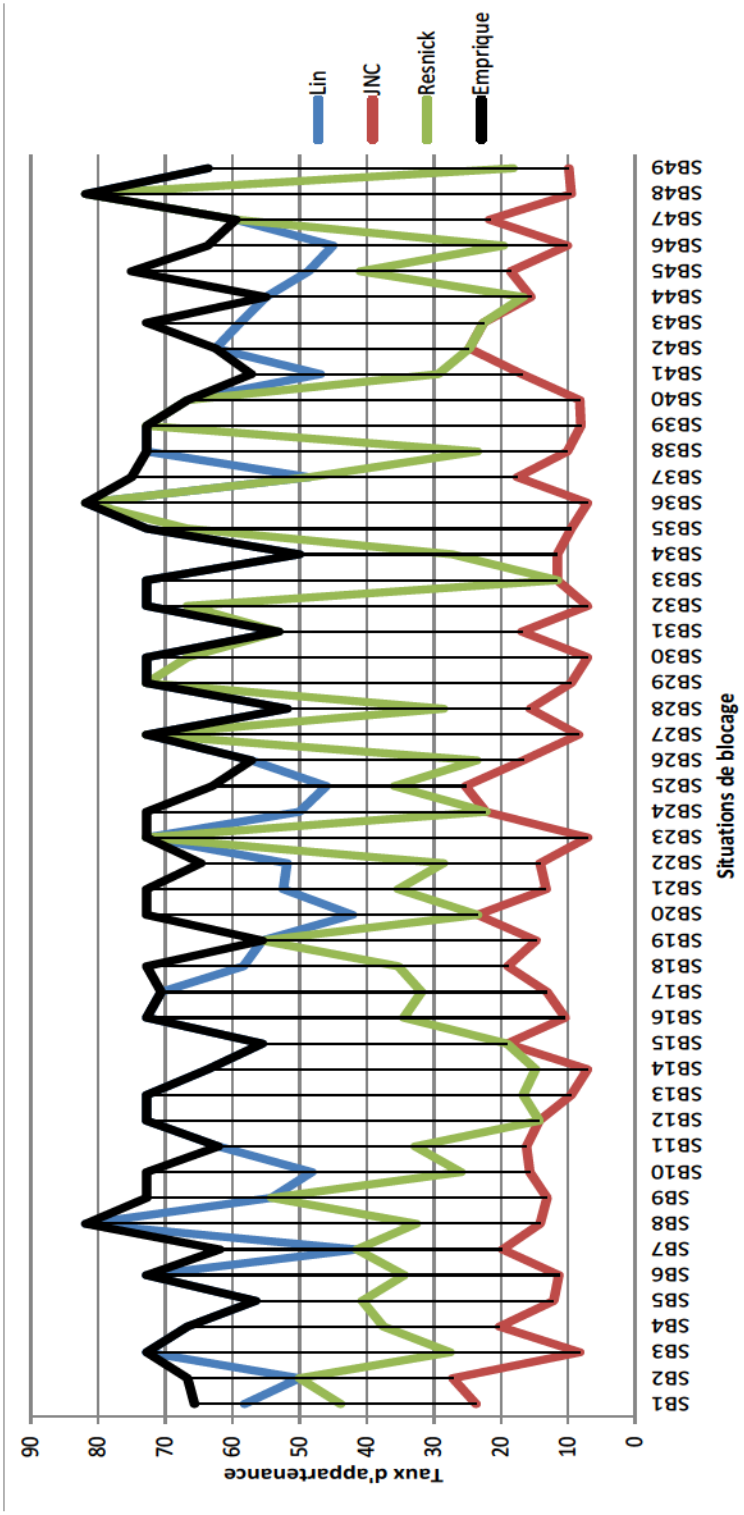


FIGURE 8.24 – Évaluation du raisonnement par analogie

8.5 Conclusion

Les expérimentations menées de ce chapitre se sont déroulées en deux étapes en environnement réel puis en laboratoire. La première étape s'est déroulée dans le cadre du projet ANR ARMEN supervisé par le CEA-LIST et ont exigé la préparation d'un système robotique sur lequel nous avons intégré notre contribution. Ce système robotique (le robot SAM avec le système AVISO) a été testé dans un environnement réel, dans les centres de réadaptation de Berck sur Mer et Cerbère, avec des personnes dépendantes sans RSAW. Ceci a permis de valider l'efficacité des composants logiciels, développés et intégrés dans AVISO au moyen de la mise en place et le test de différent scénarios.
La deuxième série d'expérimentations a été effectuée en laboratoire et a concerné l'évaluation de l'intégration de RSAW dans AVISO. En premier lieu, des scénarios d'évaluation ont été mis en œuvre et préétablis sur SAM avec un raisonnement par analogie basé sur un calcul de similarité syntaxique. Les résultats obtenus ont montré la faisabilité de rendre un robot conscient de la situation en réduisant l'intervention de l'opérateur de 66,7% avec un seuil de tolérance élevé (de l'ordre de 0,95).
En second lieu, RSAW à base de raisonnement par analogie fondé sur l'appariement de graphes mettant en œuvre une combinaison hybride de techniques structurelles avec un calcul de similarité sémantique, a été testé en simulation sur une base de données, développée à cet effet, contenant 1000 situations de blocage. Ces expérimentations ont permis de suivre le comportement des algorithmes de RSAW face aux différentes situations. Les résultats obtenus montrent que l'intervention de l'opérateur est réduite de 66,6% avec un seuil de tolérance moindre de l'ordre de 0,7.
Les expérimentations menées et décrites dans ce chapitre montrent qu'il est possible d'améliorer l'autonomie d'un robot en le rendant conscient de la situation. Les résultats obtenus en appliquant un raisonnement par analogie fondé sur un appariement de graphes sont très prometteurs.

Quatrième partie

CONCLUSION GÉNÉRALE

Chapitre 9

CONCLUSION ET PERSPECTIVES

La problématique abordée dans notre travail a porté sur la façon de rendre un robot sensible au contexte afin d'accomplir le désir de son opérateur. En particulier, nous nous sommes intéressés à la question suivante : comment rendre un robot capable de comprendre, s'adapter et réagir face au changement dynamique du contexte en vue de satisfaire l'objectif fixé par l'opérateur ?
Pour répondre à cette problématique, une étude des systèmes robotiques et de l'autonomie des robots a été menée. Ainsi, nous avons mis en évidence que :

1. pour être autonome, un robot doit être doté de fonctions délibératives le rendant capable de comprendre.
2. pour augmenter la capacité de compréhension, ces fonctions délibératives devraient reposer sur une représentation de connaissances pour la modélisation du contexte -défini par le robot, son environnement et son opérateur- qui favorise l'application d'un raisonnement réduisant l'intervention de l'opérateur.

Ces besoins sont les verrous scientifiques auxquels nous avons cherché à répondre.

9.1 Contributions

L'étude et l'analyse de l'état de l'art relatif aux avancées dans les domaines de l'intelligence artificielle et de la robotique autonome, ont permis de constater que les travaux cherchant à répondre à l'amélioration de la capacité de compréhension intègrent seulement certaines des fonctions délibératives et que ceux cherchant à représenter des connaissances proposent des représentations de connaissances spécifiques à leur contexte de travail et donc ne sont pas réutilisables par un système robotique différent. De plus, les travaux cherchant à munir un robot de raisonnement utilisent généralement des raisonnements symboliques, géométriques ou une combinaison des deux négligeant parfois les aspects sémantiques.
L'ensemble de ces constats a orienté nos choix dans les travaux menés dans le cadre de cette thèse. Principalement, pour améliorer l'autonomie du robot et réduire l'intervention de l'opérateur, nous avons misé sur la manière de rendre un robot conscient

de la situation dans laquelle il se trouve en lui permettant de reconstruire l'état courant et de produire un nouveau plan, non seulement en planifiant (avec un générateur de plan d'action), mais aussi en appliquant un raisonnement fondé sur l'exploitation de son expérience passée. C'est ainsi que nous avons proposé notre approche RSAW, générique et facilement intégrable dans un système robotique, inspirée de la notion de conscience de la situation ou Situation Awareness (SA) reconnue comme un élément clé des processus cognitifs en situation dynamique. Plus précisément, en intégrant dans un même cadre les fonctions délibératives assurant la capacité de compréhension, RSAW permet à un robot de comprendre les situations de blocage et réagir en générant le comportement adéquat pour les surpasser et satisfaire le désir de l'opérateur. L'approche RSAW est basée sur un processus à quatre phases :

1. la **détection** des évènements provoquant une anomalie,
2. la **perception** des données capteurs du robot et la reconnaissance de la situation de blocage courante,
3. la **compréhension de la situation de blocage** en appliquant un raisonnement déductif pour l'interprétation des données perçues et un raisonnement par analogie pour préparer la prise de décision,
4. la **projection** où sera prise la décision sur le comportement adéquat à prendre en compte pour surpasser la situation de blocage.

Dans ce processus, la seule intervention de l'opérateur se fait dans la prise de décision. La représentation des connaissances adoptée dans RSAW est basée sur des ontologies, représentées par des graphes RDFS, dont l'organisation est conçue conformément à la méthodologie de l'Ingénierie Dirigée par les Modèles. Le raisonnement déductif, utilisé pour interpréter les données perçues et construire la situation de blocage courante, est mis en œuvre à l'aide des règles rendues opérationnelles avec un moteur d'inférence. Pour la préparation de la prise de décision, un raisonnement par analogie est appliqué pour trouver la situation la plus semblable à la situation de blocage courante. L'appariement de graphes est à la base de ce raisonnement par analogie. Nous avons fondé ce dernier sur une combinaison hybride de techniques d'appariement structurelle et sémantique.
L'approche RSAW ainsi élaborée a été implémentée et intégrée au système AVISO mis en œuvre pour le robot SAM par le CEA. L'architecture utilisée est basée sur les services web et se présente en couches (fonctionnelle, de supervision et délibérative). La préparation de ce système robotique (SAM avec AVISO) a été faite dans le cadre du projet ANR ARMEN pour servir de plateforme d'expérimentations.
Les expérimentations se sont déroulées en trois stades correspondant chacun à un objectif précis. Les premières expérimentations du système robotique préparé se sont déroulées dans un environnement réel. Elles ont permis de confirmer l'amélioration du comportement du robot SAM doté de RSAW et de constater que cette dernière est très sensible à l'erreur. Les secondes expérimentations effectuées au CEA ont permis de tester l'apport de RSAW avec un raisonnement par analogie fondé sur une

technique de similarité syntaxique (similarité de Levenshtein). Elles ont permis de constater l'amélioration du comportement de SAM avec une réduction de l'intervention de l'opérateur. Toutefois, les analogies trouvées sont en général correctes (pas d'analogies erronées) car elles sont déterminées avec un seuil au-dessus duquel RSAW utilise la solution de la situation similaire dans la résolution de la situation de blocage courante (seuil de tolérance) de 0,95.
Enfin, RSAW avec un raisonnement fondé sur la combinaison hybride de techniques d'appariement structurelle (PCG) et sémantique (distance de Lin) a été évalué sur un jeu de tests (base de données de 1000 situations) élaboré à cet effet. Cette méthode a permis de diminuer le seuil de tolérance à 0,7.
Les résultats obtenus sont concluants car ils ont permis de montrer qu'en intégrant la capacité de compréhension, RSAW minimise effectivement l'intervention de l'opérateur.

9.2 Perspectives

Le travail élaboré dans cette thèse présente une recherche d'actualité, très riche et très active dans le sens où la mouvance des technologies innovantes favorise l'automatisation des services quotidiens dans la nouvelle société. C'est d'autant plus vrai dans des domaines critiques et sensibles tels que la défense, l'industrie ou la santé imposant le recours aux robots de service pour assurer des tâches délicates ou d'assistance. L'intégration de la capacité de compréhension dans les robots de service vise à épargner l'opérateur de tâches pénibles ou répétitives.
Dans ce contexte, RSAW se présente comme une nouvelle technologie utilisable dans les systèmes robotiques pour minimiser l'intervention de l'opérateur, en leur offrant plus d'autonomie grâce à l'intégration de la compréhension dynamique du contexte. Certes, comme pour tout travail de recherche, plusieurs perspectives sont envisageables pour RSAW.

9.2.1 Privilégier les modèles

Dans l'état actuel des choses, RSAW considère que les trois modèles (Robot, Environnement et Opérateur) qui interviennent dans la construction d'une situation de blocage ont le même niveau d'importance, et que, par suite, le raisonnement par analogie utilisé s'appuie sur une pondération égale de tous les triplets RDFS intervenant dans le calcul de similarité. Or, dans certaines applications, ces modèles ne devraient pas avoir obligatoirement la même considération. A titre d'exemple, dans une application de co-manipulation, le robot devrait soutenir l'opérateur dans ses actions et par conséquent, le modèle de l'opérateur sera le modèle privilégié par rapport au modèle de l'environnement et celui du robot.
Il est donc possible d'envisager d'adapter le raisonnement dans RSAW selon l'importance du domaine par l'utilisation de pondérations sur les modèles dans le calcul de similarité, en vue de sélectionner la situation qui s'adapte le mieux à l'application

concernée par le robot.

9.2.2 Amélioration de la fiabilité des solutions proposées

L'état actuel de RSAW permet de réduire l'intervention de l'opérateur dans 66,6% des cas. Pour aller plus loin et en vue d'augmenter encore ce pourcentage, on pourrait envisager d'améliorer la fiabilité des solutions proposées par le robot. Ceci peut être envisagé en donnant au système robotique la possibilité soit de préparer la prise de décision sur un ensemble de situations similaires de ses expériences passées plutôt qu'une seule situation comme c'est le cas actuellement, soit de prendre en compte la contextualisation du raisonnement, par exemple les relations temporelles et/ou cartographiques entre les situations. De même, il serait intéressant de prendre en compte les données erronées ou ou non collectées par les capteurs du robot. Dans la continuité de cette thèse une étude pourrait mettre l'accent sur la vérification et l'ajustement des situations de blocage.
Des méthodes d'optimisation du temps de calcul dans l'application du raisonnement par analogie pour la recherche dans la base de des situations les doivent être mise en place afin d'assurer une réponse rapide à l'opérateur.

9.2.3 Prise en compte de l'intention de l'opérateur en ligne

Dans nos travaux, le modèle de l'opérateur a été conçu sur la base de son intention en prenant en compte uniquement la mission à accomplir par le robot. Cependant, l'intention de l'opérateur pourrait changer lors de l'exécution de la mission courante. Une des perspectives de ce travail serait de prendre en compte l'intention de l'opérateur en lui permettant de modifier la mission du robot en ligne. Cela peut être introduit par l'architecture Belief Desire Intention (BDI) dans les systèmes robotiques permettant de définir différentes stratégies de décision en respectant les intentions de l'opérateur et la modélisation d'un large panel d'opérateurs [61].

9.2.4 Mise en œuvre de RSAW dans une architecture Multi-Agents

Dans nos travaux, nous avons mis en œuvre et expérimenté notre approche sur un robot présentant une architecture en couches (centralisée). Une suite possible des travaux de recherche de RSAW serait son adaptation et son intégration dans une architecture distribuée (Multi-Agents) collaborative faisant intervenir différents acteurs (différents robots, opérateurs, capteurs, ...). Cette perspective de recherche mettra en évidence le problème de la prise en compte des différents contextes des acteurs intervenants (intelligence distribuée).

9.2.5 Interaction de RSAW avec les nouvelles technologies

La nouvelle génération de l'industrie, l'industrie 4.0 [80] issue d'une révolution du processus industriel et utilisant les nouvelles technologies innovantes, vise à réorganiser complètement le mode de production avec les outils existants et donnant une plus grande importance au réseau internet. Cette industrie est souvent désigner comme l'usine connectée, la cyber usine ou encore l'usine du futur. En fait, cette usine connectée est caractérisée par une fusion entre le numérique, l'Internet et les usines. Chaque ensemble de chaines de production et d'approvisionnement est associé à des outils et des postes de travail communiquant en permanence grâce à Internet et aux réseaux virtuels. Les machines, les systèmes et les produits échangent de l'information, entre eux ainsi qu'avec l'extérieur.
Dans ces futures générations, les automates et les robots ayant été au cœur de la troisième révolution industrielle du 20ème siècle en proposant d'automatiser la production, doivent évoluer pour s'aligner avec les technologies émergentes telles que le cloud computing, pour être au cœur de l'usine connectée. Pour ce faire, les robots doivent atteindre un certain degré d'autonomie leur donnant la capacité d'interagir avec ces technologies pour augmenter la productivité des chaines de production, par exemple. A cet effet, RSAW, facilement intégrable et développée selon une architecture web service, se prête à offrir un cadre de base fondé sur la sémantique communément utilisée dans ces nouvelles technologies. De ce fait, son avenir est prometteur.

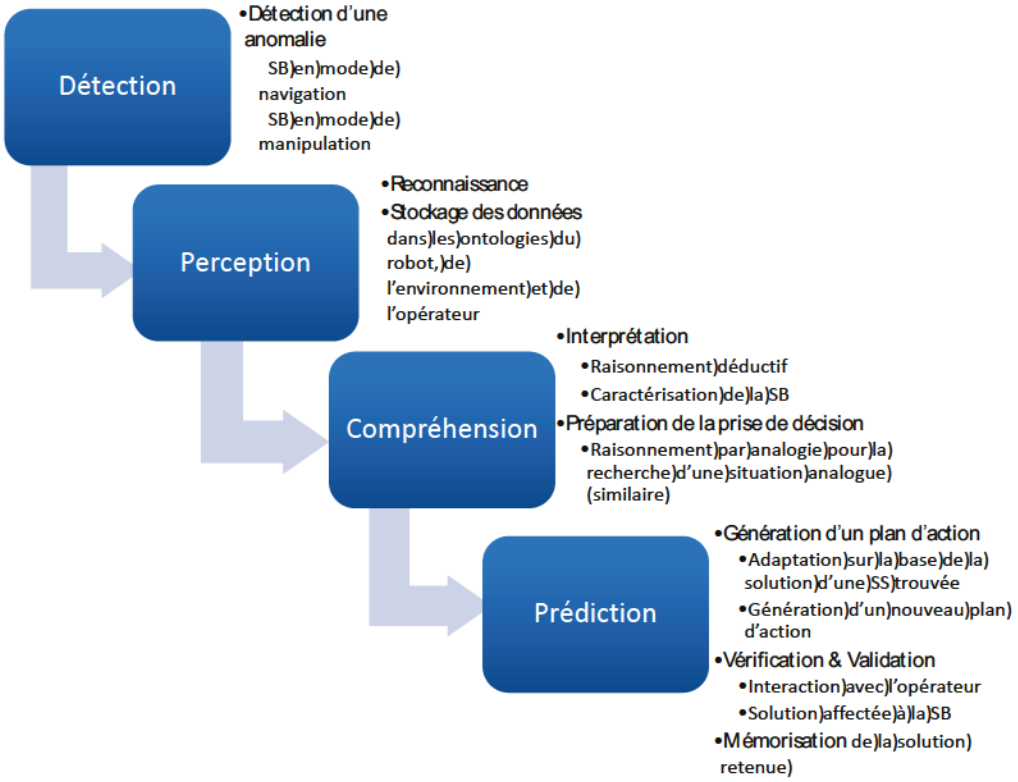


FIGURE 9.1 – Approche de Robot Situation AWareness

Bibliographie

[1] Gregory D Abowd, Anind K Dey, Peter J Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In *Handheld and ubiquitous computing*, pages 304–307. Springer, 1999.

[2] Projet Accord. La démarche mda. *Projet RNTL ACCORD*, 2002.

[3] Julie A Adams. Unmanned vehicle situation awareness : A path forward. In *Human systems integration symposium*, pages 31–89. Citeseer, 2007.

[4] Rachid Alami, Raja Chatila, Sara Fleury, Malik Ghallab, and Félix Ingrand. An architecture for autonomy. *The International Journal of Robotics Research*, 17(4) :315–337, 1998.

[5] Rachid Alami, Aurélie Clodic, Raja Chatila, and Séverin Lemaignan. Reasoning about humans and its use in a cognitive control architecture for a collaborative robot. In *Cognitive Architectures for Human-Robot Interaction Workshop@ HRI" 14*, number EPFL-TALK-196443, 2014.

[6] Éric Auriol. *Intégration d'approches symboliques pour le raisonnement à partir d'exemples. L'induction et le raisonnement par cas dans le cadre du diagnostic technique.* PhD thesis, 1995.

[7] J-C Baillie. Urbi : Towards a universal robotic low-level programming language. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 820–825. IEEE, 2005.

[8] Marco Baioletti, Stefano Marcugini, and Alfredo Milani. An extension of satplan for planning with constraints. In *Artificial Intelligence : Methodology, Systems, and Applications*, pages 39–49. Springer, 1998.

[9] Eric Beaudry, Yannick Brosseau, Carle Côté, Clément Raïevsky, Dominic Létourneau, Froduald Kabanza, and François Michaud. Reactive planning in a motivated behavioral architecture. In *Proceedings of the National Conference on Artificial Intelligence*, volume 20, page 1242. Menlo Park, CA ; Cambridge, MA ; London ; AAAI Press ; MIT Press ; 1999, 2005.

[10] Michael Beetz and Drew V McDermott. Improving robot plans during their execution. In *AIPS*, pages 7–12, 1994.

[11] Michael Beetz, Lorenz Mosenlechner, and Moritz Tenorth. Cram a cognitive robot abstract machine for everyday manipulation in human environments. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1012–1017. IEEE, 2010.

[12] Renaud Bellais, Joseph Huysseune, Philippe Jarry, Sébastien Rolet, Med Kechidi, Marc-Daniel Seiffert, and Clair Juilliet. Marché du futur, usine du futur, avion du futur : quelles perspectives pour l'industrie aéronautique ? *Entreprises et histoire*, pages 146–160, 2014.

[13] Vieri Benci, Mauro Di Nasso, and Marco Forti. An euclidean measure of size for mathematical universes. *Logique et analyse*, 50(197) :43–62, 2007.

[14] Mouna Kamel Bessagnet, Eric Kergosien, Chantal Reynaud, Brigitte Safar, and Christian Sallaberry. Analyses linguistiques et techniques d'alignement pour creer et enrichir une ontologie topographique.

[15] Jean Bézivin and Xavier Blanc. Promesses et interrogations de l'approche mda. *Journal Développeur Référence*, pages 1–14, 2002.

[16] Jean Bézivin and Olivier Gerbé. Towards a precise definition of the omg/mda framework. In *Automated Software Engineering, 2001.(ASE 2001). Proceedings. 16th Annual International Conference on*, pages 273–280. IEEE, 2001.

[17] Xavier Blanc and Olivier Salvatori. *MDA en action : Ingénierie logicielle guidée par les modèles.* Editions Eyrolles, 2011.

[18] Max Blanchet. La nouvelle ère industrielle : une opportunité pour la france. *Géoéconomie*, (3) :159–173, 2014.

[19] Jonathan Bohren, Radu Bogdan Rusu, Edward Gil Jones, Eitan Marder-Eppstein, Caroline Pantofaru, Melonee Wise, Lorenz Mosenlechner, Wim Meeussen, and Stefan Holzer. Towards autonomous robotic butlers : Lessons learned with the pr2. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 5568–5575. IEEE, 2011.

[20] B Bonet and H Geffner. Hsp : Planning as heuristic search. *Entry at the AIPS-98 Planning Competition, Pittsburgh*, 1998.

[21] James Bornholt, Todd Mytkowicz, and Kathryn S McKinley. Uncertain : a first-order type for uncertain data. In *Proceedings of the 19th international conference on Architectural support for programming languages and operating systems*, pages 51–66. ACM, 2014.

[22] Caroline Boronad-Thierry. *Planification et ordonnancement multi-site : une approche par satisfaction de contraintes.* PhD thesis, 1994.

[23] Gerard Bourdon. *Strategie reactive d'accostage entre robots mobiles autonomes en milieu contraint. Approche par techniques floues.* PhD thesis, 1996.

[24] Paul Bourret, James Reggia, and Manuel Samuelides. Réseaux neuronaux(une approche connexionniste de l'intelligence artificielle). 1991.

[25] Leo Breiman, Jerome Friedman, Richard Olshen, Charles Stone, D Steinberg, and P Colla. Cart : Classification and regression trees. *Wadsworth : Belmont, CA*, 156, 1983.

[26] Thomas Breuer, Geovanny R Giorgana Macedo, Ronny Hartanto, Nico Hochgeschwender, Dirk Holz, Frederik Hegger, Zha Jin, Christian Müller, Jan Paulus, Michael Reckhaus, et al. Johnny : An autonomous service robot for domestic environments. *Journal of intelligent & robotic systems*, 66(1-2) :245–272, 2012.

[27] Rodney A Brooks. Symbolic reasoning among 3-d models and 2-d images. *Artificial intelligence*, 17(1) :285–348, 1981.

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

[29] Peter J Brown, John D Bovey, and Xian Chen. Context-aware applications : from the laboratory to the marketplace. *Personal Communications, IEEE*, 4(5) :58–64, 1997.

[30] Herman Bruyninckx. Open robot control software : the orocos project. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 3, pages 2523–2528. IEEE, 2001.

[31] Herman Bruyninckx. European phd school on robotic systems, January 2014.

[32] Herman Bruyninckx, Markus Klotzbücher, Nico Hochgeschwender, Gerhard Kraetzschmar, Luca Gherardi, and Davide Brugali. The brics component model : a model-based development paradigm for complex robotics software systems. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, pages 1758–1764. ACM, 2013.

[33] Oana Bucur, Philippe Beaune, and Olivier Boissier. Définition et représentation du contexte pour des agents sensibles au contexte. In *Proceedings of the 2nd French-speaking conference on Mobility and ubiquity computing*, pages 13–16. ACM, 2005.

[34] JL Burke and RR Murphy. Human-robot interaction in usar technical search : Two heads are better than one. In *Robot and Human Interactive Communication, 2004. ROMAN 2004. 13th IEEE International Workshop on*, pages 307–312. IEEE, 2004.

[35] A Caldas et al. Adaptive residual filtering for safe human-robot collision detection under modeling uncertainties. In *IEEE/ASME AIM*, pages 722–727, 2013.

[36] Gloria L Calhoun, Mark H Draper, Michael F Abernathy, Michael Patzek, and Francisco Delgado. Synthetic vision system for improving unmanned aerial vehicle operator situation awareness. In *Defense and Security*, pages 219–230. International Society for Optics and Photonics, 2005.

[37] Stéphane Cambon, Fabien Gravot, and Rachid Alami. A robot task planner that merges symbolic and geometric reasoning. In *ECAI*, volume 16, page 895, 2004.

[38] Robert H Cannon and Eric Schmitz. Initial experiments on the end-point control of a flexible one-link robot. *The International Journal of Robotics Research*, 3(3) :62–75, 1984.

[39] Evelyne Cauzinille-Marmèche, Jacques Mathieu, and Annick Weil-Barais. Raisonnement analogique et résolution de problèmes. *L'année psychologique*, 85(1) :49–72, 1985.

[40] Tarak Chaari. *Adaptation d'applications pervasives dans des environnements multi-contextes*. PhD thesis, institut national des sciences appliquées de Lyon, 2007.

[41] Wei-Fu Chang, Yu-Chi Wu, and Chui-Wen Chiu. Development of a web-based remote load supervision and control system. *International Journal of Electrical Power & Energy Systems*, 28(6) :401–407, 2006.

[42] Sam Chapman. Simmetrics-open source similarity measure library. *URL : http ://nazou. fiit. stuba. sk/home/documentation/concom/concom. doc, Visited :(April 2007)*, 2005.

[43] Raja CHATILA. *Robotique et simplexite, modeles, architecture, decision et conscience*. College de France, 2014, 2014.

[44] Leo H Chiang, Richard D Braatz, and Evan L Russell. *Fault detection and diagnosis in industrial systems*. Springer, 2001.

[45] Yves Chicheportiche, Paul R Bourdon, Haoda Xu, Yen-Ming Hsu, Hamish Scott, Catherine Hession, Irene Garcia, and Jeffrey L Browning. Tweak, a new secreted ligand in the tumor necrosis factor family that weakly induces apoptosis. *Journal of Biological Chemistry*, 272(51) :32401–32410, 1997.

[46] Azzeddine Chikh. Une approche méthodologique de réutilisation en ingénierie de document. *Document numérique*, 7(1) :59–88, 2003.

[47] E. Chouraqui. Le raisonnement analogique : sa problematique, ses applications. *Actes des Journees Nationales sur l'Intelligence Artificielle, Aix-les-Bains*, pages 107–117, 1986.

[48] Yan Ping Chu and Chang Jiang Zhu. The survey for ontology matching. In *Applied Mechanics and Materials*, volume 556, pages 6219–6222. Trans Tech Publ, 2014.

[49] Jose Luis Blanco Claraco. Development of scientific applications with the mobile robot programming toolkit. *The MRPT reference book. Machine Perception and Intelligent Robotics Laboratory, University of Málaga, Málaga, Spain*, 2008.

[50] Alexandra M Coddington and Michael Luck. A motivation-based planning and execution framework. *International Journal on Artificial Intelligence Tools*, 13(01) :5–25, 2004.

[51] Albert COLIN. Bras de manipulation et robots industriels. *Techniques de l'ingénieur. L'Entreprise industrielle*, (A9150) :A9150–1, 1995.

[52] Benoît Combemale. *Approche de métamodélisation pour la simulation et la vérification de modèle–Application à l'ingénierie des procédés*. PhD thesis, Institut National Polytechnique de Toulouse-INPT, 2008.

[53] Service communautaire d'information sur la Recherche et le developpement. Fabriquer un robot a l'image des plantes, 05 2014.

[54] Carle Cote, Yannick Brosseau, Dominic Letourneau, Clément Raïevsky, and Francois Michaud. Robotic software integration using marie. *International Journal of Advanced Robotic Systems*, 3(1) :55–60, 2006.

[55] Thierry Coudert, Bernard Grabot, and Bernard Archimède. Systèmes multiagents et logique floue pour un ordonnancement cooperative production/maintenance. *Journal of decision systems*, 13(1) :27–62, 2004.

[56] Douglas Crockford. The application/json media type for javascript object notation (json). 2006.

[57] Francisco Curbera, Matthew Duftler, Rania Khalaf, William Nagy, Nirmal Mukhi, and Sanjiva Weerawarana. Unraveling the web services web : an introduction to soap, wsdl, and uddi. *IEEE Internet computing*, 6(2) :86–93, 2002.

[58] Ernest Davis. *Organizing spatial knowledge.* PhD thesis, Yale University, Department of Computer Science, 1981.

[59] Elyane De Moura Braga. *Enseignement apprentissage, tice et environnement numerique de travail : etude des effets de ssupport didactiques numeriques, mediateurs dans la conceconceptualisation en statistique.* PhD thesis, Universite de Lyon 2, 2009.

[60] Kathrin Dentler, Ronald Cornet, Annette Ten Teije, and Nicolette De Keizer. Comparison of reasoners for large ontologies in the owl 2 el profile. *Semantic Web*, 2(2) :71–87, 2011.

[61] Karl Devooght. Modélisation de la capacité d'un agent intentionnel en fonction de ses activités, 2007.

[62] Samba Diaw, Redouane Lbath, and Bernard Coulette. État de l'art sur le développement logiciel basé sur les transformations de modèles. *Technique et Science Informatiques*, 29(4-5) :505–536, 2010.

[63] Rose Dieng-Kuntz and Fabien Gandon. Ontologies pour le web sémantique et le e-learning. *Web sémantique pour le e-Learning*, page 45, 2005.

[64] Patrick Doherty, Gösta Granlund, Krzysztof Kuchcinski, Erik Sandewall, Klas Nordberg, Erik Skarman, and Johan Wiklund. The witas unmanned aerial vehicle project. In *ECAI*, pages 747–755, 2000.

[65] Jill L Drury, Brenden Keyes, and Holly A Yanco. Lassoing hri : analyzing situation awareness in map-centric and video-centric interfaces. In *Human-Robot Interaction (HRI), 2007 2nd ACM/IEEE International Conference on*, pages 279–286. IEEE, 2007.

[66] V Dupourqué. Robulab 10, a service robot designed to aging-in-place. *Gerontechnology*, 8(3) :183, 2009.

[67] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping : part i. *Robotics & Automation Magazine, IEEE*, 13(2) :99–110, 2006.

[68] Ngo Duyhoa and Zohra Bellahsene. *Overview of YAM++-(not) Yet Another Matcher for ontology alignment task*. PhD thesis, LIRMM, 2014.

[69] Yassine El Ghayam and Mohammed Erradi. Distributed context management in collaborative environment. In *New Technologies of Distributed Systems (NOTERE), 2011 11th Annual International Conference on*, pages 1–8. IEEE, 2011.

[70] Abdeltif Elbyed. *ROMIE, une approche d'alignement d'ontologies à base d'instances*. PhD thesis, Institut National des Télécommunications, 2009.

[71] Karolina Eliasson. *The Use of Case-Based Reasoning in a Human-Robot Dialog System*. PhD thesis, Linköpings universitet, 2006.

[72] Karolina Eliasson. Case-based techniques used for dialogue understanding and planning in a human-robot dialogue system. In *IJCAI*, pages 1600–1605, 2007.

[73] John Ellson, Emden Gansner, Lefteris Koutsofios, Stephen C North, and Gordon Woodhull. Graphviz open source graph drawing tools. In *Graph Drawing*, pages 483–484. Springer, 2002.

[74] M. Endsley. Toward a theory of situation awareness in dynamic systems. *The Journal of the Human Factors and Ergonomics Society*, pages pages 32–64, 1995.

[75] M. R. Endsley. Designing for situation awareness : An approach to user-centered design. *Taylor & Francis US*, 2003.

[76] Mica R Endsley. Situation awareness misconceptions and misunderstandings. *Journal of Cognitive Engineering and Decision Making*, 9(1) :4–32, 2015.

[77] Rafael Epstein, Andrés Weintraub, John Sessions, Bren Sessions, Pedro Sapunar, Enrique Nieto, Fernando Bustamante, and Hugo Musante. Planex : A system to identify landing locations and access. In *Proc. of the International Mountain Logging and 11th Pacific Northwest Skyline Symp. P. Schiess and F. Krogstad [Eds.]. December*, pages 10–12, 2001.

[78] Jérôme Euzenat, Petko Valtchev, et al. Similarity-based ontology alignment in owl-lite. In *ECAI*, volume 16, page 333, 2004.

[79] Jonathan St BT Evans and Valerie A Thompson. Informal reasoning : theory and method. *Canadian Journal of Experimental Psychology/Revue canadienne de psychologie expÃ© rimentale*, 58(2) :69, 2004.

[80] Niels Fallenbeck and Claudia Eckert. It-sicherheit und cloud computing. In *Industrie 4.0 in Produktion, Automatisierung und Logistik*, pages 397–431. Springer, 2014.

[81] Daniel Faria, Catia Pesquita, Emanuel Santos, Matteo Palmonari, Isabel F Cruz, and Francisco M Couto. The agreementmakerlight ontology matching system. In *On the Move to Meaningful Internet Systems : OTM 2013 Conferences*, pages 527–541. Springer, 2013.

[82] Diego R Faria, Pedro Trindade, Jorge Lobo, and Jorge Dias. Knowledge-based reasoning from human grasp demonstrations for robot grasp synthesis. *Robotics and Autonomous Systems*, 62(6) :794–817, 2014.

[83] Charles Fattal and Violaine leynart. Processus d'evaluation d'armen. Technical report, Association Approche, 2012.

[84] Dieter Fensel. *Ontologies*. Springer, 2001.

[85] Richard E Fikes and Nils J Nilsson. Strips : A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3) :189–208, 1972.

[86] R. J. Firby. An investigation into reactive planning in complex domains. In *AAAI (Vol. 87, pp. 202-206).*, July, 1987.

[87] Philippe Fraisse. *Contribution à la commande robuste position/force des robots manipulateurs à architecture complexe. Application à un robot à deux bras.* PhD thesis, 1994.

[88] Peter M Frank. Analytical and qualitative model-based fault diagnosis–a survey and some new results. *European Journal of control*, 2(1) :6–28, 1996.

[89] Xianjin Fu, Zhenbang Chen, Yufeng Zhang, Chun Huang, and Ji Wang. Mpise : Symbolic execution of mpi programs. *arXiv preprint arXiv :1403.4813*, 2014.

[90] Jérémi Gancet. *Systèmes multi-robots aériens : architecture pour la planification, la supervision et la coopération.* PhD thesis, Institut National Polytechnique de Toulouse-INPT, 2005.

[91] Erann Gat et al. On three-layer architectures, 1998.

[92] P Haslum H Geffner and P Haslum. Admissible heuristics for optimal planning. In *Proceedings of the 5th Internat. Conf. of AI Planning Systems (AIPS 2000)*, pages 140–149, 2000.

[93] Olivier Gerbé, Guy W Mineau, and Rudolf K Keller. Un métamodèle des graphes conceptuels. *Revue d'intelligence artificielle*, 21(2) :255–284, 2007.

[94] Malik Ghallab and A Mounir Alaoui. Managing efficiently temporal relations through indexed spanning trees. In *IJCAI*, pages 1297–1303, 1989.

[95] Yassine El Ghayam. *La Sensibilite au contexte dans un environnement mobile.* PhD thesis, Ecole Nationale Superieure d'Informatique et d'Analyse des Systemes (ENSIAS), Rabat, 2011.

[96] M.W Ben Ghezala et al. Armen : Assistant robotique pour le maintien en environnement naturel. *Handicap 2014*.

[97] M.W Ben Ghezala et al. Rsaw : A situation awareness for autonomous robot. *ICARCV*, 2014.

[98] Maria Gini and Richard Voyles. *Distributed Autonomous Robotic Systems 7.* Springer, 2007.

[99] James Gips. *Shape grammars and their uses.* PhD thesis, Stanford University Palo Alto, CA, 1974.

[100] Yogesh Girdhar, Anqi Xu, Bir Bikram Dey, Malika Meghjani, Florian Shkurti, Ioannis Rekleitis, and Gregory Dudek. Mare : Marine autonomous robotic explorer. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 5048–5053. IEEE, 2011.

[101] Clark N Glymour. *The mind's arrows : Bayes nets and graphical causal models in psychology.* MIT press, 2001.

[102] Laurent Guigue and Christophe Richard. Le big data en santé préfigure-t-il la ≪médecine 3.0≫ ? *HEGEL [ISSN 2115-452X], 2014, 3*, 2014.

[103] D. Hahnel et al. Golex bridging the gap between logic (golog) and a real robot. *In KI : Advances in Artificial Intelligence*, 1998.

[104] Fayçal Hamdi, Brigitte Safar, Chantal Reynaud, and Haïfa Zargayouna. Alignment-based partitioning of large-scale ontologies. In *Advances in knowledge discovery and management*, pages 251–269. Springer, 2010.

[105] Fayçal Hamdi, Brigitte Safar, Haïfa Zargayouna, Chantal Reynaud, et al. Partitionnement d'ontologies pour le passage à l'échelle des techniques d'alignement. In *9eme Journées Francophones' Extraction et Gestion des Connaissances'*, 2009.

[106] Jean-Paul Haton and Marie-Christine Haton. *L'intelligence artificielle.* Presses universitaires de France, 1989.

[107] Nick Hawes. A survey of motivation frameworks for intelligent systems. *Artificial Intelligence*, 175(5) :1020–1036, 2011.

[108] Nick Hawes and Jeremy Wyatt. Engineering intelligent information-processing systems with cast. *Advanced Engineering Informatics*, 24(1) :27 – 39, 2010.

[109] Joachim Hertzberg, Herbert Jaeger, and Frank Schönherr. Learning to ground fact symbols in behavior-based robots. In *ECAI*, pages 708–712, 2002.

[110] Douglas R Hofstadter. *Godel, Escher, Bach.* Penguin, 2000.

[111] James C Houk and Andres G Barto. Distributed sensorimotor learning. *Advances in psychology*, 87 :71–100, 1992.

[112] Wei Hu, Ningsheng Jian, Yuzhong Qu, and Yanbing Wang. Gmo : A graph matching for ontologies. In *Proceedings of K-CAP Workshop on Integrating Ontologies*, pages 41–48, 2005.

[113] Charlotte Hug, Dominique Rieu, et al. Ingénierie des processus : une approche à base de patrons. *Actes du XXVème Congrès INFORSID*, pages 471–486, 2007.

[114] Wonil Hwang, Jinyoung Park, Hyowon Suh, Hyungwook Kim, and Il Hong Suh. Ontology-based framework of robot context modeling and reasoning for object recognition. In *Fuzzy Systems and Knowledge Discovery*, pages 596–606. Springer, 2006.

[115] F Ingrand et al. Prs : A high level supervision and control language for autonomous mobile robots. *ICRA (Vol. 1, pp. 43-49)*, April, 1996.

[116] Félix Ingrand. Architectures logicielles pour la robotique autonome. *JNRR*, 3 :8–10, 2003.

[117] Rolf Isermann. Estimation of physical parameters for dynamic processes with application to an industrial robot. *International Journal of Control*, 55(6) :1287–1298, 1992.

[118] Peter Jackson. *Introduction to expert systems*. Addison-Wesley Longman Publishing Co., Inc., 1990.

[119] Henrik Jacobsson, Nick Hawes, G-J Kruijff, and Jeremy Wyatt. Crossmodal content binding in information-processing architectures. In *Human-Robot Interaction (HRI), 2008 3rd ACM/IEEE International Conference on*, pages 81–88. IEEE, 2008.

[120] François Jammes, Antoine Mensch, and Harm Smit. Service-oriented device communications using the devices profile for web services. In *Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing*, pages 1–8. ACM, 2005.

[121] Jean-Marc Jézéquel, Benoit Combemale, Didier Vojtisek, et al. *Ingénierie Dirigée par les Modèles : des concepts à la pratique*. Ellipses, 2012.

[122] Ningsheng Jian, Wei Hu, Gong Cheng, and Yuzhong Qu. Falcon-ao : Aligning ontologies with falcon. In *Proceedings of K-CAP Workshop on Integrating Ontologies*, pages 85–91, 2005.

[123] JR Johnson and P Geissler. Surface changes observed by the mars exploration rovers. In *EPSC-DPS Joint Meeting 2011*, volume 1, page 1205, 2011.

[124] Marouen Kachroudi, Sami Zghal, and Sadok Ben Yahia. Bridging the multilingualism gap in ontology alignment. *International Journal of Metadata, Semantics and Ontologies*, 9(3) :252–262, 2014.

[125] Kenji Kaneko, Fumio Kanehiro, Mitsuharu Morisawa, Kanako Miura, Shinichiro Nakaoka, and Shuuji Kajita. Cybernetic human hrp-4c. In *Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS International Conference on*, pages 7–14. IEEE, 2009.

[126] Ichiro Kato, Sadamu Ohteru, Katsuhiko Shirai, Toshiaki Matsushima, Seinosuke Narita, Shigeki Sugano, Tetsunori Kobayashi, and Eizo Fujisawa. The robot musician wabot-2 (waseda robot-2). *Robotics*, 3(2) :143–155, 1987.

[127] Kuniaki Kawabata, Tomoki Akamatsu, and Hajime Asama. A study of self-diagnosis system of an autonomous mobile robot : expansion of state sensory systems. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 2, pages 1802–1807. IEEE, 2002.

[128] Mostepha Redouane Khouadjia, Marc Schoenauer, Vincent Vidal, Johann Dréo, and Pierre Savéant. Multi-objective ai planning : Evaluating dae yahsp on a tunable benchmark. In *Evolutionary Multi-Criterion Optimization*, pages 36–50. Springer, 2013.

[129] André Klarsfeld and Christophe Tribet. Technologies innovantes pour la santé. *PSL ITI*, 24(8) :942–946.

[130] Maarja Kruusmaa and Jan Willemson. Covering the path space : a casebase analysis for mobile robot path planning. *Knowledge-Based Systems*, 16(5) :235–242, 2003.

[131] Benjamin J Kuipers and Todd S Levitt. Navigation and mapping in large scale space. *AI magazine*, 9(2) :25, 1988.

[132] Bernard Laks. Langage et cognition : l'approche connexionniste. 1996.

[133] Christian Laugier. *Raisonnement géométrique et méthodes de décision en robotique : application à la programmation automatique des robots.* PhD thesis, Institut National Polytechnique de Grenoble-INPG, 1987.

[134] O. Lebec et al. High level functions for the intuitive use of an assistive robot. In *IEEE International Conference on Rehabilitation Robotics*, volume 2013, pages 6650374–6650374, 2013.

[135] Dominique Legallois. Essai sur la temporalité et le rythme du signe linguistique. *Langages*, pages 48–60, 2003.

[136] Daniel Leidner, Alexander Dietrich, Florian Schmidt, Christoph Borst, and Alin Albu-Schäffer. Object-centered hybrid reasoning for whole-body mobile manipulation.

[137] Solange Lemai and Felix Ingrand. Interleaving temporal planning and execution : Ixtet-exec. In *In Proceedings of the ICAPS Workshop on Plan Execution.*, March, 2003.

[138] Séverin Lemaignan, Raquel Ros, L Mosenlechner, Rachid Alami, and Michael Beetz. Oro, a knowledge management platform for cognitive architectures in robotics. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 3548–3553. IEEE, 2010.

[139] Christophe Leroux, Isabelle Laffont, Nicolas Biard, Sophie Schmutz, Jean François Désert, Gérard Chalubert, and Yvan Measson. Robot grasping of unknown objects, description and validation of the function with quadriplegic people. In *Rehabilitation Robotics, 2007. ICORR 2007. IEEE 10th International Conference on*, pages 35–42. IEEE, 2007.

[140] Blanca Li, Maywa Denki, Pierre Attrait, Jacques Chatelet, Charles Carcopino, and Thomas Pachoud. Robot !, 2013.

[141] Chung-Sheng Li and Rajiv Ramaswami. Automatic fault detection, isolation, and recovery in transparent all-optical networks. *Lightwave Technology, Journal of*, 15(10) :1784–1793, 1997.

[142] Jean Lieber. *Contributions à la conception de systèmes de raisonnement à partir de cas*. PhD thesis, Université Henri Poincaré-Nancy I, 2008.

[143] Jean Lieber and Amedeo Napoli. Raisonnement à partir de cas et résolution de problèmes dans une représentation par objets. *Revue d'intelligence artificielle*, 13 :9–35, 1999.

[144] Gi Hyun Lim, Il Hong Suh, and Hyowon Suh. Ontology-based unified robot knowledge for service robots in indoor environments. *Systems, Man and Cybernetics, Part A : Systems and Humans, IEEE Transactions on*, 41(3) :492–509, 2011.

[145] Dekang Lin. An information-theoretic definition of similarity. In *ICML*, volume 98, pages 296–304, 1998.

[146] Shuang Liu, Xiaoru Wanyan, and Damin Zhuang. Modeling the situation awareness by the analysis of cognitive process. *Bio-medical materials and engineering*, 24(6) :2311–2318, 2014.

[147] Kevin Lynch. *The image of the city*, volume 11. MIT press, 1960.

[148] V Maheu, Julie Frappier, PS Archambault, and F Routhier. Evaluation of the jaco robotic arm : Clinico-economic study for powered wheelchair users with upper-extremity disabilities. In *Rehabilitation Robotics (ICORR), 2011 IEEE International Conference on*, pages 1–5. IEEE, 2011.

[149] Maria Makarov. *Contribution à la modélisation et la commande robuste de robots manipulateurs à articulations flexibles. Applications à la robotique interactive*. PhD thesis, Supélec, 2013.

[150] Jacques Malenfant and Simon Denier. Architecture réflexive pour le contrôle de robots modulaires. *L'OBJET*, 10(2-3) :17–30, 2004.

[151] Maurice Maloux. *Dictionnaire des proverbes, sentences et maximes*. Larousse, 1990.

[152] Frédérique Maris, Pierre Régnier, and Vincent Vidal. Planification sat : Amélioration des codages et traduction automatique. *Proceedings du 14ème Congrès Francophone AFRIF-AFIA de Reconnaissance des Formes et Intelligence Artificiell e (RFIA'04)*, page 1019, 2004.

[153] Shani Martin. Navigation and taxis behaviorof the kompaï robot. 2013.

[154] Laure Martins-Baltar, Yann Laurillau, Gaëlle Calvary, et al. Débridons l'interaction homme-machine pour une meilleure qualité des soins. *Ingénierie des Systèmes d'Information*, 18(6) :113–139, 2014.

[155] Petr Matas. Architecture du capteur intelligent adaptatif. 2014.

[156] Berry Mbaiossoum, Ladjel Bellatreche, Stéphane Jean, and Michael Baron. Comparaison théorique et empirique de systèmes de bases de données sémantiques. *Ingénierie des Systèmes d'Information*, 18(3) :39–63, 2013.

[157] Brian McBride. Jena : Implementing the rdf model and syntax specification. In *SemWeb*, 2001.

[158] Mark McClelland, Tara Estlin, and Mark Campbell. Qualitative relational mapping and navigation for planetary rovers. *arXiv preprint arXiv :1402.0009*, 2014.

[159] Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. Pddl-the planning domain definition language. 1998.

[160] Drew V McDermott. A theory of metric spatial inference. In *AAAI*, pages 246–248, 1980.

[161] Vincent Creuze MCF and Montpellier LIRMM. Etat des lieux en robotique marine et sous-marine cas particulier de la commande des mini-véhicules sous-marins.

[162] K Meiners, JL Lieser, and GD Williams. Drones and robots monitor polar ice. 2014.

[163] Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm. Similarity flooding : A versatile graph matching algorithm and its application to schema matching. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 117–128. IEEE, 2002.

[164] Samir Menon et al. Mapping stiffness perception in the brain with an fmri-compatible particle-jamming haptic interface. In *Proceedings of the 14th Annual Conference of the IEEE Engineering in Medicine and Biology Society*, 2014.

[165] Giorgio Metta, Paul Fitzpatrick, and Lorenzo Natale. Yarp : yet another robot platform. *International Journal on Advanced Robotics Systems*, 3(1) :43–48, 2006.

[166] François Michaud, Carle Côté, Dominic Létourneau, Yannick Brosseau, J-M Valin, E Beaudry, Clément Raïevsky, Arnaud Ponchon, Pierre Moisan, Pierre Lepage, et al. Spartacus attending the 2005 aaai conference. *Autonomous Robots*, 22(4) :369–383, 2007.

[167] Rada Mihalcea and Janyce Wiebe. Simcompass : Using deep learning word embeddings to assess cross-level similarity. *SemEval 2014*, page 560, 2014.

[168] Takashi Minato, Michihiro Shimada, Hiroshi Ishiguro, and Shoji Itakura. Development of an android robot for studying human-robot interaction. In *Innovations in applied artificial intelligence*, pages 424–434. Springer, 2004.

[169] Marvin Minsky, Seymours Papert, and M IoT Perceptrons. Press. *Cambridge, Ma*, pages 105–110, 1969.

[170] Hossein S Mirheydar and J Kellogg Parsons. Diffusion of robotics into clinical practice in the united states : process, patient safety, learning curves, and the public health. *World journal of urology*, 31(3) :455–461, 2013.

[171] Mohamed Mohamed, Djamel Belaïd, and Samir Tata. Monitoring of sca-based applications in the cloud. In *CLOSER 2013 - Proceedings of the 3rd International Conference on Cloud Computing and Services Science, Aachen, Germany, 8-10 May, 2013*, pages 47–57, 2013.

[172] Douglas C Montgomery. Introduction to statistical quality control. 1991.

[173] Andrew W Moore, L Baird, and LP Kaelbling. Multi-value-functions : E cient automatic action hierarchies for multiple goal mdps. In *Proceedings of the international joint conference on artificial intelligence*, pages 1316–1323, 1999.

[174] James N Morgan and John A Sonquist. Problems in the analysis of survey data, and a proposal. *Journal of the American statistical association*, 58(302) :415–434, 1963.

[175] Morignot et al. Generating scenarios for a mobile robot with an arm. case study : Assistance for handicapped persons. *Eleventh International Conference on Control, Automation, Robotics and Vision (ICARCV'10)*, page P1054, December 2010,.

[176] Philippe Morignot, Mariette Soury, Christophe Leroux, Hélène Vorobieva, and Patrick Hède. Generating scenarios for a mobile robot with an arm : Case study : Assistance for handicapped persons. In *Control Automation Robotics & Vision (ICARCV), 2010 11th International Conference on*, pages 976–981. IEEE, 2010.

[177] Riadh Haj Mtir. *Le Raisonnement a Base de Cas dans les Systemes de E Learning Adaptation et Personnalisation.* PhD thesis, University of Manouba, Tunisia, 2010.

[178] Satoshi Murata, Eiichi Yoshida, Akiya Kamimura, Haruhisa Kurokawa, Kohji Tomita, and Shigeru Kokaji. M-tran : Self-reconfigurable modular robotic system. *Mechatronics, IEEE/ASME Transactions on*, 7(4) :431–441, 2002.

[179] Nicola Muscettola, Gregory A Dorais, Chuck Fry, Richard Levinson, and Christian Plaunt. Idea : Planning at the core of autonomous reactive agents. 2002.

[180] Nicola Muscettola, Chuck Fry, Kanna Rajan, Ben Smith, Steve Chien, Gregg Rabideau, and David Yan. On-board planning for new millennium deep space one autonomy. In *Aerospace Conference, 1997. Proceedings., IEEE*, volume 1, pages 303–318. IEEE, 1997.

[181] Karen L Myers. Cpef : A continuous planning and execution framework. *AI Magazine*, 20(4) :63, 1999.

[182] Swanson Dr Nancy. ≪la philosophie et les sciences, ou l'évangile selon popper≫ par le dr nancy swanson. 2014.

[183] Maximilien Naveau. Armen, assist mise en oeuvre de fonctions de manipulation dextre pour les robots sam et assist. Technical report, Technical Report CEA-LIST, France, 2013.

[184] Thomas Dyhre Nielsen and Finn Verner Jensen. *Bayesian networks and decision graphs*. Springer, 2009.

[185] Romeo Sanchez Nigenda, XuanLong Nguyen, and Subbarao Kambhampati. Altalt : Combining the advantages of graphplan and heuristic state search. In *Knowledge Based Computer Systems : Proceedings of the International Conference : KBCS–2000*, page 409. Allied Publishers, 2000.

[186] Erich George Nold. Describing students' pragmatic reasoning using" natural mathematics computer interfaces (nmi)". 2007.

[187] Anders Orebäck and Henrik I Christensen. Evaluation of architectures for mobile robotics. *Autonomous robots*, 14(1) :33–49, 2003.

[188] Fernando Santos Osório. *INSS : un système hybride neuro-symbolique pour l'apprentissage automatique constructif*. PhD thesis, Institut National Polytechnique de Grenoble-INPG, 1998.

[189] Lorena Otero-Cerdeira, Francisco J Rodríguez-Martínez, and Alma Gómez-Rodríguez. Ontology matching : A literature review. *Expert Systems with Applications*, 42(2) :949–971, 2015.

[190] Dömötör Pálvölgyi. Partitioning to three matchings of given size is np-complete for bipartite graphs. Technical report, Technical Report QP-2013-01, Egerváry Research Group, Budapest, 2013.

[191] Amit Kumar Pandey, Rodolphe Gelin, Rachid Alami, Renaud Viry, Axel Buendia, Roland Meertens, Mohamed Chetouani, Laurence Devillers, Marie Tahon, David Filliat, et al. Romeo2 project : Humanoid robot assistant and companion for everyday life : I. situation assessment for social intelligence. In *International Workshop on Artificial Intelligence and Cognition, 2nd Edition*, volume 1315, pages 140–147. CEUR Workshop Proceedings (CEUR-WS. org).

[192] Dejan Pangercic, Moritz Tenorth, Dominik Jain, and Michael Beetz. Combining perception and knowledge processing for everyday manipulation. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1065–1071. IEEE, 2010.

[193] Lynne E Parker. Path planning and motion coordination in multiple mobile robot teams. *Encyclopedia of Complexity and System Science. Springer, Heidelberg*, 2009.

[194] Simon Parsons, Ola Pettersson, Alessandro Saffiotti, and Michael Wooldridge. Robots with the best of intentions. In *Artificial Intelligence Today*, pages 329–338. Springer, 1999.

[195] Richard P Paul. *Robot manipulators : mathematics, programming, and control : the computer control of robot manipulators*. Richard Paul, 1981.

[196] Judea Pearl. *Probabilistic reasoning in intelligent systems : networks of plausible inference*. Morgan Kaufmann, 1988.

[197] Adam Pease. The sigma ontology development environment. In *Working Notes of the IJCAI-2003 Workshop on Ontology and Distributed Systems*, volume 71, 2003.

[198] D Pellier. Pddl4j. *Online] http ://sourceforge. net/projects/pdd4j*, 2011.

[199] Jorge Luis Pérez-Medina, Stéphanie Marsal-Layat, and JM Favre. Transformation et vérification de cohérence entre modèles du génie logiciel et modèles de l'interface homme-machine. In *INFORSID*, pages 382–397, 2007.

[200] Jesus Pestana, Jose Luis Sanchez-Lopez, Paloma de la Puente, Adrian Carrio, and Pascual Campoy. A vision-based quadrotor swarm for the participation in the 2013 international micro air vehicle competition. In *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*, pages 617–622. IEEE, 2014.

[201] Jean-Luc Petit. *Les neurosciences et la philosophie de l'action*. Vrin, 1997.

[202] Mathieu Petit, Brigitte Le Pévédic, and Dominique Duhaut. Génération d'émotion pour le robot maph : média actif pour le handicap. In *Proceedings of the 17th international conference on Francophone sur l'Interaction Homme-Machine*, pages 271–274. ACM, 2005.

[203] Ola Pettersson. Execution monitoring in robotics : A survey. *Robotics and Autonomous Systems*, 53(2) :73–88, 2005.

[204] AB Phillips, JIR Blake, SW Boyd, S Ward, and G Griffiths. Nature in engineering for monitoring the oceans (nemo) : an isopycnal soft bodied approach for deep diving autonomous underwater vehicles. In *Autonomous Underwater Vehicles (AUV), 2012 IEEE/OES*, pages 1–8. IEEE, 2012.

[205] Joelle Pineau, Michael Montemerlo, Martha Pollack, Nicholas Roy, and Sebastian Thrun. Towards robotic assistants in nursing homes : Challenges and results. *Robotics and Autonomous Systems*, 42(3) :271–281, 2003.

[206] Henri Poincaré. Les mathématiques et la logique. *Revue de métaphysique et de morale*, pages 294–317, 1906.

[207] Diego Pol and Mark A Norell. Comments on the manhattan stratigraphic measure. *Cladistics*, 17(3) :285–289, 2001.

[208] Eric Prud Hommeaux, Andy Seaborne, et al. Sparql query language for rdf. *W3C recommendation*, 15, 2008.

[209] Alain Pruski. Robots mobiles autonomes. *Techniques de l'Ingénieur*, pages 1–18, 1998.

[210] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros : an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5, 2009.

[211] Morgan Quigley, Michael A Goodrich, and Randal W Beard. Semi-autonomous human-uav interfaces for fixed-wing mini-uavs. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2457–2462. IEEE, 2004.

[212] M Ross Quillan. Semantic memory. Technical report, DTIC Document, 1966.

[213] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1) :81–106, 1986.

[214] Samer Qutub, Rachid Alami, and Félix Ingrand. How to solve deadlock situations within the plan-merging paradigm for multi-robot cooperation. In *Intelligent Robots and Systems, 1997. IROS'97., Proceedings of the 1997 IEEE/RSJ International Conference on*, volume 3, pages 1610–1615. IEEE, 1997.

[215] Ammar Rabaaoui. *Asservissement visuel par réseaux de neurones : Un algorithme neuronal pour l'asservissement visuel d'un robot mobile.* Éditions universitaires européennes, 2014.

[216] Ricco Rakotomalala. Arbres de décision. *Revue Modulad*, 33 :163–187, 2005.

[217] Brian Ricks and Ole J Mengshoel. Methods for probabilistic fault diagnosis : An electrical power system case study. In *Annual Conference of the Prognostics and Health Management Society*, 2009.

[218] Hayley Robinson, Bruce MacDonald, and Elizabeth Broadbent. The role of healthcare robots for older people at home : A review. *International Journal of Social Robotics*, pages 1–17, 2014.

[219] Raquel Ros, Ramon López De Màntaras, Carles Sierra, and Josep Lluís Arcos. A cbr system for autonomous robot navigation. In *CCIA*, pages 299–306, 2005.

[220] Michel Rueher. Modèle de traitement de rdf basé sur les graphes conceptuels.

[221] James Rumbaugh, Ivar Jacobson, and Grady Booch. *Unified Modeling Language Reference Manual, The.* Pearson Higher Education, 2004.

[222] Alessandro Saffiotti, Kurt Konolige, and Enrique H Ruspini. A multivalued logic approach to integrating planning and control. *Artificial intelligence*, 76(1) :481–526, 1995.

[223] Claude Samson. Commande non lineaire robuste des robots manipulateurs. 1983.

[224] Luis Sánchez, Ignacio Elicegui, Javier Cuesta, and Luis Munoz. On the energy savings achieved through an internet of things enabled smart city trial. In *Communications (ICC), 2014 IEEE International Conference on*, pages 3836–3841. IEEE, 2014.

[225] Emmanuel Sander. *L'analogie, du natif au creatif : analogie et categorisation.* Editions L'Harmattan, 2000.

[226] Matthias Scheutz, Paul Schermerhorn, and James Kramer. The utility of affect expression in natural language interactions in joint human-robot tasks. In

Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction, pages 226–233. ACM, 2006.

[227] Matthias Scheutz, Paul Schermerhorn, James Kramer, and David Anderson. First steps toward natural human-like hri. *Autonomous Robots*, 22(4) :411–423, 2007.

[228] Bill Schilit, Norman Adams, and Roy Want. Context-aware computing applications. In *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on*, pages 85–90. IEEE, 1994.

[229] Mohamed Sellami, Walid Gaaloul, and Samir Tata. Implementation of communities of web service registries. In *IEEE International Conference on Web Services, ICWS 2011, Washington, DC, USA, July 4-9, 2011*, pages 690–691, 2011.

[230] Varsha Shankar, Lena Sherbakov, Byron Galbraith, Aisha Sohail, Gennady Livitz, Anatoli Gorchetchnikov, Heather Ames, Frank H Guenther, and Massimiliano Versace. A co-robotic assistant capable of object selection and search via a brain machine interface. In *Neural Engineering (NER), 2013 6th International IEEE/EMBS Conference on*, pages 1441–1444. IEEE, 2013.

[231] Pavel Shvaiko and Jérôme Euzenat. Ontology matching : state of the art and future challenges. *Knowledge and Data Engineering, IEEE Transactions on*, 25(1) :158–176, 2013.

[232] Paul J Springer. *Military Robots and Drones : A Reference Handbook*. ABC-CLIO, 2013.

[233] S. Srinivasa, D. Ferguson, C. Helfrich, D. Berenson, A. Collet, R. Diankov, G. Gallagher, G. Hollinger, J. Kuffner, and M. VandeWeghe. Herb : A home exploring robotic butler. 2009. doi :10.1007/s10514-009-9160-9.

[234] Biplav Srivastava, Subbarao Kambhampati, and Minh B Do. Planning the project management way : Efficient planning by effective integration of causal and resource reasoning in realplan. *Artificial Intelligence*, 131(1) :73–134, 2001.

[235] George Stiny. Introduction to shape and shape grammars. *Environment and planning B*, 7(3) :343–351, 1980.

[236] Il Hong Suh, Gi Hyun Lim, Wonil Hwang, Hyowon Suh, Jung-Hwa Choi, and Young-Tack Park. Ontology-based multi-layered robot knowledge framework (omrkf) for robot intelligence. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 429–436. IEEE, 2007.

[237] Mikael Svenstrup, Søren Tranberg, Hans Jørgen Andersen, and Thomas Bak. Pose estimation and adaptive robot behaviour for human-robot interaction. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3571–3576. IEEE, 2009.

[238] Marie Tahon. *Analyse acoustique de la voix émotionnelle de locuteurs lors d'une interaction humain-robot*. PhD thesis, Université Paris-Sorbonne, 2013.

[239] Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2) :146–160, 1972.

[240] Moritz Tenorth and Michael Beetz. Knowrob knowledge processing for autonomous personal robots. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 4261–4266. IEEE, 2009.

[241] Maria Terzi, Matthew Rowe, Maria-Angela Ferrario, and Jon Whittle. Text-based user-knn : measuring user similarity based on text reviews. In *User Modeling, Adaptation, and Personalization*, pages 195–206. Springer, 2014.

[242] Cédric Tessier. *Système de localisation basé sur une stratégie de perception cognitive appliqué à la navigation autonome d'un robot mobile.* PhD thesis, Clermont-Ferrand 2, 2007.

[243] Yvon Tharrault, Gilles Mourot, José Ragot, David Fiorelli, Serge Gillé, et al. Identification de relations de redondance analytique pour le diagnostic de fonctionnement de capteurs d'une station d'épuration. *Journées Identification et Modélisation Expérimentale, JIME'2006, Poitiers, France*, 2006.

[244] Sebastian Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1) :21–71, 1998.

[245] Hai Bang Truong, Quoc Uy Nguyen, Ngoc Thanh Nguyen, and Trong Hai Duong. A new graph-based flooding matching method for ontology integration. In *Cybernetics (CYBCONF), 2013 IEEE International Conference on*, pages 86–91. IEEE, 2013.

[246] Cristina Urdiales, Eduardo J Perez, F Sandoval, and Javier Vazquez-Salceda. A hybrid architecture for autonomous navigation using a cbr reactive layer. In *Intelligent Agent Technology, 2003. IAT 2003. IEEE/WIC International Conference on*, pages 225–232. IEEE, 2003.

[247] Bert Van Oers. From context to contextualizing. *Learning and instruction*, 8(6) :473–488, 1998.

[248] Vincent Vidal and Héctor Geffner. Branching and pruning : An optimal temporal pocl planner based on constraint programming. *Artificial Intelligence*, 170(3) :298–335, 2006.

[249] Vincent Vidal and Sébastien Tabary. The new version of cpt, an optimal temporal pocl planner based on constraint programming. *ICAPS 2006*, page 50.

[250] Richard Volpe, Issa Nesnas, Tara Estlin, Darren Mutz, Richard Petras, and Hari Das. The claraty architecture for robotic autonomy. In *Aerospace Conference, 2001, IEEE Proceedings.*, volume 1, pages 1–121. IEEE, 2001.

[251] Hélène Vorobieva, Mariette Soury, Patrick Hède, Christophe Leroux, and Philippe Morignot. Object recognition and ontology for manipulation with an assistant robot. In *Aging Friendly Technology for Health and Independence*, pages 178–185. Springer, 2010.

[252] Ian Watson and Farhi Marir. Case-based reasoning : A review. *The knowledge engineering review*, 9(04) :327–354, 1994.

[253] Barbara Webb. Robots in invertebrate neuroscience. *Nature*, 417(6886) :359–363, 2002.

[254] Mark Weiser. Connecting the physical world with pervasive networks. 2002.

[255] Daniel S Weld, Corin R Anderson, and David E Smith. Extending graphplan to handle uncertainty & sensing actions. In *Aaai/iaai*, pages 897–904, 1998.

[256] Andy Wigley, Mark Sutton, Stephen Wheelwright, Robert Burbidge, and R Mcloud. *Microsoft. net compact framework : Core reference.* Microsoft Press, 2002.

[257] Jeannette M Wing and Farhad Arbab. Geometric reasoning : a new paradigm for processing geometric information. In *Design Theory for CAD, Elsevier Science Publishers, 1985, 145.* Citeseer, 1985.

[258] Holly A Yanco and Jill Drury. " where am i ?" acquiring situation awareness using a remote robot platform. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 3, pages 2835–2840. IEEE, 2004.

[259] Sami Yangui, Iain-James Marshall, Jean-Pierre Laisne, and Samir Tata. Compatibleone : The open source cloud broker. *Journal of Grid Computing*, 12(1) :93–109, 2014.

[260] Juan Ye, Graeme Stevenson, and Simon Dobson. A top-level ontology for smart environments. *Pervasive and mobile computing*, 7(3) :359–378, 2011.

[261] Hsueh-Chuan Yen, Tai-Chang Hsia, and Ren-Chieh Liao. Machine learning in a humanoid intelligent service robot. *Journal of Information and Optimization Sciences*, 35(2) :129–141, 2014.

[262] Jie Zhang. Improved on-line process fault diagnosis using stacked neural networks. In *Control Applications, 2002. Proceedings of the 2002 International Conference on*, volume 2, pages 689–694. IEEE, 2002.

[263] Stéphane Zieba. *Contribution à la résilience d'un système coopératif Homme-robot par une gestion de l'autonomie ajustable.* PhD thesis, Valenciennes, 2009.

[264] DA Zighed. Sipina for windows, ver 2.5. *Laboratory ERIC, University of lyon2*, 1996.

[265] Cong Zong, , et al. A mobile 3d vision based embedded system for robust estimation and analysis of human locomotion. In *Proceedings of the Sixteenth International Conference on Climbing and Walking Robots*, volume 2008, page 12. Hindawi Publishing Corporation, 2011.

Cinquième partie

ANNEXES

Annexe A

RAISONNEMENT DÉDUCTIF

Sommaire

A.1 Vocabulaire

Mot reservé	Type	Domain	Range
http://www.CEA-LIST.fr/Robot	URL		
Robot	Class		
Gripper	Class		
MobileBase	Class		
Arm	Class		
robotComposedBy	Object Property	Robot,	Arm, MobileBase
Characteristics	Class		
hasCharacteristics	Object Property	Robot,	Characteristics
ClampingForce	Class		
OpticalBarrier	Class		
Pressure	Class		
Weight	Class		
characteristicIs	Object Property	Characteristics,	ClampingForce, OpticalBarrier, Weight, Pressure
characteristicValue	Data Type Property	ClampingForce, OpticalBarrier, Weight, Pressure	float
hasName	Data Type Property	Robot,	string
armComposedBy	Object Property	Arm,	Gripper
HasFingerPos1	Data Type Property	Gripper,	float
hasFingerPos2	Data Type Property	Gripper,	float
hasFingerPos3	Data Type Property	Gripper,	float
hasFingerPos4	Data Type Property	Gripper,	float
hasFingerPos5	Data Type Property	Gripper,	float
Close	Data Type Property	Gripper,	float
Open	Data Type Property	Gripper,	float
hasDistance	Data Type Property	MobileBase,	float
hasVelocity	Data Type Property	MobileBase,	float
hasBattery	Data Type Property	MobileBase,	float
hasDOF1	Data Type Property	Arm,	float
hasDOF2	Data Type Property	Arm,	float
hasDOF3	Data Type Property	Arm,	float
hasDOF4	Data Type Property	Arm,	float
hasDOF5	Data Type Property	Arm,	float
hasDOF6	Data Type Property	Arm,	float
hasType	Data Type Property	Arm,	string
DefaultInitPosition	Class		
DefaultTransportPosition	Class		
http://www.CEA-LIST.fr/Environment	URL		
Object	Class		
hasXMAX	Data Type Property	Object	float
hasYMAX	Data Type Property	Object	float
hasXMIN	Data Type Property	Object	float
hasYMIN	Data Type Property	Object	Float
hasDistanceToRobot	Data Type Property	Object	Float
hasName	Data Type Property	Object	float
hasImage	Data Type Property	Object	float
IsIn	Object Property	Object	Object
isOn	Object Property	Object	Object
isUnder	Object Property	Object	Object
isAbove	Object Property	Object	Object
isLeft	Object Property	Object	Object
isRight	Object Property	Object	Object
isBehind	Object Property	Object	Object
isFront	Object Property	Object	Object
isManipulable	Data Type Property	Object	boolean
isClear	Data Type Property	Object	boolean
isTangible	Data Type Property	Object	boolean
GeopmetryShape	Class		
hasGeopmetryShape	Object Property	Object	GeopmetryShape
Location	Class		
hasLocation	Object Property	Object	Location

A.2 Règles

A.2.1 R-Rules

$@prefix ns\_robot :< http://www.cea-list.fr/Robot.owl\# > .$
//////ARM START//////////
$[armInitPositionTrue :$

$(ns\_robot : Robot-Default-Init-Positionns\_robot : hasDOF1?initDOF1value)$
$(ns\_robot : JacoArmPositionns\_robot : hasDOF1?jacoDOF1)\ equal(?initDOF1value, ?jacoDOF1)$
$(ns\_robot : Robot-Default-Init-Positionns\_robot : hasDOF2?initDOF2value)$
$(ns\_robot : JacoArmPositionns\_robot : hasDOF2?jacoDOF2)\ equal(?initDOF2value, ?jacoDOF2)$

$(ns\_robot : Robot-Default-Init-Positionns\_robot : hasDOF1?initDOF3value)$
$(ns\_robot : JacoArmPositionns\_robot : hasDOF3?jacoDOF3)\ equal(?initDOF3value, ?jacoDOF3)$

$(ns\_robot : Robot-Default-Init-Positionns\_robot : hasDOF4?initDOF1value)$
$(ns\_robot : JacoArmPositionns\_robot : hasDOF4?jacoDOF4)\ equal(?initDOF4value, ?jacoDOF4)$

$(ns\_robot : Robot-Default-Init-Positionns\_robot : hasDOF5?initDOF5value)$
$(ns\_robot : JacoArmPositionns\_robot : hasDOF5?jacoDOF5)\ equal(?initDOF5value, ?jacoDOF5)$

$(ns\_robot : Robot-Default-Init-Positionns\_robot : hasDOF6?initDOF6value)$
$(ns\_robot : JacoArmPositionns\_robot : hasDOF6?jacoDOF6)\ equal(?initDOF6value, ?jacoDOF6)$

$\rightarrow$

$(ns\_robot : jacons\_robot : arm - Isns\_robot : init - position)$

$]\ [armtransportPositionTrue :$

$(ns\_robot : Robot-Default-Transport-Positionns\_robot : hasDOF1?transportDOF1value)$
$(ns\_robot : JacoArmPositionns\_robot : hasDOF1?jacoDOF1)\ equal(?transportDOF1value, ?jacoDOF1)$
$(ns\_robot : Robot-Default-Transport-Positionns\_robot : hasDOF2?transportDOF2value)$
$(ns\_robot : JacoArmPositionns\_robot : hasDOF2?jacoDOF2)\ equal(?transportDOF2value, ?jacoDOF2)$

$(ns\_robot : Robot-Default-Transport-Positionns\_robot : hasDOF1?transportDOF3value)$
$(ns\_robot : JacoArmPositionns\_robot : hasDOF3?jacoDOF3)\ equal(?transportDOF3value, ?jacoDOF3)$

$(ns\_robot : Robot-Default-Transport-Positionns\_robot : hasDOF4?transportDOF1value)$
$(ns\_robot : JacoArmPositionns\_robot : hasDOF4?jacoDOF4)\ equal(?transportDOF4value, ?jacoDOF4)$

$(ns\_robot : Robot-Default-Transport-Positionns\_robot : hasDOF5?transportDOF5value)$
$(ns\_robot : JacoArmPositionns\_robot : hasDOF5?jacoDOF5)\ equal(?transportDOF5value, ?jacoDOI$

$(ns\_robot : Robot-Default-Transport-Positionns\_robot : hasDOF6?transportDOF6value)$
$(ns\_robot : JacoArmPositionns\_robot : hasDOF6?jacoDOF6)\ equal(?transportDOF6value, ?jacoDOI$

$\rightarrow$

$(ns\_robot : jacons\_robot : arm - Isns\_robot : transport - position)$

]
////////ARM END//////////////
///////GRIPPERSTART ///////////////
[openGripper :

(ns_robot :gripperofjaco ns_robot :gripper-HasFingerPos1 ?fingerDegree1) equal(?FingerPos1, 50)

(ns_robot :gripperofjaco ns_robot :gripper-HasFingerPos2 ?fingerDegree2) equal(?fingerDegree2, 50)

(ns_robot :gripperofjaco ns_robot :gripper-HasFingerPos3 ?fingerDegree3) equal(?fingerDegree3, 50) (ns_robot :gripperofjaco ns_robot :gripper-HasFingerPos4 ?fingerDegree4) equal(?fingerDegree4, 50) (ns_robot :gripperofjaco ns_robot :gripper-HasFingerPos5 ?fingerDegree5) equal(?fingerDegree5, 50)
→

(ns_robot :gripperofjaco ns_robot :gripper-Is ns_robot :open)

]

[closeGripper :

(ns_robot :gripperofjaco ns_robot :gripper-Has-Finger-Dregree1 ?fingerDegree1) equal(?fingerDegree1, 0)
(ns_robot :gripperofjaco ns_robot :gripper-Has-Finger-Dregree2 ?fingerDegree2) equal(?fingerDegree2, 0)

(ns_robot :gripperofjaco ns_robot :gripper-Has-Finger-Dregree3 ?fingerDegree3) equal(?fingerDegree3, 0)

(ns_robot :gripperofjaco ns_robot :gripper-HasFingerPos4 ?fingerDegree4)

equal(?fingerDegree4, 0) (ns_robot :gripperofjaco ns_robot :gripper-HasFingerPos5 ?fingerDegree5)
equal(?fingerDegree5, 0) →
(ns_robot :gripperofjaco ns_robot :gripper-Is ns_robot :close)

]

[gripper001ExempleJacothreefinger :

(ns_robot :gripperofjaco ns_robot :gripper-HasFingerPos1 ?fingerDegree1)
equal(?fingerDegree1, 0)
(ns_robot :gripperofjaco ns_robot :gripper-HasFingerPos2 ?fingerDegree2)
equal(?fingerDegree2, 0)

(ns_robot :gripperofjaco ns_robot :gripper-HasFingerPos3 ?fingerDegree3)
equal(?fingerDegree3, 50)

→

(ns_robot :gripperofjaco ns_robot :gripper-Is ns_robot :finger1-close)
(ns_robot :gripperofjaco ns_robot :gripper-Is ns_robot :finger2-close)
(ns_robot :gripperofjaco ns_robot :gripper-Is ns_robot :finger3-open)

]
[gripper010ExempleJacothreefinger :

(ns_robot :gripperofjaco ns_robot :gripper-HasFingerPos1 ?fingerDegree1)
equal(?fingerDegree1, 0)

(ns_robot :gripperofjaco ns_robot :gripper-HasFingerPos2 ?fingerDegree2)
equal(?fingerDegree2, 50)

(ns_robot :gripperofjaco ns_robot :gripper-HasFingerPos3 ?fingerDegree3)
equal(?fingerDegree3, 0)

→

(ns_robot :gripperofjaco ns_robot :gripper-Is ns_robot :finger1-close)
(ns_robot :gripperofjaco ns_robot :gripper-Is ns_robot :finger2-open)
(ns_robot :gripperofjaco ns_robot :gripper-Is ns_robot :finger3-close)

]
[gripper011ExempleJacothreefinger :

(ns_robot :gripperofjaco ns_robot :gripper-HasFingerPos1 ?fingerDegree1)
equal(?fingerDegree1, 0)

(ns_robot :gripperofjaco ns_robot :gripper-HasFingerPos2 ?fingerDegree2)
equal(?fingerDegree2, 50)

(ns_robot :gripperofjaco ns_robot :gripper-HasFingerPos3 ?fingerDegree3)
equal(?fingerDegree3, 50)

→

(ns_robot :gripperofjaco ns_robot :gripper-Is ns_robot :finger1-close)
(ns_robot :gripperofjaco ns_robot :gripper-Is ns_robot :finger2-open)
(ns_robot :gripperofjaco ns_robot :gripper-Is ns_robot :finger3-open)

]
[gripper100ExempleJacothreefinger :

(ns_robot :gripperofjaco ns_robot :gripper-HasFingerPos1 ?fingerDegree1)
equal(?fingerDegree1, 50)

(ns_robot :gripperofjaco ns_robot :gripper-HasFingerPos2 ?fingerDegree2)
equal(?fingerDegree2, 0)

(ns_robot :gripperofjaco ns_robot :gripper-HasFingerPos3 ?fingerDegree3)
equal(?fingerDegree3, 0)

→

(ns_robot :gripperofjaco ns_robot :gripper-Is ns_robot :finger1-open)
(ns_robot :gripperofjaco ns_robot :gripper-Is ns_robot :finger2-close)
(ns_robot :gripperofjaco ns_robot :gripper-Is ns_robot :finger3-close)

]
[gripper101ExempleJacothreefinger :

(ns_robot :gripperofjaco ns_robot :gripper-HasFingerPos1 ?fingerDegree1)
equal(?fingerDegree1, 50)

(ns_robot :gripperofjaco ns_robot :gripper-HasFingerPos2 ?fingerDegree2)
equal(?fingerDegree2, 0)

(ns_robot :gripperofjaco ns_robot :gripper-HasFingerPos3 ?fingerDegree3)
equal(?fingerDegree3, 50)

→

(ns_robot :gripperofjaco ns_robot :gripper-Is ns_robot :finger1-open)
(ns_robot :gripperofjaco ns_robot :gripper-Is ns_robot :finger2-close)
(ns_robot :gripperofjaco ns_robot :gripper-Is ns_robot :finger3-open)

]
[gripper110ExempleJacothreefinger :

(ns_robot :gripperofjaco ns_robot :gripper-HasFingerPos1 ?fingerDegree1)
equal(?fingerDegree1, 50)

(ns_robot :gripperofjaco ns_robot :gripper-HasFingerPos2 ?fingerDegree2)
equal(?fingerDegree2, 50)

(ns_robot :gripperofjaco ns_robot :gripper-HasFingerPos3 ?fingerDegree3)
equal(?fingerDegree3, 0)

→

(ns_robot :gripperofjaco ns_robot :gripper-Is ns_robot :finger1-open)
(ns_robot :gripperofjaco ns_robot :gripper-Is ns_robot :finger2-open)
(ns_robot :gripperofjaco ns_robot :gripper-Is ns_robot :finger3-close)

]
///////GRIPPER GRIPPER END////////////////
////// MOBILEBASE START//////// [mobileBaseBatteryHigh :
(ns_robot :robosoft ns_robot :mobile-Base-Has-Battery ?batteryValue)
greaterThan(?batteryValue, 50.0)

→

(ns_robot :robosoft ns_robot :mobile-Base-Has-Battery ns_robot :high)

]
[mobileBaseBatteryLow :

(ns_robot :robosoft ns_robot :mobile-Base-Has-Battery ?batteryValue)
lessThan(?batteryValue, 50.0)

→

(ns_robot :robosoft ns_robot :mobile-Base-Has-Battery ns_robot :low)

]
[mobileBaseAtDestination :

(ns_robot :robosoft ns_robot :mobile-Base-Has-Distance-To-Destination ?distance)
(ns_robot :robosoft ns_robot :mobile-Base-Has-Velocity ?velocity)
equal(?distance, 0)
equal(?velocity, 0)

→

(ns_robot :robosoft ns_robot :mobile-Base-Is ns_robot :at-destination)

]
[mobileBaseInMotion :

(ns_robot :robosoft ns_robot :mobile-Base-Has-Distance-To-Destination ?distance)
(ns_robot :robosoft ns_robot :mobile-Base-Has-Velocity ?velocity)
notEqual(?distance, 0)
notEqual(?velocity, 0)

→

(ns_robot :robosoft ns_robot :mobile-Base-Is ns_robot :in-motion)

]
[mobileBaseInMotion :

(ns_robot :robosoft ns_robot :mobile-Base-Has-Distance-To-Destination ?distance)
(ns_robot :robosoft ns_robot :mobile-Base-Has-Velocity ?velocity)
notEqual(?distance, 0)
equal(?velocity, 0)

→

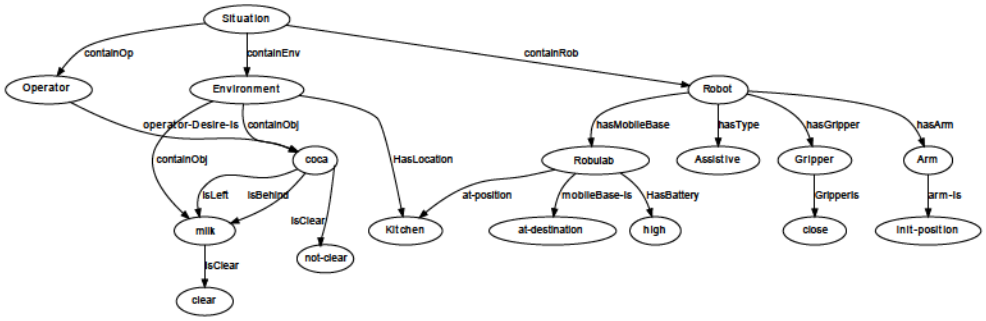


FIGURE A.1 – Exemple de situation déduite

(ns_robot :robosoft ns_robot :mobile-Base-Is ns_robot :off-state)

] /////// MOBILEBASE END ///////

A.2.2 E&OP-Rules

@prefix ns_environment : ¡http ://www.cea-list.fr/Environment.owl¿.

//////////////Spatial Relation Between Object//////// [leftright :

(?o1 ns_environment :HasDistanceToCamera ?o1_cameraToObjectDistance)
(?o1 ns_environment :HasXmax ?o1_xMax)
(?o1 ns_environment :HasXmin ?o1_xMin)
(?o1 ns_environment :HasYmax ?o1_yMax)
(?o1 ns_environment :HasYmin ?o1_yMin)

(?o2 ns_environment :HasDistanceToCamera ?o2_cameraToObjectDistance)
(?o2 ns_environment :HasXmax ?o2_xMax)
(?o2 ns_environment :HasXmin ?o2_xMin)
(?o2 ns_environment :HasYmax ?o2_yMax)
(?o2 ns_environment :HasYmin ?o2_yMin)

notEqual(?o1, ?o2)
equal(?o1_cameraToObjectDistance, ?o2_cameraToObjectDistance)
lessThan(?o1_yMax, ?o2_yMin)

→

(?o1 ns_environment :IsLeft ?o2)
(?o1 ns_environment :IsClear ns_environment :Clear)

(?o2 ns_environment :IsRight ?o1)
(?o2 ns_environment :IsClear ns_environment :Clear)
]
[frontbehind :

(?o1 ns_environment :HasDistanceToCamera ?o1_cameraToObjectDistance)
(?o1 ns_environment :HasXmax ?o1_xMax)
(?o1 ns_environment :HasXmin ?o1_xMin)
(?o1 ns_environment :HasYmax ?o1_yMax)
(?o1 ns_environment :HasYmin ?o1_yMin)

(?o2 ns_environment :HasDistanceToCamera ?o2_cameraToObjectDistance)
(?o2 ns_environment :HasXmax ?o2_xMax)
(?o2 ns_environment :HasXmin ?o2_xMin)
(?o2 ns_environment :HasYmax ?o2_yMax)
(?o2 ns_environment :HasYmin ?o2_yMin)

notEqual(?o1, ?o2)
lessThan(?o1_cameraToObjectDistance, ?o2_cameraToObjectDistance)
lessThan(?o1_yMin, ?o2_yMin)
lessThan(?o2_yMin, ?o1_yMax)

→
(?o2 ns_environment :IsBehind ?o1)
(?o2 ns_environment :IsClear ns_environment :Not-Clear)
(?o1 ns_environment :IsFront ?o2)
(?o1 ns_environment :IsClear ns_environment :Clear)

]
[aboveunder :

(?o1 ns_environment :HasDistanceToCamera ?o1_cameraToObjectDistance)
(?o1 ns_environment :HasXmax ?o1_xMax)
(?o1 ns_environment :HasXmin ?o1_xMin)
(?o1 ns_environment :HasYmax ?o1_yMax)
(?o1 ns_environment :HasYmin ?o1_yMin)

(?o2 ns_environment :HasDistanceToCamera ?o2_cameraToObjectDistance)
(?o2 ns_environment :HasXmax ?o2_xMax)

(?o2 ns_environment :HasXmin ?o2_xMin)
(?o2 ns_environment :HasYmax ?o2_yMax)
(?o2 ns_environment :HasYmin ?o2_yMin)

notEqual(?o1, ?o2)
le(?o2_xMin, o1_xMin)
le(?o2_xMin, o1_xMax)
le(?o2_xMax, o1_xMax)

→

(?o2 ns_environment :IsUnder ?o1)
(?o2 ns_environment :IsClear ns_environment :Clear)

(?o1 ns_environment :IsAbove ?o2)
(?o1 ns_environment :IsClear ns_environment :Clear)

] ///////END Spatial Relation Between Object////////
//////////BEGIN Location Of the Robot in the environment////////// [kitchen-
location : (?o1 ns_environment :locationHasX ?x)
(?o1 ns_environment :locationHasY ?y)
(?o1 ns_environment :locationHasTheta ?z)
equal(?x, 20.0)
equal(?y, 10.0)
equal(?z, 30.0)
→
(?o1 ns_environment :HasLocation ns_environment :Kitchen)
] [Doorlocation :
(?o1 ns_environment :locationHasX ?x)
(?o1 ns_environment :locationHasY ?y)
(?o1 ns_environment :locationHasTheta ?z)
equal(?x, 2.5)
equal(?y, 3.0)
equal(?z, 25.0)
→
(?o1 ns_environment :mobile-Has-location ns_environment :Door)
] ///////END Location Of the Robot in the environment////////

ANNEXE B

PLANIFICATION DES TACHES

Sommaire

B.1 ISEN : Interactive Scenarization ENgine

ISEN constitue le moteur de scénarisation du robot. Il permet de décrire le comportement du robot grâce à un enchaînement séquentiel d'actions géré par l'envoi et la réception d'événements. Ces actions correspondent à des commandes envoyées à l'entité "Robot". Cette description de comportement est décrite dans un fichier XML appelé scénario. Un séquenceur codé en C++ permet de faire l'interface entre ISEN et le robot, elle-même codée en C++. Il permet de charger les scénarios et les actions correspondantes ainsi que les fonctions du module ISEN. Les actions appelées dans les scénarios ISEN sont codées en C++ sous forme de fonctions. Cette architecture permet une lecture dynamique du scénario. ISEN envoie alors les ordres au robot qui fournit en retour des informations sur son état.

B.1.1 Concepts de base pour ISEN

Un scénario constitue la description globale de la tâche complexe à réaliser, comme "aller chercher à boire dans la cuisine". Il se décompose en séquences. Chaque séquence correspond à une tâche plus ou moins simple à effectuer telle que "bouger la base mobile", "attraper l'objet". A chaque séquence est associée une entité active appelée **agent** pour contrôler la progression d'une séquence et nous indiquer dans quel état se trouve le système. Ces agents peuvent agir en parallèle, du moment qu'ils n'utilisent pas la même ressource (par exemple deux agents ne peuvent ordonner au bras de

bouger dans deux directions différentes, mais un agent contrôlant le bras et un autre la base peuvent s'exécuter en même temps). Une séquence est elle-même décomposée en états. En fait, un état est une situation dans laquelle le comportement de tout ou partie du système n'évolue pas par rapport à ses entrées et sorties. A chaque état peut être associé une ou plusieurs actions. Les actions vont modifier l'état général du système. Elles font référence à des fonctions codées en C++ (soit des fonctions de base dans ISEN, soit des actions dans notre superviseur). Il est possible de faire appel à des actions dans trois cas différents : actions qui s'exécutent lors de l'entrée dans un état, lors de la sortie d'un état ou bien lorsqu'un événement précis intervient. Les événements sont des informations instantanées. Ils peuvent intervenir dans plusieurs séquences ou états différents. Ils peuvent servir à exécuter une action dans un état donné qui ne doit être exécutée que si un événement particulier se produit. Ils servent également pour passer d'un état à l'autre. Dans ce cas, une transition dédiée est associée. Une transition est une connexion unidirectionnelle reliant deux états. Pour que le scénario se déroule normalement les règles suivantes doivent être respectées :

- Initialisation : Chaque séquence doit posséder un état initial. Cet état est actif au début du fonctionnement. Lorsqu'on exécute en entier une séquence, pour la remettre à zéro, on renvoie la transition vers l'état initial.
- Transition : On ne peut effectuer une transition que si l'on a reçu l'événement qui lui correspond. De plus, on peut mettre une valeur conditionnelle à cette transition (égalité) : dans ce cas, lors de la réception de l'événement correspondant, la transition ne s'effectuera que si la condition est vérifiée. C'est cette notion, associée à une fonction de comparaison propre à ISEN, qui nous a permis par la suite de réaliser des boucles à l'intérieur d'une séquence.
- Evolution de l'état actif : Le franchissement d'une transition entraîne l'activation de l'état indiqué dans la transition et la désactivation de l'état de départ de la transition. Ainsi, dans chaque séquence, il existe toujours un unique état actif.

B.2 Notion de parallélisme et hiérarchisation

Avec le logiciel ISEN, il est possible de créer un modèle d'automate hiérarchique qui introduit donc la notion de sous-automates. Lorsque survient un événement prioritaire, on stocke l'état courant dans la pile, on lance le sous automate qui traite l'événement, et quand le sous-automate est terminé, on retourne à l'état que l'on a précédemment stocké au sommet de la pile. L'usage d'une pile permet d'empiler un nombre infini de traitements d'interruption de ce type, la machine est donc capable de gérer une combinatoire infinie des états élémentaires qui ont été décrits. La machine à état d'ISEN a été constituée à partir de cette notion d'automates à pile. Ainsi, on trouve dans ISEN la notion de ressource permettant d'éviter l'exécution de deux actions appelées en même temps. La notion de priorité règle le problème de conflit dans le cas où deux actions requièrent la même ressource. L'état prioritaire détient la

ressource puis la relâche lorsque l'on passe à l'état suivant. A ce moment-là, l'autre état qui détenait la ressource et qui était appelé en même temps, récupère la ressource et on passe alors dans cet état. Cette architecture permet donc d'aborder la modélisation de comportements complexes, réalisant des actions en parallèle si elles ne sont pas concurrentes.

B.3 PDDL du domaine

(define (domain PickandPlace)
(:predicates (room ?r)
(object ?b)
(gripper ?g)
(arm ?a)
(mobile-base ?b)
(init-position ?a)
(not-init-position ?a)
(transport-position ?a)
(not-transport-position ?a)
(at-position ?r ?b)
(at ?b ?r)
(free ?g)
(carry ?o ?g)
(door-closed ?pt)
(door-open ?pt)
(clear ?ob)
(not-clear ?ob)
(isRight ?x ?y)
(isLeft ?x ?y)
(isUnder ?x ?y)
(isAbove ?x ?y)
(isIn ?x ?y)
(isBehind ?x ?y)
(isFront ?x ?y)
(close ?x ?y))

(:action init-grasping ; Init-grasp
:parameters (?a ?r ?b)
:precondition (and (arm ?a)(room ?r)(mobile-base ?b) (not(init-position ?a))(not-init-position ?a)(at-position ?r ?b))
:effect (and (init-position ?a) (not (not-init-position ?a))(not-transport-position ?a)(not(transport-position ?a))(at-position ?r ?b)))

(:action arm-transport ; Arm-Transport
:parameters (?a ?r ?b)
:precondition (and (arm ?a) (room ?r)(mobile-base ?b)(not(transport-position ?a))(not-transport-position ?a)(at-position ?r ?b))
:effect (and (transport-position ?a) (not (not-transport-position ?a))(at-position ?r ?b)))
(:action move ; MoveToStation
:parameters (?from ?to ?a ?b)
:precondition (and (room ?from) (arm ?a)(room ?to) (mobile-base ?b) (at-position ?from ?b) (not (at-position ?to ?b))(door-open ?from)(transport-position ?a)(not-init-position ?a))
:effect (and (at-position ?to ?b) (transport-position ?a)(not-init-position ?a)(not (at-position ?from ?b))))

(:action opendoor ; opendoor
:parameters (?pt ?g ?a ?b)
:precondition (and (gripper ?g)(room ?pt) (arm ?a)(mobile-base ?b) (at-position ?pt ?b)(door-closed ?pt)(free ?g)(transport-position ?a))
:effect (and (door-open ?pt)(not(init-position ?a))(not-init-position ?a)(not(transport-position ?a))(not-transport-position ?a)))

(:action pick ; pick
:parameters (?obj ?a ?b ?room ?gripper)
:precondition (and (arm ?a)(object ?obj) (room ?room) (gripper ?gripper) (mobile-base ?b) (at ?obj ?room) (at-position ?room ?b) (free ?gripper)(clear ?obj)(init-position ?a)(not-transport-position ?a))
:effect (and (carry ?obj ?gripper) (not (at ?obj ?room))(not (free ?gripper))(not(clear ?obj)) (not-clear ?obj)(not(init-position ?a))(not-init-position ?a)))

(:action place ; drop
:parameters (?obj ?a ?b ?room ?gripper)
:precondition (and (object ?obj) (arm ?a) (room ?room) (gripper ?gripper) (mobile-base ?b) (carry ?obj ?gripper)(at-position ?room ?b) (not-transport-position ?a))
:effect (and (at ?obj ?room) (free ?gripper) (not (carry ?obj ?gripper))(not(init-position ?a))))

)

B.4 PDDL du problème

```
    (define (problem PickandPlace)


( :domain PickandPlaceDomaine)
( :objects
kitchen operator-position coca gripperjaco jaco robulab)
                                                        ( :init
(arm jaco)
(transport-position jaco)
(not-init-position jaco)
(not-gripper-open)
(gripper gripperjaco)
(free gripperjaco)
(room kitchen)
(room operator-position)
(mobile-base robulab)
(at-position operator-position robulab)
(object coca)
(clear coca)
(at coca kitchen)
)
( :goal (and
(at coca operator-position)))
)
```

Annexe C

MODÉLISATION

Sommaire

C.1 Principe de l'Ingénierie Dirigée par les Modèles

Dans cette section, nous définissions l'ingénierie Dirigée par les Modèles (IDM) ainsi que les concepts de modèle et de méta modèle avec les transformations dans cette méthodologie.

C.1.1 Définition de l'IDM

Model Driven Engineering (MDE) en langue anglaise ou L'Ingénierie Dirigée par les Modèles (IDM) est une nouvelle discipline du génie logiciel. Cette discipline met en valeur les modèles et les rend similaires à des entités de première classe dans le développement des logiciels. L'intérêt pour cette discipline a vu le jour en l'an 2000, quand l'Object Management Group (OMG) a rendu l'Architecture Dirigée par les Modèle (ADM) comme étant une marque déposée [16] visant un standard pour l'IDM. Cet intérêt s'est accru au niveau de la recherche académique [32][62][15] ainsi que dans les laboratoires industriels [17][2]. L'IDM a relancé le développement des systèmes complexes en leur offrant des améliorations significatives. Ces améliorations permettent de mettre l'accent sur un niveau plus abstrait que la programmation classique.
L'IDM est une forme d'ingénierie générative. Elle se distingue par une démarche dans laquelle tout ou partie de l'application informatique est engendré à partir de modèles. L'Ingénierie Dirigée par les Modèles [16] est une méthodologie offrant la possibilité de :

- Concevoir des applications tout en ayant une abstraction des technologies cibles ;

- Assurer la durabilité des applications conçues ;
- Simplifier la maintenance et l'adaptation aux changements ;
- Augmenter la productivité ;
- Cibler les différentes plateformes d'exécutions depuis une seule conception ;
- Réutiliser ce qui existe ;
- Faire le contrôle, la simulation et les tests à différents niveaux ;

Elle propose à travers les concepts de méta modèle et de transformation de modèles les moyens d'opérationnaliser la création et la manipulation des modèles. Un méta modèle est une description plus ou moins abstraite d'un langage de modélisation utilisé pour décrire un système. Sous la forme de règles définies au niveau des métas modèles, les transformations de modèles permettent de mettre en relation des modèles. En particulier, des transformations de modèles peuvent servir à projeter un modèle abstrait vers une représentation textuelle, avec pour conséquence la possibilité de bénéficier des outils utilisés dans un ou plusieurs domaines.

C.1.2 Modèles et méta modèle

Le modèle est défini dans l'IDM, montré dans la Fig. C.1, comme étant *Une abstraction d'un système, modélisé sous la forme d'un ensemble de faits construits dans une intention particulière. Un modèle doit pouvoir être utilisé pour répondre à des questions sur le système modélisé* [52].
Il doit être nécessaire et suffisant afin d'apporter des réponses à certaines questions à propos du système qu'il représente. Ce modèle doit avoir les mêmes réponses que le système lui-même. Ce principe s'appelle "la substituabilité". La première relation majeure de l'IDM est entre le modèle et le système qu'il représente. Cette relation est appelée "représentationDe". Un méta modèle est défini comme étant un modèle qui définit le langage d'expression d'un modèle [199], c'est à dire, le langage de modélisation. Cette notion conduit, donc, à une relation entre le modèle et le langage utilisé pour le construire, appelée "conforme à". La technique de modélisation des métamodèles est la métamodélisation.
D'après [121] : "*La métamodélisation est l'activité consistant à définir le méta modèle d'un langage de modélisation. Elle vise donc à bien modéliser un langage, qui joue alors le rôle de système à modéliser*".
Ainsi, l'IDM apparaît comme une solution pour spécifier, gérer et utiliser des modèles issus de domaines variés. De plus, elle permet la capitalisation des connaissances en fournissant la possibilité de réutiliser les modèles dans des développements différents.

C.1.3 Transformations

La deuxième problématique clé de l'IDM est d'opérationaliser les modèles à l'aide des transformations. Cette technique est au cœur de cette ingénierie et plus précisément de celle des DSML. Dans la littérature, il existe quatre formes de transformation de modèle.

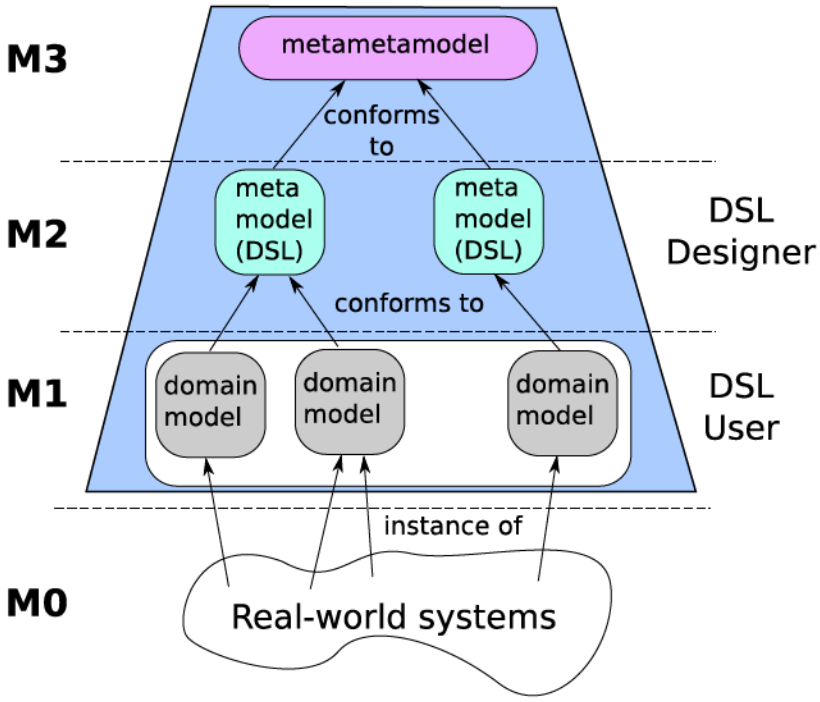


FIGURE C.1 – IDM : Les niveaux de modélisation [32]

1. Transformation endogène : Une transformation d'un modèle à un autre et que les deux modèles ont le même méta modèle.
2. Transformation exogène : Une transformation d'un modèle à un autre et que les deux modèles ont un méta modèle différent.
3. Transformation horizontale : Une transformation entre deux modèles ayant le même niveau d'abstraction.
4. Transformation verticale : Une transformation pour changer le niveau d'abstraction.

La Fig .C.2, montre les transformations possibles dans une approche basée sur l'Architecture Dirigée par les Modèles ainsi que leurs utilisations dans les systèmes.

C.2 Représentation des connaissances

Les applications modernes de l'informatique ont conduit à un usage généralisé des représentations des connaissances dans des contextes variés. Ces représentations s'ap-

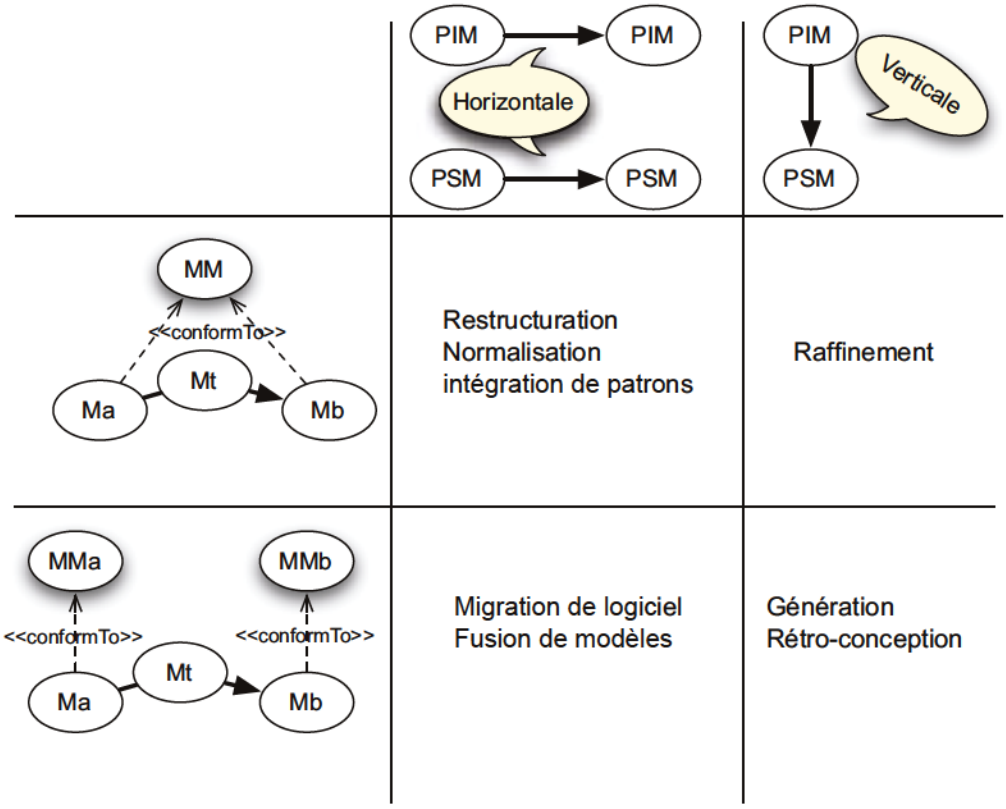


FIGURE C.2 – IDM : Les transformations et leurs utilisations [121]

puient sur des formalismes de représentation de connaissances, déterminant à la fois les types de connaissances qui peuvent être représentées et les mécanismes de raisonnement sur ces connaissances. Il existe différentes manières d'envisager la représentation des connaissances. Une première approche consiste à utiliser des représentations à base de faits et de règles. Cette approche, appelée ingénierie des connaissances, a été employée dans la mise au point des premiers systèmes experts, notamment Mycin [1] qui utilisait des outils inspirés des logiques multivaluées et qui est parvenu à de bons résultats, prouvant ainsi l'intérêt des logiques pour représenter des connaissances. Viennent après les systèmes à base de connaissances qui encodent des faits et des règles. Ils ont été conçus dans une perspective de résolution de problèmes relativement spécifiques, et leur fonctionnement est éloigné des capacités du cerveau humain (inférences très rapides, mais connaissances peu expressives). De ce fait,

1. http ://en.wikipedia.org/wiki/Mycin

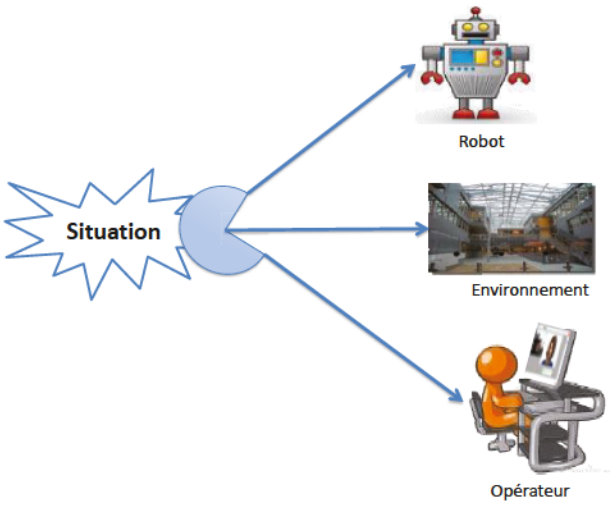


FIGURE C.3 – Concepts d'une situation courante

pour des tâches plus proches des préoccupations humaines, qui peuvent nécessiter des opérations de traitement automatique de la langue, d'autres approches ont été envisagées, en particulier des modèles de représentations des connaissances basés sur les objets et les catégories, qui sont plus aptes à simuler les modèles de représentations internes du "savoir humain" issus des sciences cognitives. Ces approches sont parfois qualifiées d'ingénierie ontologique.

Annexe D

RAISONNEMENT PAR ANALOGIE ET APPARIEMENT D'ONTOLOGIES

D.1 Raisonnement par analogie

L'analogie est au cœur de la pensée humaine [110]. Elle permet à l'être humain d'aborder les situations inconnues à partir de situations connues [225]. En effet, l'être humain utilise l'analogie dans la vie de tous les jours lorsqu'il doit affronter des situations nouvelles, jusque-là inconnues. En pédagogie, l'analogie est le mode de raisonnement le plus naturel et par conséquent le plus facile utilisé par les enfants. Les tests psychologiques d'évaluation de l'intelligence (tel que le test QI) comprennent aussi des épreuves de raisonnement par analogie.
L'étude de l'analogie constitue un objet d'étude pour plusieurs chercheurs dans des domaines différents. En particulier, le raisonnement par analogie, dans les domaines de la psychologie cognitive et l'intelligence artificielle, joue un rôle fondamental dans la résolution des problèmes et dans la construction des connaissances [47]. Les processus élémentaires constituant le raisonnement par analogie sont la modélisation du problème et la recherche d'une situation analogue [39]. Robot Situation AWareness (RSAW) vise à augmenter l'autonomie d'un système robotique en le rendant conscient des situations non prévues par l'opérateur et qui sont bloquantes pour le robot. On parle, alors, de situation de blocage. Par RSAW, nous voulons introduire la fonction d'apprentissage et de raisonnement dans le robot. Chez l'être humain, l'apprentissage repose sur quatre modes de raisonnement :

- La déduction : Ce mécanisme stipule que si on connaît la relation (A, B) et la relation1 (B, C) donc on peut déduire la relation $R(A, C)$ [79]. Exemple : Walid est un homme, or Tous les hommes sont mortels, Donc Walid est mortel.
- L'abduction : Ce mécanisme stipule que si on connaît $R(A, C)$ et $R(B, C)$, alors on "abduit" $R(A, B)$ [186] Exemple : Walid est mortel, Or tous les hommes sont mortels, Donc Walid est un homme.

- L'induction : Ce mécanisme stipule que si on connaît la relation (A, B) et la relation (A,C) alors on "induit" la relation(B, C) [206]. Exemple : Walid est un homme, Or Walid est mortel, Donc tous les hommes sont mortels.
- Analogie : (du fait au fait) : ce mécanisme de raisonnement consiste à faire l'hypothèse suivante : ≪ une propriété vraie d'un objet peut également l'être pour un autre présentant des similitudes avec le premier ≫. L'acceptabilité de la conclusion dépend de la similarité (ou ressemblance) entre ces deux objets.

Parmi ces modes, le choix du raisonnement par analogie est motivé par le fait que l'analogie est utilisée pour comprendre de nouvelles situations à partir de situations antérieures déjà connues (stockées dans la mémoire). Une analogie comporte une situation inconnue S et une situation connue S'. Faire une analogie, c'est trouver la similarité qui peut exister entre les deux situations S et S' [46]. Ce raisonnement repose sur le fait que :

"Si deux problèmes sont similaires alors leurs solutions sont similaires" [142].

Il s'agit donc d'un raisonnement fait à partir de cas connus. En informatique, la modélisation informatique du raisonnement par analogie est appelé le Raisonnement à Partir de Cas. Dans notre approche RSAW et comme décrit dans 5, la constitution de la situation courante est conforme à S-Ontology. La situation courante est décrite sous la forme d'un schéma RDF. Afin de trouver la situation la plus similaire à la situation courante, il faut parcourir S-Ontology, faire le filtrage des situations dont leurs types ne correspondent pas au type de la situation de blocage courante, extraire les situations non-filtrées en schéma RDF et les comparer à la situation courante en calculant la similarité entre eux. La comparaison de la situation courante et situation extraite revient à faire de l'appariement (ou matching) sur la base de schéma d'ontologie, voir [70]. Ce chapitre présente notre approche pour développer le raisonnement par analogie permettant à un robot de résoudre les situations de blocage durant son évolution dans l'environnement.

D.2 Raisonnement par analogie basée sur la mesure de similarité de Levenshtein

D.3 Raisonnement par analogie basée sur la comparaison d'ontologies

D.3.1 Concepts de base

L'analogie recherche les relations de cause à effet dans les situations passées afin de les adapter à la situation courante, en se basant sur les ressemblances entre les situations passées et la situation courante. Le raisonnement par analogie se limite à la recherche de similarité entre les situations passées et la situation courante. En

Alg. 9 CalculLevenshteinSimilarity

Input: `CS-RDF-Graph, S-Ontology ;`
Output: `SimilarSituation, float Similarity;`

`StoreSemantics(NodesOfCS-Graph, CS-StatementList);` {Transform the CS-RDF-Graph into StatementList of the current situation with Depth First Search (DFS)}
`Store(NumberOfNode-CS-StatementList);`
`CreateModel(S-Ontology);`

For each (**D**onode representing an individual)$individual - Graph \in S - Ontology$
  If $Individual.NumberOfNode = NumberOfNode - CS - StatementList$ **Then**
    `Extract (individual-Graph);`
    `StoreSemantics(NodesOfIndividual, S-StatementList);` {} LD = LevenshteinDistance $\in [0, 1]$
    $DBSits(CS - StatementList, S - StatementList) = \frac{\sum_{i=0}^{NumberOfNodes} LD(CS(i),S(i))}{NumberOfNode-CS-StatementList}$ {}DBSits = DistancebetweenSituations $\in [0, 1]$
    `Store(Levenshtein_DIST, S-StatementList.name, Similarity);`

    If $HighSimilarity \leq Similarity$ **Then**
      $HighSimilarity \leq Similarity$;
      $SimilarSituation = S - StatementList.name$;
    End If
  End If
End Forreturn $SimilarSituation$, $Similarity$

fait, ce raisonnement est un processus d'évocation d'un petit ensemble de situations concrètes (les cas). Il prend les décisions sur la comparaison de la situation courante (cas inconnu) avec celles de mémoire (cas connus). Il s'appuie sur l'hypothèse suivante : si une situation connue et la situation courante sont de similarité acceptable, alors tout ce qui peut être expliqué ou appliqué à la situation connue (dans la base de cas) reste valable si on l'applique à la situation courante qui représente le nouveau problème à résoudre [142]. D'un point de vue global, le raisonnement par analogie met en œuvre :

- une base d'expériences ou de cas,
- un mécanisme de recherche et d'extraction des cas similaires
- et une évaluation des solutions des cas extraits pour résoudre le problème.

Dans l'approche RSAW, le raisonnement par analogie s'effectue sur une situation représentée en RDFS avec S-Ontology décrit dans 5. Pour se faire d'une manière optimale, tout d'abord, nous avons conçu une étape de filtrage qui impose une typologie dans S-Ontology selon le type de blocage rencontré par le robot. Ensuite, une extraction, à travers une requête SPARQL, des situations non-filtrées est effectué afin d'obtenir des situations en schéma RDF. Après, un calcul de similarité entre chaque situation non-filtrée et la situation courante est élaboré pour choisir la situation possédant la similarité la plus élevée avec la situation courante. La similarité que nous voulons mettre en place est une similarité sémantique car il est important pour un robot de connaître la quantité d'information apportée par la situation courante

par rapport à son expérience. Enfin, selon le seuil que nous fixons l'algorithme choisira entre adapter ou regénérer un plan d'action.
Le calcul de similarité entre deux situations est de chercher les correspondances entre les éléments des deux ontologies représentant les situations. Cette correspondance entre les ontologies est appelé "Appariement ou Matching d'ontologies". Il existe plusieurs niveaux sur lesquels l'appariement d'ontologies peut être fait.

D.4 Technique de similarité topologique

Dans la littérature du niveau topologique (ou structurelle), il existe deux méthodes. La première est une méthode basée sur le graphe et la deuxième est une méthode basée sur la taxonomie. Cette dernière méthode prend en compte les relations "is-a" ou "part-of" et, dans les ontologies manipulées dans notre contribution, nous n'avons pas ce type de relation. La méthode basée sur les graphes s'agit de considérer une ontologie comme un graphe conceptuel. Il est composé par des nœuds représentant les concepts et des propriétés qui associent les concepts par des arcs orientés et étiquetés [93]. L'évaluation de la similarité de deux éléments dans deux ontologies (entre deux nœuds de deux graphes à comparer) est réalisée par la comparaison de leurs voisinages dans chaque ontologie (au sein de chacun des graphes). Les techniques distinguées dans la littérature sont OLA, GMO et la propagation de la similarité. Les techniques OLA et GMO se basent sur les graphes bipartis. En effet, les systèmes qui utilisent ce genre de techniques comme Falcon [231] transforment les ontologies représentées en RDF en un graphe biparti ; D'après Wikipédia, un graphe biparti est un graphe contenant deux sous-ensembles de sommets U et V telle que chaque arête ait une extrémité dans U et l'autre dans V. Dans ces techniques, deux triplets (issus de deux ontologies différentes) sont similaires si les concepts auxquels ils sont reliés ont les mêmes rôles. L'algorithme de la construction du PCG est décrit dans 10

Alg. 10 PCG-Construction

Input: SB, S ;
Output: PCG ; {S représente une situation non-filtrée}
{SB représente la situation de Blocage courante}
{Etape d'initialisation}

sbSuivant ← nœudRacine(SB) ;

sSuivant ← nœudRacine(S) ;

$sbTriplets \leftarrow extraireTriplet(SB)$;

$sTriplets \leftarrow extraireTriplet(S)$

$sbTypeTriplets \leftarrow extraireTripletDeType(SB)$;

sTypeTriplets ← extraireTripletDeType(S) ;

While $sb_suivant \not\leq videETs_suivant \not\leq vide$ **Do**
`sbNœudCandidat ← sbSuivant.extract();`

$sNœudCandidat \leftarrow sSuivant.extract()$;

$sbTripletsCandidats \leftarrow extraireTripletBaseSurSurjet(SB, sbNœudCandidat)$;

$sTripletsCandidats \leftarrow extraireTripletBaseSurSurjet(S, sNœudCandidat)$;

$supprimerElement(sbNœudCandidat, sbSuivant)$;

$supprimerElement(sNœudCandidat, sSuivant)$;

For each $sbtcTriplet \in sbTripletsCandidats$ **Do**
$sbtcPredicate \leftarrow extrairePredicat(sbtcTriplet)$;

$stcPredicate \leftarrow extraireTriplet(sTripletsCandidats, sbtcPredicate)$;

If $appariementTerminologique(EXACT, sbtcPredicate, stcPredicate) \leq 1$ **Then**
$stcTriplet \leftarrow extraireTriplet(sTripletsCandidats, stcPredicate)$;

$sbTypeTriplet \leftarrow extraireTriplet(sbTypeTriplets, sbtcTriplet.sujet)$;

$sTypeTriplet \leftarrow extraireTriplet(sTypeTriplets, stcTriplet.sujet)$;

If $appariementStructurel(sbtcTriplet, stcTriplet, sbTypeTriplet, sTypeTriplet) \leq 1$ **Then**
$pcgTriplet \leftarrow creerTriplet(sbtcTriplet, stcTriplet)$;

$ajouter(pcgTriplet, PCG)$;

$ajouter(sbtcTriplet.sujet, sbSuivant)$;

$ajouter(stcTriplet.sujet, sSuivant)$;

$supprimer(sbtcTriplet, sbTriplets)$;

$Supprimer(stcTriplet, sTriplets)$;

End If
End If
End For
End Whilereturn `PCG;`