Framework for context analysis and planning of an assistive robot

Mohamed Walid Ben Ghezala, Philippe Morignot, Amel Bouzeghoub and Christophe Leroux

Abstract— This paper presents the developments with the SAM robot, established in the ARMEN project. We are interested in cognitive robotics. We have developed two complementary modules. The first one deals with the representation of knowledge, while the second develops the scenario generation. Indeed, the representation of knowledge tells us about the scene, the current state of the robot and the strategy to be adopted by the robot to achieve goals specified by an assisted person. The information extracted from the knowledge representation is the starting point to generate the action plan and the implementation of the scenario by the robot.

I. Introduction

People losing their autonomy (disabled, elderly persons) and needing assistance in their everyday life generally resort to caretakers. In new approaches, some easy and frequent tasks can be done by a service robot in order to give more freedom and autonomy to those people.

Let's imagine a mobile robot dedicated to servicing a person (dependent person or caretakers) in the apartment and that this person asks for an object. This will entail for the robot to go to the room, to deploy its arm, to grasp the object, to retract the arm, to come back to the person's location and to hand the object over to the person. The ARMEN project is very ambitious for showing support to person. In this project we are working to implement a scenario regardless of the position and the location of the object. To respond to the desire of a person, the robot must know the environment and be able to generate and apply scenarios without the intervention of any technician, but with an intuitive interface with the person needing help.

In this paper we present a knowledge representation approach to determine the current state of SAM; we also show how to generate and execute scenarios for the robot SAM.

SAM is certified to life with people. The arm of SAM is MANUS. It is manufactured by Exact Dynamics. This arm is not heavy and does not represent a danger to person unlike other robotic arms.

We start this section by presenting the ARMEN project and the robot SAM. Then we present the knowledge representation aspect for SAM. After deducing the current state of knowledge representation and the behavior to be performed by SAM, we show how we develop the generation of action to achieve goals imposed by a disabled person.

Manuscript received March 10, 2012. This work was supported by the DGE of the French Ministry of Economy, Finance and Industry through contract ITEA 2 MIDAS, as part of a EUREKA European project.

M.W. Ben Ghezala (mohamedwalid.benghezala@cea.fr , phone +33 (0)1 46 54 91 98, fax : +33 (0)1 46 54 89 90), P. Morignot (philippe.morignot@cea.fr , phone : +33 (0)1 46 54 97 27, fax : +33 (0)1 46 54 89 90), C. Leroux (christophe.leroux@cea.fr) are with CEA, LIST, Interactive Robotics Laboratory, 18, route du panorama, B.P. 6, 92265 Fontenay -aux-Roses, France.

A. Bouzeghoub (amel.bouzeghoub@it-sudparis.eu), phone +33 (0) 1 60 76 47 14 is with Institute TELECOM, TELECOM & Management SudParis, 9, Rue Charles Fourier, 91000 Evry, France.

II. Context

1) The project ARMEN

The goal of this project, developed by CEA-LIST, is to design a robotic assistant with a navigation system independent of location and obstacle avoidance. Thanks to its sensors, the system is able to avoid obstacles by taking into account the volume occupied by the robot and the equipment it carries. Controlled by an intuitive man-machine interface, the robot is able to move in a domestic environment, to recognize and to grasp an object and bring it to the desired location. The robot is easy to use, so that a person, not a specialist in robotics or computer science, can adapt and configure the robot as well as create usage scenarios tailored to each user.

2) The robot SAM

Fig 1: The robot SAM includes a 6 DOF arm and a gripper.

The robot SAM (Smart Autonomous Majordomo [19]) is a non-holonomic mobile base ROBULAB 10 with a 6-DOF MANUS arm ending with a gripper (see Fig1). Its sensors are forward- and backward-oriented sonars located on the base (for obstacle avoidance), a panoramic camera located on top of the base (for scene detection), 2 webcams located on the arm (for object recognition, distance stereo-measurement, and visual servoing [15]) and an optical

barrier located in the gripper (for decision on clamp closure/opening).

3) Software

In the ARMEN project, the robot SAM must be able to know the environment and generate the appropriate action plans to achieve the goals set by the operator (dependent person or caregivers). Knowledge of the environment is made by the knowledge representation aspect of the robot. The knowledge representation is useful to know the state of the robot at any moment. The generation of action plans is the generation of a scenario to achieve the goals from the current state of the robot. The current state is issued from the knowledge representation. The major advantage of this method is that it is possible to avoid an obstacle by re-planning a scenario.

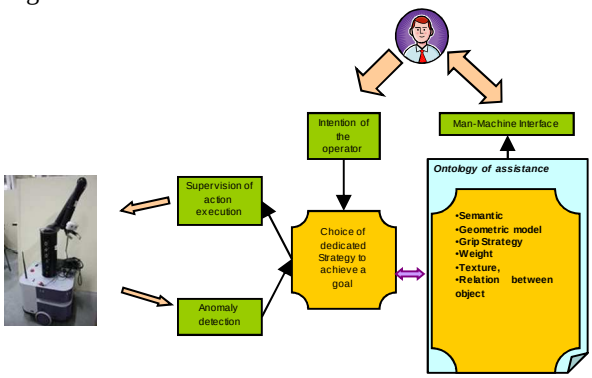

Fig2: Software for SAM

III. KNOWLEDGE REPRESENTATION

We developed a knowledge representation for knowing the current state of the robot. As a branch of symbolic Artificial Intelligence, knowledge representation and reasoning aims at designing computer systems that reason about a machine-interpretable representation of the world, similar to human reasoning. Different technique of knowledge representation may be used. We will present in this paper some of the most popular approaches to knowledge representation and our approach.

1) Rules

This representation model is widespread. It can easily be understood by human. The rules allow dynamic knowledge representation. Syntax representation of rules is:

IF Premise (s) THEN consequence (s).

In this approach, the attributes represent the internal data. The rules require the experience of developer. They are dynamic and can be archived and updated if necessary. In a system of automated reasoning, it is easy to apply this approach in building a robot.

2) Frames [25]

Frames assume that human knowledge is not complicated but structured around units of information. All scenarios of everyday life can be represented as frames. A frame is a data structure including both declarative and procedural information. It represents a typical situation and includes slots for objects. Each attribute (slot) has a unique aspect (facets) of the description of the concepts that it represents.

3) Semantic network

From psychological models of Quillian and Raphael [26], semantic networks are tools that simulate the performance of memory. This is a model that shows 1) how the information could be represented in memory and 2) how one can access this information. A semantic network consists of nodes whose interrelationships are established by labeled pointers. The nodes are the different types of information in memory. Every node can be associated proposals and statements that characterize the properties applying to the network nodes. The label attached to the pointer indicates what type of relationship between two nodes. There is no standard in relationships, but there are common relations:

X Instance Y; X isa Y; X haspart Y

Semantic networks use metadata to represent the definition of different information.

4) Logic

Logic is in a family of knowledge representation languages which can be used to represent the terminological knowledge of an application domain in a formal and structured. It was developed as an extension of frames and semantic networks, which did not have formal semantics based on logic.

5) Ontology

An ontology is an explicit specification of a conceptualization. The term is borrowed from philosophy, where Ontology is a systematic account for Existence. For A.I. systems, what “exists” is what can be represented. When the knowledge of a domain is represented in a declarative model, the set of objects that can be represented is called the universe of discourse. This set of objects, and the describable relationships among them, are reflected in the representational vocabulary with which a knowledge-based program represents knowledge.

This approach offers expressiveness and understanding in knowledge representation. An ontology represents a set of structured concepts, concepts are organized in a graph whose the relation can be semantic and / or composition and inheritance. An ontology offers the possibility to have a shared vocabulary to describe a domain as well as primitive typing classes and relationships. The most important is that the ontology can make reasoning (deduce new facts from existing ones).

6) The choice of ontology for knowledge representation

The different techniques of knowledge representation mentioned above have certainly been a contribution to the introduction of intelligence into robotic systems. They have at least helped to the test feasibility.

However, some drawbacks are noted, especially with techniques based on rules, frames, semantic networks, concept diagrams and logic. These gaps are actually due to a slow system, the increasing complexity when it is appropriate to consider the classification and all causal links and others that may exist in the representation of the context and environment in which the robot will move. With the need for more interactivity between the operator and the robot, the ontology is presented as an approach that could help remedy the negative findings of knowledge representation techniques mentioned above. In addition, working with ontology allows gaining interoperability by providing common access to information and a shared understanding of concepts. They allow the reuse of knowledge sources.

IV. ONTOLOGY FOR SAM

In this paragraph, we will demonstrate the use of Protégé for build our ontology and we present a conception of SAM's ontology.

MLCOF (The Multi-Layered Context Ontology Framework) describes the context of a robot. MLCOF includes six Knowledge Layers (KLayer): image, 1D geometry, 2D geometry, 3D geometry, object and space. The main propose of MLCOF is to help robots in object identification tasks. [30]

OMRKF: Ontology-based Multi-layered Robot Knowledge Framework is an extension of MLCOF. This robot centered description ontology is organized in knowledge boards with four knowledge levels: perception, model, context and activity. [33]

KnowRob, a knowledge processing framework based on Prolog. Its underlying storage is based on OWL ontology such as researchCyc and OMICS (indoor common-sense knowledge database) [32].

K-CoPMan (knowledge enable Cognitive Perception for Manipulation) system is an extension to KnowRob. This technology enables autonomous robots to grasp and manipulate object. K-CoPMan, uses CAD for matching 3D point clouds in order to identify the queried object in the environment. [31]

ORO: The OpenRobots Ontology is a socket server aimed to be run on robots that maintains a consistent storage of facts, represented as RDF triples and runs several background processes, including ontology classification and reasoning, management of several independent models for each different agent the robot meets, and updating of statements according to bio-inspired memory models.[34]

Our project is ambitious, we want that if the cameras are broken the robot will warn the person. So we find in this ontology a Robot concept. This concept is necessary because it allows SAM to do its self-assessment. When one of sensor is broken the robot can know it and trigger a command for warning an operator.

Also with this ontology we can know the interlocutor by the concept "user profile". With this concept we have a GIR (Groupe Iso Ressource) of dependent person and the coordinate of his doctor.

In this ontology, objects are categorized by their type (e.g., coca-type cans). Each object in the ontology has several properties. Among other properties there is a set of images of the object. Each image is an image of the object which depends on the point of view of the object relatively to the robot. Each image is associated to grip strategies. We use this method because our recognition method uses image indexing and allows estimating the angle or point of view on the object regarding the position of the arm [23]. So when SAM is in front of a scene, we load the images from the ontology and thus we recognize the current state (where is Sam, what its position is) and how we can achieve the goal by knowing the grip strategy of the objects. The grip strategy is propriety of each object and it depends on the point of view of object. Each image in our ontology is linked to this propriety. This propriety is essential to know how to behave with object. (Fig 3) is extracted from the ontology created for SAM to represent his knowledge.

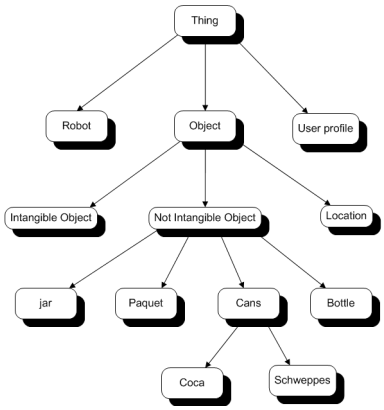

Fig 3: Ontology of SAM

With this ontology we know a situation of the object which the robot should grasp. Now we turn to the generation of action plan (scenario) after having presented related work in the development of scenarios.

V. SCENARIO GENERATION

1) State of the art

Many robots can carry out a single scenario (e.g., ROLLIN JUSTIN [6], NAO [9], TWENDY ONE [13], HRP-4 [1]). In these systems, a scenario is represented as a

piece of source code which calls functions of a programming language for the robot to successively carry out all the prescribed actions. But to change a scenario (e.g., for a new demo), changing this source code is necessary.

Other robots use a high level language for representing scenarios (e.g., CARE-O-BOT [10], among others). In these systems, high level actions can be synthetically encoded in a language, so reprogramming the robot for a new scenario is not necessary: Only the high level actions in the scenario have to be changed. The robot CARE-O-BOT uses a high level language, i.e., a module in Python, to encode scenarios using activities linked by discrete, cyclic or wait-for relations [10]. A.I. planning is performed by a request to a database, which provides possible actions that fulfill a given task. When compared to our approach, CARE-O-BOT's high level language is subsumed by the ISEN one: The previous high level primitives can be encoded in ISEN, due to the absence of any constraint imposed by ISEN on the graph structure of states. Most importantly, CARE-O-BOT does not include an A.I. task planner *per se*, hence leading to manually encoding all the possible actions for all the tasks.

Still other robots' designers acknowledge the fact that there are many scenarios which would have to be manually written in real applications (not only for demos). And that it is unpractical, if not impossible, to manually write all of them in advance (for that, even a high level language is not sufficient any longer). So these systems generate their own scenarios (e.g., SHAKEY [5], PR2 [24], DALA & LAMA [11]) using an A.I. task planner: Given goals, specified by a user, task planning generates a plan of actions (a sequence of instantiated action descriptions), this plan is considered as a scenario (a high level description of the successive actions to take) and then this scenario is executed by the robot (each action description in the scenario is linked to an executable function in the underlying programming language, and these functions are executed sequentially).

First of all, the robot SHAKEY pioneered the field of domain-independent A.I. task planning, with the STRIPS task planner and the PLANEX execution mechanism [5]. But nowadays A.I. planners, e.g. CPT, are degrees of magnitude faster. Indeed, CPT (Constraint Programming Temporal planner [22]) is an optimal temporal planner combining the connection using the causal links in the partial plans (Partial Order Causal Link or POCL) and pruning rules based on constraint programming [28]. CPT was awarded a second prize in the optimal planning of IPC-2004.

The robot PR2 uses no specific high level language to represent scenarios, but uses a modified Hierarchical Task Network (HTN) planner (see [7] for an introduction) to generate them [24]. HTN planners represent additional knowledge on tasks (i.e., a sequence of low level subtasks decomposes a high level task) to reduce search complexity. In contrast, the CPT planner is domain-independent (hence does not need extra knowledge on task decomposition) and still performs fast --- it won the Distinguished Performance award at IPC'06 [12]

The robots DALA & LAMA [11] use an A.I. task planner, IxTeT [16], which is a domain-independent task planner based on constraint programming (CPT is based on the same underlying principle but with a different model). But IxTeT has a larger representation capability than PDDL --- it was designed before. As such, IxTeT is not involved in any International Planning Competition [12].

Finally, Dornhege *et al.* [3] proposes to extend PDDL: The truth value of specific preconditions is not determined by the successive postconditions of previous operators, but is set by calling functions querying and analyzing sensors --- symmetrically, postconditions can also own a call back function, to act on actuators. This approach merges plans and scenarios, hence leading to an architecture less clear than ours.

Generating and executing scenarios for SAMIn this section, we describe the high level language ISEN, capable of representing and executing scenarios, and a way to generate these scenarios through an A.I. task planner (e.g., CPT).

B. ISEN: a high level language and an engine

The ISEN engine is a virtual machine which reacts to events sent by the application in which it is integrated, and triggers actions that can act on this application. And this, by conforming to scenarios which specify a behavior model.

At initialization time, the application must provide to ISEN a library of elementary actions which can be taken by the engine, and a behavior model, or scenario (provided as an XML file). At execution time, the application sends to the engine the events generated either by the application or by the external environment. The engine calls functions from the previous library of actions, as specified by scenarios, to act on the application and on the external environment.

A scenario specifies any number of state machines (or agents[1]), which autonomously react to events by changing states and/or triggering actions. Every event sent by the application is transmitted to all state machines included into the scenario. Only the state machines tailored for reacting to this event actually does --- the other state machines simply ignore it.

Programming a set of ISEN state machines hence sums up to: (1) specifying a list of states in which the state machine can fall into; (2) for each previous state, specifying the next state to which to branch (i.e., a transition); And (3) specifying the actions to take for each state transition or event reception.

[1] This term should not be confused with the term "agent" in the A.I. multiple agent systems community, for example.

A scenario is composed of a set of automata (or sequences), a set of agents and a list of global constants (visible by all states of all automata of all agents). An automaton is composed of a set of states and a set of transitions, or state change, which can be activated in the considered state. The target state of a transition can be another state of the current sequence, or the initial state of another sequence. Each agent specifies its initialization sequence and its local constants.

Three kinds of actions can be attached to a state: ON_ENTRY actions (noted $f_{1,1}$ to $f_{1,n1}$ for state *n* in Fig. 4), which are activated in sequence just before the current state is activated; ON_DO actions (noted $f_{2,1}$ to $f_{2,n2}$ for the same state in the same figure), which are activated in sequence during the current state's activation; and ON_EXIT actions (noted $f_{3,1}$ to $f_{3,n3}$ for the same state in the same figure), which are activated in sequence right after the current state is activated. As a consequence, from the time when a state is about to become active (state *n*, in green in Fig 4), the actions are called in the following order: $f_{1,1}$, …, $f_{1,n1}$, $f_{2,1}$, …, $f_{2,n2}$, $f_{3,1}$, …, $f_{3,n3}$.

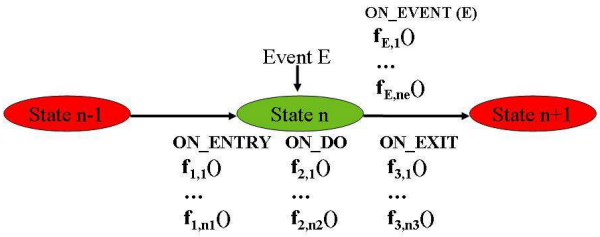

Fig 4: Actions and transitions in an ISEN sequence

Actions (noted $f_{E,1}$ to $f_{E,ne}$ in Fig 4) related to an event (noted *E* in the same figure) can also be attached to a state. When event *E* is received by the active state, the actions $f_{E,1}$, …, $f_{E,ne}$ are activated, in this order. And once these event-related actions are all activated, the control flow passes from the current state (*n*) to the state specified by the transition (i.e., state *n*+1 in Fig 4)--- the same mechanism applies for the actions of this new state. Any number of events (and transitions) can be attached to a state of an automaton. Therefore an automaton can exhibit any graph structure.

C. *Scenario generation*

A.I. task planners use operators, describing the actions which can be taken, an ontology-generated initial state and goals, to build a sequence of instantiated operators (a plan). A plan moves step by step a world state from the initial state to a final state that contains the goals [7]. All entities are described in the Planning Domain Definition Language (PDDL) syntax, e.g. version 2.1 [19] in our case. The shortest syntactic element is a fluent i.e., a term which can be positive or negative depending on the time of observation and on the operators' postconditions (potentially changing the truth value of this fluent) before this observation time. A fluent contains a functor followed by variables or constants, e.g., "*(position ?arm ?location)*", "*(at SAM kitchen)*". An operator is composed of preconditions (i.e., fluents which must hold in the incoming state for the operator to apply) and postconditions (i.e., fluents the truth value of which changes when compared to those of the incoming state).

CPT is a fast classical task planner, which turns a planning problem (i.e., the operators list, the initial state, the goals) into a constraint programming problem [22].

The actions which can be taken by the robot are gathered in the SAM domain: this is a symbolic representation in PDDL of these actions in terms of preconditions and post conditions (see [20] for a first version of this PDDL domain).

D. *Turning plans into scenarios*

A plan, generated by the task planner, is first parsed and each instantiated operator of this plan is identified to a state in a sequence. Encoding instantiated operators as event-triggered transitions (this way, ISEN states would be close to planning states) would prevent the robot from receiving action termination events during the execution of an action. So this option, although theoretically appealing, is not appropriate in practice.

Low level ISEN functions are used to turn this internal representation into ISEN's one. For each instantiated operator, the callback functions (the actions of section B) and the triggering event (to jump out of this ISEN state) are read from a handler file, by matching a handler name against the name of an operator. For example, the handler named "*position-arm-for-grasping*" is associated to the instantiated operator named "*position-arm-for-grasping pt-ref rot-ref kitchen pt-kitchen rot-kitchen*" --- CPT is canonical (an operator appears only once in a plan) [22], which prevents from matching the same handler to several instantiated operators with this name.

Finally, that internal representation is saved into a file describing all the states, transitions and sequences available: this is the generated scenario (a temporary file), which is dynamically re-loaded by ISEN for execution on the robot SAM.

Errors during the execution of an action are handled by a specific, always active, state, called "all_states", which is the default event-catcher. When this state is reached, it can branch to either CPT-generated states or additional states (not generated by CPT) dedicated to error recovery. These additional states in the handler file are the ones which do not match against the name of actions in the generated plan --- the remaining handlers, e.g., a handler named "FailureState" typically is not part of a plan but anyway maps to an ISEN state to which planned states can branch to for error recovery.

VI. Conclusion

The work presented in this paper consisted of developing an ontology representing the environment in which the robot will evolve. This ontology is used to generate the initial state of a planning problem, which together with PDDL operatiors and goals, produce an action plan through an A.I. task planner. Execution of scenarios is performed through an event-based finite state automate executor (ISEN). In the remainder, we will focus on the learning of new objects and new scenes by the robot, to enable him to understand the wishes of the dependant person and be able to generate a suitable scenario.

Acknowledgment

The authors thank Vincent Vidal (ONERA, Toulouse) and Christophe Leroy (CEA LIST LSI, Fontenay-aux-roses).

References

[1] K. Akachi, K. Kaneko, N. Kanehira, S. Ota, G. Miyamori, M. Hirata, S. Kajita, F. Kanehiro, "Development of humanoid robot HRP-3P". 5th IEEE-RAS International Conference on Humanoid Robots, pages 50-55, 2005.

[2] O. Bartheye, E. Jacopin, "A real-time PDDL-based planning component for video games". In Proceedings of 5th Artificial Intelligence for Interactive Digital Entertainment Conference, Stanford, California, 2009, pages 130-135.

[3] C. Dornhege, P. Eyerich, T. Keller, S. Trüg, M. Brenner, B. Nebel. "Semantic attachment for domain-independent planning systems". In Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS'09), Thessaloniki, Greece, 2009.

[4] J. Dumora. "Design of behaviors for a robot assisting handicapped persons" ("Conception de comportements pour un robot d'assistance aux personnes handicapées"). Technical Report, CEA, LIST, DTSI/SRI/08-XXX, August 2008, unpublished.

[5] R. Fikes, N. Nilsson, "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving", Artificial Intelligence, Vol. 2 (1971), pp 189-208.

[6] M. Fuchs, Ch. Borst, P. Robuffo Giordano, A. Baumann, E. Kraemer, J. Langwald, R. Gruber, N. Seitz, G. Plank, K. Kunze, R. Burger, F. Schmidt, T. Wimboeck, G. Hirzinger. "Rollin' Justin – Design considerations and realization for a humanoid upper body", in Proceedings opf the IEEE International Conference on Robotics and Automation (ICRA), Kobe, Japan, 2009, pages 4131-4137.

[7] E. Gat, "On three-layer architectures", D. Kortenkazmp et als eds., A.I. and Mobile Robots, AAAI Press, 1998.

[8] M. Ghallab, D. Nau, P. Traverso. "Automated planning : theory and practice". Morgan Kaufmann, Elsevier, San Francisco, 2004, 635 pages.

[9] D. Gouaillier, V. Hugel, P. Blazevic, C. Kilner, J. Monceaux, P. Lafourcade, B. Marnier, J. Serre, B. Maisonnier. "Mehatronic design of NAO humanoid". IEEE International Conference on Robotics and Automation, Kobe, Japan, 2009, pages 769-774.

[10] B. Graf, M. Hans,, R. Schraft, "Care-o-bot II development of a next generation robotic home assistant". Auton. Robots, 16(2), pp, 193—205, 2004.

[11] F. Ingrand, S. Lacroix, S. Lemai, F. Py, « Decisional autonomy of planetary rovers ». Journal of Field Robotics, vol. 24, n° 7, pages 559-580, 2007.

[12] International Planning Competition, http://ipc.icaps-conference.org/

[13] H.Iwata, S.Sugano: "Design of Human Symbiotic Robot TWENDY-ONE," Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA), pp. 580-586, 2009.

[14] F. Jammes, A. Mensch, H. Smit. "Service-oriented device communications using the devices profile for web services", In AINA Work., pages 947–955, Washington, USA, May 2007.

[15] M. Joint, P.-A. Moëllic, P. Hède, P. Adam, "HEKA : A general tool for multimedia indexing and research by content". 16th Annual Symposium Electronic Imaging (SPIE), San Jose 2004, Image Processing, Algorithms and Systems III.

[16] P. Laborie, M. Ghallab. "Planning with sharable resource constraints". International Joint Conference on Artificial Intelligence, Montreal, Canada, 1995, pages 643-1651.

[17] C. Leroux, I. Laffont, B. Biard, S. Schmutz, J. - F. Désert, G. Chalubert. « Robot grasping of unknown objects, description and validation of the function with quadriplegic people ». in Proceedings of the 2007 IEEE 10th International Conference on Rehabilitation Robotics. Noordwijk, The Netherlands, 2007.

[18] C. Leroy. ISENEdit: User Manual. CEA, LIST, LRI, Technical Report, 2005, unpublished.

[19] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, D. Wilkins. "PDDL – The Planning Domain Definition Language", http://cs-www.cs.yale.edu/homes/dvm/

[20] P. Morignot, M. Soury, C. Leroux, H. Vorobieva, P. Hède. 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). Singapore, December 2010, P1054.

[21] A. Remazeilles, C. Leroux , G. Chalubert, "SAM: a robotic butler for handicapped people", 17th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN - 2008), 01/08/2008-03/08/2008, Munich, Germany.

[22] V. Vidal, H. Geffner, "Branching and Pruning: An Optimal Temporal POCL Planner based on Constraint Programming", Artificial Intelligence 170 (3), pp. 298-335, 2006.

[23] H. Vorobieva, C. Leroux, P. Hède, M. Soury, P. Morignot, "Object Recognition and Ontology for Manipulation with an Assistant Robot", In Proceedings of the Eighth International Conference on Smart Homes and Health Telematics (ICOST'10), L.N.C.S., Aging Friendly Technology for Health and Independence, Springer, Berlin - Heidelberg, Germany, vol. 6159, 2010, pages 178-185.

[24] J. Wolfe, B. Marthi, S. Russell, "Combined Task and Motion Planning for Mobile Manipulation". International Conference on Automated Planning and Scheduling, Toronto, Canada, 2010.

[25] Minsky, M. & Papert, S., Perceptrons. MIT Press, Cambridge, MA, 1969.

[26] Quillan, M.R. Semantic Memory. In SIP, pp.216-270, 1986.

[27] Remazeilles, A., Leroux, C., Chalubert, G.: SAM: a robotic butler for handicapped people. In: IEEE RO-MAN, Munich, Germany (2008).

[28] Zied Loukil, Abdelmajid Ben Hamadou, Pierre Marquis, Vincent Vidal, " Les resources et la planification temporelle", 2005.

[29] Geffner H. and Haslum P., « Admissible heuristics for optimal planning », Proceedings of the Fifth International Conference on AI Planning Systems (AIPS-2000), 2000.

[30] W. Hwang et al. , "pp. 596 – 606, Springer-Verlag Berlin Heidelberg" 2006.

[31] K.M. Varadarajan and M. Vincze, "Ontological Knowledge Managment Framework for grasping and Manipulation",2011.

[32] M.Tenorth and M. Beetz, "KNOWROB — Knowledge Processing for Autonomous Personal Robots", 2009.

[33] Wonil Hwang and al. "Ontology-Based Framework of Robot Context Modeling and Reasoning for Object Recognition",2006.

[34] S Lemaignan and al. ,"ORO, a knowledge management platform for cognitive architectures in robotics",2010.