FP6-IST-002020

COGNIRON

The Cognitive Robot Companion

Integrated Project

Information Society Technologies Priority

D6.1.1
Specification of an architecture for a cognitive robot

Due date of deliverable: 31/12/2004
Actual submission date: 31/12/2004

Start date of project: January 1st, 2004 **Duration:** 48 months

Organisation name of lead contractor for this deliverable:
LAAS-CNRS

Revision: Final
Dissemination Level: PU

Executive Summary

Our efforts, during this first period, were dedicated to preparatory work for building a robot control architecture:

- that integrates a probabilistic approach at various levels: comparison of several models for implementing decision in presence of uncertainty.
- that provides a framework for implementing human-robot interaction in a systematic way

This is based on a study of relevant literature and the investigation of various interaction schemes.

We also developed and accumulated some experience through the construction of human-robot interactive scenarios involving a mobile robot, called Rackham, equipped with vision and several interaction modalities (speech, tactile screen, virtual reality).

Role of the topic in Cogniron

Cf. next section.

Relation to the Key Experiments

The work described here will conduct to the development of concepts and algorithms that will be implemented and illustrated in the Key Experiments: a control architecture for a cognitive robot that provides a framework for implementing human-robot interaction in a systematic way.

1 Topic of the deliverable

1.1 Introduction

The Research Activity 6 deals with system organization, decision-making, task distribution, attribution of intentionality and expression of robot intentions.

It addresses the decision-making mechanisms that are induced by the demanding context of scenarios for a personal robot. Two key topics are crucial: (1) evolution and learning in an open environment and (2) interaction with humans.

This approach requires architectural aspects and decisional abilities to be revisited and adapted to provide a framework, and to elaborate the models and paradigms that should allow the robot: to take into account context and user dependencies while performing its tasks to take the initiative, and then establish and conduct an interactive session with a human to adopt various interaction styles depending on the context and allow the human to act physically the robot (push, guide...) but also ”influence” its on-going decisional processes

The main topics that we propose to tackle in a coherent and constructive approach are Control architectures, Context dependent Task refinement, Human-robot interaction, Social interaction, Collaborative problem solving.

Our investigation will follow two streams that will be developed in close interaction with the other topics of the project:

- A control architecture for a robot companion
- A scheme for interactive human-robot problem solving.

Workpackage 6.1 deals with the first topic.

The purpose of this workpackage is not to develop a new control architecture. The control architecture principles that we will apply will be based on the state of the art. Our target is to provide a framework that will allow to integrate in a coherent fashion context-dependent task refinement and human-robot social interaction.

This study has been conducted and will be pursued, based on two aspects:

- the necessity to take explicitly uncertainty into account various levels
- the study of the task-oriented human-robot interaction processes induced by the scenarios elaborated by the key experiments, in order to have a constructive and convergent perspective.

1.2 Summary of work

Our efforts, during this first period, were dedicated to preparatory work for building a robot control architecture:

- that integrates a probabilistic approach at various levels: comparison of several models for implementing decision in presence of uncertainty.
- that provides a framework for implementing human-robot interaction in a systematic way

This is based on a study of relevant literature and the investigation of various interaction schemes.

We also developed and accumulated some experience through the construction of human-robot interactive scenarios involving a mobile robot, called Rackham, equipped with vision and several interaction modalities (speech, tactile screen, virtual reality).

2 Rackham: an interactive autonomous robot

We have designed and implemented a new tour-guide robot. Besides robustness and efficiency in the robot basic navigational abilities in a dynamic environment, our focus was to develop and test a methodology to integrate human-robot interaction abilities in a systematic way.

To test and validate our developments, we have decided to bring regularly our robot to a museum in Toulouse (by regularly, we mean two weeks every three months). The robot, called Rackham, has already been used in an exhibition for hundreds of hours (July 2004, October 2004), accumulating valuable data and information for future enhancements. The project is conducted so as to incrementally enhance the robot functional and decisional capabilities based on the observation of the interaction between the public and the robot.

The paper, in the annex, presents the robot and some of its key design issues. It also discusses a number of lessons that we have drawn from its use in interaction with the public. and that will serve to refine our design choices and to enhance the robot efficiency and acceptability.

The software architecture is an instance of the LAAS[1] architecture ([1]). It is a hierarchical architecture including a supervisor written with openPRS[2] (a Procedural Reasoning System) that controls a distributed set of functional modules.

A number of features have been installed for human-robot interaction:

- the detection of dynamic “obstacles” ,
- a vision-based face detector,
- an animated face with speech synthesis,
- displays and inputs from the touch screen,
- control of robots lights.

While the two firsts allow to detect the presence or the departure of people, the last ones permit the robot to “express” itself and thus establish exchanges.

[1]LAAS stands for: “LAAS Architecture for Autonomous Systems”.

[2]The set of tools used to build an instance of this architecture (GenoM, openPRS, pocolibs, etc) are freely distributed at the following url: http://softs.laas.fr/openrobots.

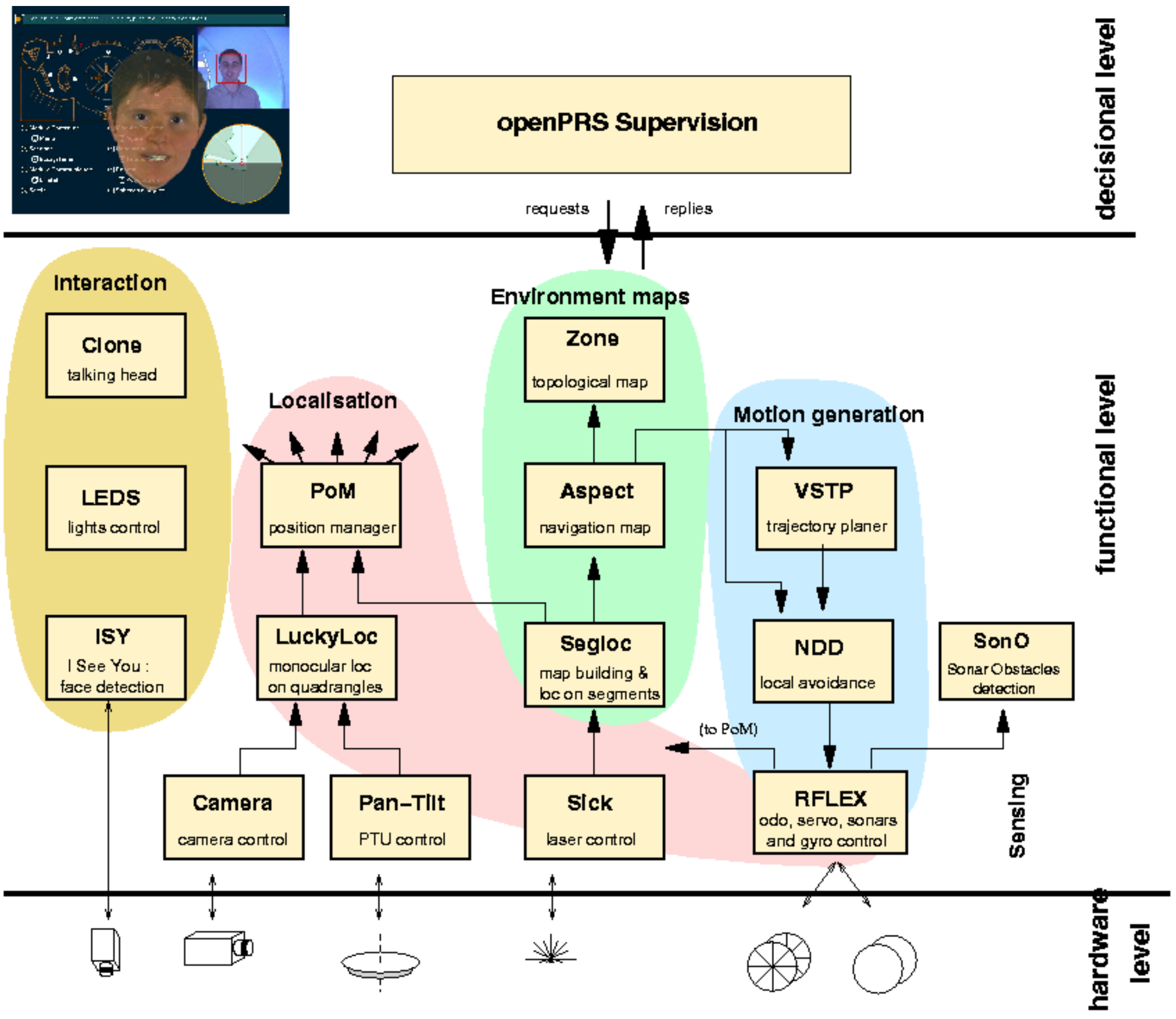

Figure 1: The functional level of Rackham and its 15 modules.

The vocal synthesis is highly enriched by a 3D animated head displayed on the screen. This talking head, or `clone`, is developed by the Institut de la Communication Parlee (http://www.icp.inpg.fr). It is based on a very accurate articulatory 3D model of the postures of a human speaker with realistic synthetic rendering thanks to 3D texture projection. From a given text, the speech synthesizer produces coordinated voice and facial movements (jaw, teeth, lips, etc.).

The directions of the head and of the eyes can be dynamically controlled. This capability is important as it allows to reinforce an interaction, looking towards the interlocutor face detected , or to point out an object or a part of the exhibition currently mentioned by the robot.

The clone appears in front of the touch-screen each time the robot has to speak.

3 Specification of an architecture for a cognitive robot

We discuss in the following several questions of interest from the literature as well as from our experience in developing an interactive autonomous robot.

3.1 The context representation

Control architectures have been a very hot subject in the past decade. This debate has now reached a certain level of maturity. However, there are still several open issues.

It is said in ([17, 10]) that one of the problem of the three level architecture is that "each level has its own representation and thus several different models of the robot and its environment must be created and maintained. This representation of information storage is often redundant and causes additional overhead in maintaining the system over time".

We do not think that the problem lies in the three layer architecture itself. It may happen if the information is not available at the right level because it is hidden in some process or has been considered a priori as non relevant. This was exhibited, for instance, in the tour-guide robot implementation.

The lessons drawn showed that it was a priori useful to provide as much context information as possible to the robot supervision level. Indeed, context-based task refinement is essential for autonomous robots but even more for cognitive robots in human environments.

Another aspect that we have begun to tackle in the ability to integrate learning capabilities [8] in a modular general-purpose robot architecture.

3.2 Human-robot interaction

The Interaction strongly depends on the persons the robot interacts with. The robot needs to model the interacting person in terms of abilities, preferences, etc..

The Joint Intention theory [4, 5] is an interesting concept to model how a robot may cooperate with humans (as a team member). It has been applied in robotics application [3], or even in situations where the activity of heterogeneous agents (persons, robots, software agents) [13, 14] has to be coordinated.

This theory defines that, for a team θ a joint intention $JPG(\theta, p, q)$ holds iff :

1. All team members mutually believe that p is currently false.
2. All team members have p as their mutual goal, i.e. they mutually know that they want p to be eventually true and that its relevance is expressed by a scalar value q.
3. All team members mutually believe that until p is mutually known to be achieved, unachievable or irrelevant, they mutually believe that they each hold p as a weak achievement goal (WAG).

For team member μ, $WAG(\mu, p, q, \theta)$ implies that one of the following holds :

- μ believes p is currently false and wants it to be eventually true, i.e. p is a normal achievement goal or
- having privately discovered p to be achieved, unachievable or irrelevant (because q is false), μ has committed to having this private belief become θ's mutual belief.

Notice that *mutual belief* is an interesting notion in itself, representing what an agent believes about what the other believes. It should be implemented somehow at the robot level in order to allow us to model what the robot knows the human knows about it. For instance, if the robot knows the person it interacts with does not know what it is able to do and what it currently does, it has to give much more information, explanation than otherwise.

One recent generic implementation of the joint intention theory is Machinetta [14]. In this implementation each team member has a "proxy" that represents it in team collaboration. The Machinetta proxy software is made up of five components :

- Communication: communication with other proxies
- Coordination: reasoning about team plans and communication
- State: the working memory
- Adjustable autonomy: reasoning about whether to act autonomously or pass control to the team member
- RAP (robots, agents and people): communication with team member.

A team of proxies implements "Team Oriented Plans" (TOPs), a team-level description of the activities that need to be performed in order to achieve the goals of the team.

This is certainly a rich source of inspiration for us in our construction of a framework for human-robot decisional interaction. Each person or group of persons that the robot decides to interact with may be modeled by it as particular instance of an agent, with specific information about its desires, intentions and goals. Such information will be acquired and maintained through perception and communication (in a large sense). However, we will discuss next what differences we see concerning a robot acting in a real world particularly versus the human agent commitment.

3.3 Task achievement

Once the robot knows what it has to do, comes the execution time. The execution of an action/task must be as flexible as possible. It heavily depends on the execution context and consequently it has to be defined at a (sufficiently high) level that allows to take into account the execution context at the right moment.

Let us take, for instance, task "communicate an information to a person i'm committed with". The communication mean itself whether the atmosphere is noisy or quiet, whether the robot has to talk to one person or to a group... What we argue is that the action in the plan has to be at the level of: "communicate an information to the people",that is effectively the goal it is committed to. But at the execution time the robot have to be able to choose the best manner to realize this task given the context.

The task is performed following a policy (by policy we mean "a solution that specifies what the agent should do for *any* state that the agent might reach"). Considering the complexity and the means available for the task, this policy could be a simple finite state automaton or a complex POMDP policy, computed on-line or offline according to the robot perception and action abilities.

Indeed, in accordance with [2, 16], decision-theoretic planning in general and Partially Observable Markov Decision Process in particular are the standard mechanism of choice for representing problems where an agent must reason under uncertainty. POMDPs allow for optimal reasoning under the conditions of both action and state uncertainty. Action uncertainty exists when an agent only knows a probability distribution over the possible outcomes of its actions. State uncertainty exists when the observations about the world available to an agent do not provide the exact world state, but rather indicate a probability distribution over the possible states. It is this state uncertainty that both allows POMDPs to represent realistic problems and causes significant computational complexity.

To deal with this complexity, hierarchical representations have been proposed. For instance, Pineau's work [11] provides an algorithm for finding approximate solutions to POMDPs through action abstraction. Theocharous's work [15] is based on a state abstraction. Action hierarchy allows to plan at different levels of abstraction whereas state hierarchy allows to consider only the given or relevant information in a particular context.

3.4 Architecture framework

For an interactive robot, if we want to build it in a generic re-usable way, it is necessary to implement an "explicit meta-level", i.e. a level which will allow to reason explicitly on its ability and the human ability (in a given context and application) and with an explicit management of the interaction.

We will build such a meta-level by an adaptation of the joint intention theory to our needs.

The robot and the human: In the Joint Intention Theory, as we have see before, once a team member commits to a joint goal, it is also committed to maintain mutual belief towards the other team members about this goal.

The fact that an agent, in our system, could be a human will change many things. One of the most important is that when a human commits to some goal, he will not regularly inform the robot when it considers the goal achieved for him or when he wants to cancel his commitment. So, depending on the "trust" the robot will have the person, the degree of collaboration needed, etc, the robot will have to check by itself (through various modalities) whether the goal is still alive or not.

The robot decisional kernel will implement agents that will be a way to represent the beliefs and mutual beliefs of the humans. We propose to call them "interaction-agents". It will be in fact the human representation (at decisional level) for the robot. Even if a human intervention must be preemptive, it will just have a reinforcement role.

An interaction-agent: An “interaction agent” will be created when the robot detects a person or a group of persons. This interaction-agent will be instantiated with the current knowledge the robot has.

The robot will have several models of an interaction-agent. Such models will include: the means of communication, the actual level of commitment, It will also include information such as :

- type: a group or not, kid
- identification: do i know this person
- communication abilities
- confidence
- commitment
- ...

At the beginning, at the agent creation when the robot has no knowledge about the person, the model will express only the basic properties of a human in the given context.

An important point concerning the interaction agent is that its not a static agent but a way for the robot to represent its beliefs and their evolution over time (whether i am engaged with this person or not...).

Commitment establishment An procedure to establish commitments will be launched if:

- an interaction-agent asks something to the robot,
- the robot has another goal (a very high-level general goal that defines its basic utility functions) and based on this goal it decides to engage with an interaction-agent.

The robot has to consider if it can do things by itself or if it need to trigger a joint plan.

During the procedure of commitment establishment, each partner (the human, the robot) can commit or deny. The procedure by itself may have various modalities and may be quite trivial (the simple creation of the goal, everyone ever knowing what he/it has to do) or may call for the joint elaboration of a plan with associated role allocation (see Deliverable 6.2.1).

The joint goal In fact we use the term joint goal in a very large sense, that is in fact every goal of the robot will be a commitment towards somebody (towards the software engineer at a basic level)

Once the goal is created, the robot accomplishes its part (almost all of the task) and if needed, checks the commitment of the human (particularly when it is a collaborative task, ex: the human must follow the robot to go somewhere). The robot has to check the human commitment which is by definition only partially observable.

3.5 An example from the "curious robot" scenario

We illustrate here below some examples borrowed from Key experiment 2.

In script 1: Robot initiative, the robot anticipates a situation that may occur and acts in order to facilitate the future action of the person.

A person is sitting and busy with a task. The robot is doing its own tasks and also observes the human who would be doing some gestures, not meant towards the robot, which usually express a need for a drink (for example, the human "plays" with an empty cup).

- First case: the robot interrupts its own task, approaches the human and asks her if she wants a drink. Upon a positive answer, the robot fetches it.
- Second case: the robot does not ask.
- Third case: there is already a can on the table, but it is too far to be reached by the person. The robot takes the initiative to place it closer to the person.

We will now explain how this three cases may be dealt with, based on the framework discussed above.

First case: The robot creates an interaction-agent when it detects the person sitting and busy with a task. It has limited knowledge about the person. Consequently, it will interact with her using the basic social rule of human-robot interaction. So taking the initiative, it tries to establish a commitment with the person, asking her before doing anything else. Upon a positive answer, it creates a joint goal: it is committed to bring a drink and the person is committed to want a drink to be brought to her.

In the current context, the robot considers that the best way to bring a drink is to fetch it from refrigerator and it does. A negative gesture from the person during the phase where the robot hands the drink suspends or even stops the global goal.

Second case: The robot creates an interaction agent when it detects the person sitting and busy with a task. It then recognizes Bill. It knows Bill and knows how to interact with him. So taking the initiative, and knowing how to interact with Bill, the establish commitment phase creates a joint goal without asking Bill: the robot is committed to bring a drink and Bill is committed to want a drink to be brought. The establish commitment phase in this case will be transparent to Bill. In the current context, the robot considers that the best way to bring a drink etc ..

Third case: The robot creates an interaction-agent when it detects the person sitting and busy with a task. It has limited knowledge about the person. Consequently, it will interact with here using the basic social rule of human-robot interaction. So taking the initiative, it tries to establish a commitment with the person, asking her before doing anything else etc..

In the current context, there is already a can on the table (but it is too far to be reached by the person) the best way to bring a drink is to place it closer to the person.

4 Future Work

In the next period, we will refine the proposed concepts and investigate their applicability. This will be done on the basis of contexts and tasks defined in the key experiments (essentially KE 1 and KE 2)

The second workplan will aim at designing and developing an architecture based on these key ideas and that will also take into account considerations studied in WP6.2 and WP6.3. Indeed, it will be necessary to export the essential supervision data (e.g. current goals, tasks, current task/subtask hierarchy, main control parameters...) that may be used in cooperative problem solving or to exhibit robot intentions.

Another aspect that has not yet been tackled is linked to the initiative issues.

The last point is the coordination with RA1, RA4 and RA5 in order to build a framework able to incrementally include representations of environment knowledge and task knowledge that are elaborated within the project.

5 References

References

[1] R. Alami, R. Chatila, S. Fleury, M. Ghallab, F. Ingrand, “An architecture for autonomy”, *International Journal of Robotic Research*, Vol.17, N°4, pp.315-337, Avril 1998.

[2] Craig Boutilier and Thomas Dean and Steve Hanks, “Decision-Theoretic Planning: Structural Assumptions and Computational Leverage,” *Journal of AI Research (JAIR)* , 1999.

[3] C. Breazeal, et.al., “Humanoid robots as Cooperative partners for people” Submitted to IJHR(2004).

[4] P.R. Cohen and H.J. Levesque. “Intention is choice with commitment”. *Artificial Intelligence*, 42(3), 1990.

[5] P.R. Cohen and H.J. Levesque. “Teamwork”’. *Nous*, 25(4), 487-512.

[6] S.Fleury, M.Herrb, R.Chatila, “GenoM: a Tool for the Specification and the Implementation of Operating Modules in a Distributed Robot Architecture”, *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Grenoble, France, 1997.

[7] T. Fong, I. Nourbakhsh, K. Dautenhahn, “A survey of socially interactive robots,” *Robotics and Autonomous Systems, Special issue on Socially Interactive Robots*, 42(3-4), 2003.

[8] Steffen Knoop, Steffen Vacek, Raoul Zollner, Christian AU, Rudiger Dillman,“A CORBA-Based Distributed Software Architecture for Control of Service Robots”,IROS 2004.

[9] N. Muscettola, G. Dorais, C. Fry, R. Levinson, C. Plaunt, “IDEA: Planning at the Core of Autonomous Reactive Agents” in *Proceedings of the 3rd International NASA Workshop on Planning and Scheduling for Space*, October 2002.

[10] I.A.D. Nesnas, A. Wright, M. Bajracharya, R. Simmons, T. Estlin, “CLARAty and Challenges of Developing Interoperable Robotic Software,” *IROS’03 International Conference on Intelligent Robots and Systems*, Nevada, October 2003.

[11] J. Pineau and S. Thrun. “An integrated approach to hierarchy and abstraction for POMDPs,” *Technical Report CMU-RI-TR-02-21* , Carnegie Mellon University, 2002.

[12] S. Russell and P. Norvig, *Artificial Intelligence : A Modern Approach,* Prentice Hall, 2003.

[13] M. Tambe. “Agent architectures for flexible, practical teamwork”. *Journal of Artificial Intelligence Research*, 7, 83-124.

[14] P. Scerri and D. Pynadath and N. Schurr and A. Farinelli and S. Gandhe and M. Tambe. “Team Oriented Programming and Proxy Agents: The Next Generation”, In Proceedings of 1st international workshop on Programming Multiagent Systems, Springer, LNAI 3067, 2004.

[15] Georgios Theocharous, Kevin Murphy, Leslie Kaelbling, ‘’Representing hierarchical POMDPs as DBNs for multi-scale robot localization,” *ICRA’04 (Intl. Conf. on Robotics and Automation)* , New Orleans, LA, USA, 2004.

[16] Turkett, Jr., W. and Rose, J. , "Planning With Agents: An Efficient Approach Using Hierarchical Dynamic Decision Networks," *Proceedings of the Fourth International Workshop on Engineering Societies in the Agent World* , London, England, 2003.

[17] R. Volpe and I.A.D. Nesnas and T. Estlin and D. Mutz and R. Petras and H. Das, "CLARAty: Coupled Layer Architecture for Robotic Autonomy," *JPL Technical Report D-19975* , Dec 2000.

5.1 Reference documents

- Deliverable D.6.2.1, "Report on Paradigms for Decisional Interaction", Cogniron.
- Deliverable D.7.1.1 on Cogniron Key Experiments.

Annexes

- Paper presented at IROS 2004: "A CORBA-Based Distributed Software Architecture for Control of Service Robots", Steffen Knoop, Steffen Vacek, Raoul Zollner, Christian AU, Rudiger Dillman.
- Paper submitted to ICAR 2005: "Supervision and Interaction: Analysis from an Autonomous Tour-guide Robot Deployment", Aurelie Clodic, Sara Fleury, Rachid Alami, Matthieu Herrb, Raja Chatila.

Supervision and Interaction

Analysis from an Autonomous Tour-guide Robot Deployment

Aurélie Clodic, Sara Fleury, Rachid Alami, Matthieu Herrb, Raja Chatila
LAAS - CNRS
7, Avenue du Colonel Roche,
31077 Toulouse, France
Email: Aurelie.Clodic@laas.fr, Sara.Fleury@laas.fr, Rachid.Alami@laas.fr, Matthieu.Herrb@laas.fr, Raja.Chatila@laas.fr

***Abstract*— This paper presents the design and the implementation of a new tour-guide robot and reports on the first results that have been obtained after its deployment in a permament exhibition. The project is conducted so as to incrementally enhance the robot functional and decisional capabilities based on the observation of the interaction between the public and the robot.**

Besides robustness and efficiency in the robot basic navigational abilities in a dynamic environment, our focus was to develop and test a methodology to integrate human-robot interaction abilities in a systematic way.

We first present the robot and some of its key design issues. Then, we discuss a number of lessons that we have drawn from its use in interaction with the public. and that will serve to refine our design choices and to enhance the robot efficiency and acceptability.

I. Introduction

Today, one of the challenges of robotics is to have robots that achieve long terms missions outside of their laboratories and are actually helpful to human.

Rhino[4] and Minerva[17] have been the precursors of series of tour-guide robots in various museums and exhibition halls [15], [11]. Those robots had various degrees of autonomy and were using more or less sophisticated techniques. However, they have all pointed out that studying human-robot interaction was necessary, in its definition as well as its implementation.

It has standed out that robots must obey to some “social” clues [6]. And it has led to the development of service robots (e.g. Pearl [12], Care-O-bot II [7], CERO [8], Lino [9] and BIRON [19]).

To study human-robot interaction, an experimentation environment must be found, out of a laboratory and its standard rooms and halls. . . and its robotics scientists who know very well how their ”creations” work. To test and validate our developments, we have decided to bring regularly our robot to a museum in Toulouse (By regularly, we mean two weeks every three months). The robot, called Rackham, has already been used in an exhibition for hundreds of hours, accumulating valuable data and information for future enhancements. The project is conducted so as to incrementally enhance the robot functional and decisional capabilities based on the observation of the interaction between the public and the robot.

Besides robustness and efficiency in the robot basic navigational abilities in a dynamic environment, our focus was to develop and test a methodology to integrate human-robot interaction abilities in a systematic way.

In this paper, we describe this tour-guide robot. We begin with a presentation of the exhibition context. Next, we presents the LAAS architecture([1]) and the various tools already developed by our group to implement an instance of it for Rackham. After, we present the means to supervise all the system, to finish with experimental results, comments and analysis.

II. The Experimental Context and Scenarii

A. Mission Biospace

Mission BioSpace is an exhibition elaborated by the “Cité de l’Espace”[1] at Toulouse to illustrate what could be an inhabited spaceship. It presents about 14 interactive elements from “Lexigraph” to “Teleportation” that propose visitor a vision of the futur.

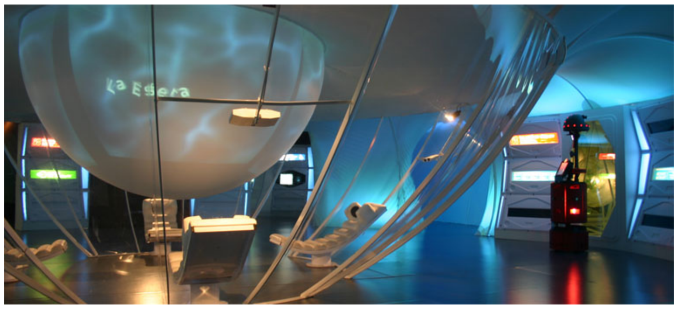

Fig. 1. The Tsiolkovski spaceship: A very difficult environment context for navigation.

B. A difficult context for navigation and interactions

The exhibition is simulating the interiors of a spaceship (25x10 square meters) with all its visual and acoustic atmosphere. Hence it presents itself as a difficult context for navigation and interaction (see picture 1):

- ambiant noises : speech synthetisis is difficult to hear
- the room is dark with changing colors : camera vision and face detection. . .
- round shapes everywhere, not easy to deal with using a model based on line segments

[1]http://www.cite-espace.com

- top and down prominent obstacles: not visible by the sensors and they can be hit by the head or the bottom part of Rackam
- some translucent obstacles : not perceived by the laser range finder
- Some narrow passages, which require a precise positionning of the robot to navigate through them

C. *A typical Rackham mission*

When Rackham is left alone with no mission, it looks forward to find out people to interact with. As soon as a person is detected, thanks to visual face detection, it presents itself through the virtual 3d face “I'm Rackham and I can guide you in the spaceship” or alternatively explains how to use its services : “Select your destination using the touchscreen”.

If the visitor finally selects a destination Rackham first confirms its new mission “OK, I will guide you to...” and plans the adequate trajectory. Once available, this trajectory is displayed on the screen and Rackham invites people to follow it.

While navigating, the robot keeps on giving information about the progress of the on going travel : a congestion will require to temporarily stop or even to compute an alternative trajectory while a given level uncertainty on the position might call for a relocalisation procedure; sporadic “disappearance” of the guided visitor are also detected and dealt with using declarations such as ”Where are you ?”,”Here you are again!”. The visitor may by himself stop and change the ongoing mission whenever he wants using various buttons displayed on the interface.

III. Rackham

A. *The robot*

Rackham is a B21r robot made by iRobot. It is a cylindar robot 4' (52cm) tall and with a diameter of 20” (118cm). It integrates 2 PCs running respectively one P3 at 800MHz and two P3 at 1GHz. We have extended the standard equipement with one pan-tilt sony camera EVI-D70, one digital camera mounted on a Directed Perception pan-tilt unit, one ELO touch screen, a pair of loudspeakers, an optical fiber gyroscope and wireless ethernet.

In order to integrate all these components in a robust and pleasant way the “Cité de l'Espace” has designed a “head” on a mast, the whole toped by an helmet which represent something between a one-eyed modern pirate and an african art statue (see picture 2). The eye is materialized by the EVI-D70 camera fixed upside-down above the helmet, the second camera is hiden in the helmet and one loudspeaker is within what represent the mouth. The “nose” is only decorative (see picture 6).

The mast has been designed as high as possible to keep away the cameras from children hands.

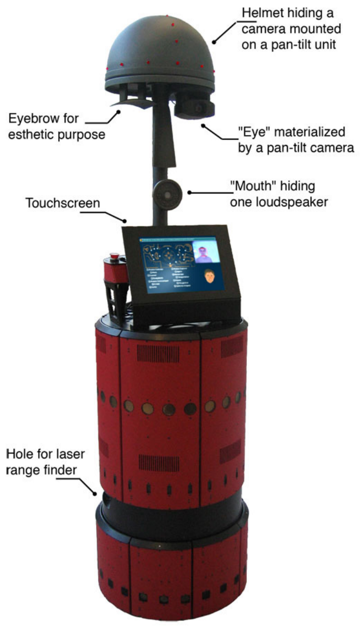
Helmet hiding a camera mounted on a pan-tilt unit
Eyebrow for esthetic purpose
"Eye" materialized by a pan-tilt camera
"Mouth" hiding one loudspeaker
Touchscreen
Hole for laser range finder

Fig. 2. Rackham and its equipement.

B. *The software architecture*

The software architecture is an instance of the LAAS[2] architecture ([1]). It is a hierarchical architecture including a supervisor written with openPRS[3] (a Procedural Reasoning System) that controls a distributed set of functional modules.

A module is an independent software component that can integrate all the operational functions with various time constraints or algorithm conplexity (control of sensors and actuators, servo-controls, monitorings, data processings, trajectory computations, etc.).

Each module is created using the generator of module GenoM and thus presents standard behavior and interfaces (see [5] and footnote 3). The functions encapsulated in a module can be dynamically started, interrupted or (re)parameterized upon asynchronous standard requests sent by the supervisor.

Once started, a service runs autonomously. A final reply that qualifies how the service has been executed is returned to the supervisor with the end of the service. During the execution a module can export data in structured public entites call posters and read data from posters produced by other modules (eg, robot positions, trajectories, maps and so on). The set of posters represent a distributed database of the state of the functional level of the architecture.

For the considered application, we have implemented 15 modules. We now present them according to their purpose in the system (see figure 3).

1) Localization: Several modules are involved in the localization of the robot.

[2]LAAS stands for: “LAAS Architecture for Autonomous Systems”.

[3]The set of tools used to build an instance of this architecture (GenoM, openPRS, pocolibs, etc) are freely distributed at the following url: http://softs.laas.fr/openrobots.

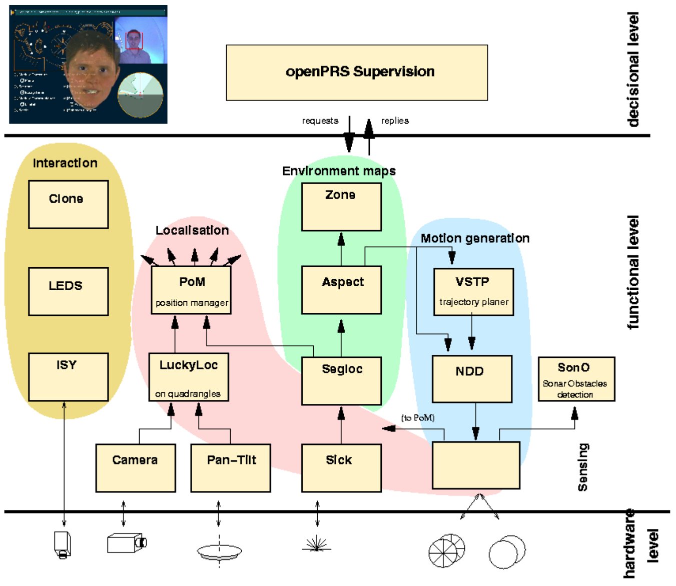

Fig. 3. The functional level of Rackham and its 15 modules.

First the `rflex` module, which interface the `rflex` driver provided by iRobot, exports in a poster the position computed by the odometry and corrected by the gyroscope. This position is associated with a covariance matrix deduced from a probabilistic model error. This position gives a good estimate of the motions of the robot.

To localize itself within its environment the robot uses a SICK laser, controlled by the module `sick`, that exports at the required rate the laser echoes, and segments deduced from aligned echoes. Another module, `segloc`, is able to match these segments with segments previously recorded in a map thanks to a classical SLAM procedure. However the map is effectively updated only during closing time. The resulting map is composed of 232 segments (see figure 4).

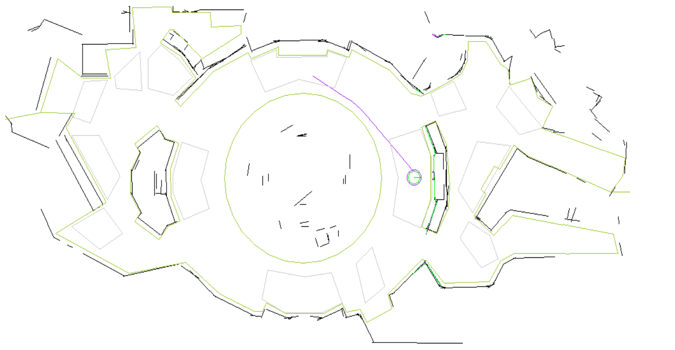

Fig. 4. The map of the environment build by the Rackham contains 232 segments (black) and has been augented with virtual obstacles (green or darkgrey) and target zones (lightgray).

The localization beeing a very critical hability, a third localization procedure, based on vision, has been designed. It consists on the identification of the furniture of the spaceship with one color camera. The camera is controled by the module `camera` that produces images themselves analysed by the module `luckyloc` that extracts, identifies and localizes planar quadrangles. However, if `luckyloc` is already able to identify the various pieces of furniture, the localisation procedure is not yet totally functional.

Finally, the various uncertain positions exported by the modules `rflex`, `segloc` and `luckyloc` are merged by `pom`, the position manager module. The module `pom` is able to integrate positions computed at various frequencies and even to propagate "old" position data (because of the time taken to acquire and process the data). Various fusion strategies can be selected like Kalman fusion or integration of the measured motions relatively to the most confident positions. The supervisor can be informed in case of localization problems with one of the module, fusion difficulties or significant uncertainties on the position. According to the problem, various strategies are applied.

It is important to notice that the `pom` module allows to centralize the robot positions and to export one and only one reference position. All the others parts of the system do not need to care how this position is obtained. This procedure can change dynamically without disturbing the position consumers. It is a very a important mechanism to manage redundancy, an essential feature for this critical function.

On top of this geometric positioning, several topological zones corresponding to places of special interest ("TARGETS"), to dangers for the navigation ("OBSTACLES" not always visible by the robot sensors like prominent or transparent furniture), or to other special areas ("SPECIAL") has been defined in the environment. The `zone` module continously monitors the entrances and the exits of the robot from these zones and informs the supervisor.

2) Obstacles and people detection: The obstacle detection is also a very critical function both for security reasons and for interaction purposes. The most efficient sensor is once again the laser. However the laser can only look forward (over 180 degrees) in an horizontal plan and echoes are poor data.

To overcome part of these limitations, the laser data are integrated in a local map by the `aspect` module and filtered using knowledge about the map, its segments and the virtual obstacles. In the context of Mission BioSpace with prominent and see-through obstacles this notion of virtual obstacles is very important.

Finally `aspect` exports every 40 miliseconds a local map all around the robot which represents the free space and which distinguishes static (ie, that belong to the environment or the virtual obstacles) and dynamic obstacles (probably visitors).

This local map is permanently displayed on the bottom right of the interface (see figure 5).

Using this representation, `aspect` is able to inform the supervisor when the robot is surrounded by unpredicted obstacles. The red leds on the helmet flicker at a frequency proportional to the obstruction density by dynamic obstacles.

To reinforce the assumption of presence near the robot, the supervisor can use the services of the `sono` module that detects motion all around the robot using the ultrasonic sensors. Unfortunatly those ultrasonic sensors produce some audible(!) noise which seem to disturb visitors interacting with the robot.

A much more robust people detector is offered by the module called `isy` (or, "I See You") which is able to detect a face in real time from one color camera image. The detector uses a cascaded classifier and a head tracker based on a particle filter (see [3]).

`Isy` controls the camera orientation in order to follow the detected face as long as possible. It informs the supervisor when it catches or looses a face. It is not (yet) a real tracking procedure but the detection procedure being fast (xxHz) the bahavior is very satisfactory.

From the direction and the size of the face it is able to estimate the 3D position of the detected person with a sufficient precision (about 10cm for the height and 20cm for the range).

The very weak and color changing sourrouding lights were a real problem. Thanks to a ring of white leds fixed around the lens the range of the detection is about 3 meters.

3) Trajectory and motion: Rackham being a guide, it must be able to take visitors to interest places of the exhibition. These places are displayed on the interactive map. For the robot they are not just a cartesian point but a `target zone` (see §III-B.1) made of a polygonal zone, the position the element of interest of this place (which can be itself out of the polygon) that the robot will have to comment, and an indicative target position within the zone.

The robot motion implies mainly three modules :

- `rflex` that manages the lower servo-control loop, transmiting the reference speeds at the micro-controller. It also checks the freshness of the produced references.
- `ndd` integrates a local avoidance procedure based on an algebraic instance of Nearness Diagrams (see [10]). The input obstacles are the those of the aspect map (see §III-B.2).
- `vstp` is a Very Simple [but very efficent] Trajectory Planner based on an algebraic visibility graph optimized with hash tables[4]. A main visibility graph is pre-computed for the static segments of the map. The dynamic obstacles can be added and removed in real-time upon supervisor requests.

The strategy used to coordinate the implied modules is dynamically established by the supervisor. The objective is of course to reach the target zone while avoiding obstacles. The planned trajectory is an Ariadne's clew for `ndd`: the vertices of the broken line are sub-goals. Usally the supervisor has to intervene only if `ndd` does not progress anymore along this path. In such a case, the strategies could be various: computing of a new trajectory taking into account the uncountered obstacles, waiting for a while, starting an interaction with people arround, etc. The motion is over when the robot is inside the target zone (the indicative target position can be occupied by visitors).

[4]VSTP is freely distributed: http://softs.laas.fr/openrobots/.

The maximum speed that the robot can achieve in this mode is about 0.6 meters per second.

4) Interactions: For now the interactions are mainly established through the following components:

- the dynamic "obstacles" detectors (`aspect` and `sono`),
- the `isy` face detector,
- an animated face with speech synthetisis,
- displays and inputs from the touch screen,
- control of the robots' lights.

While the two firsts allow to detect the presence or the departure of people, the last ones permit the robot to "express" itself and thus establish exchanges.

The vocal synthesis is highly enriched by a 3D animated head displayed on the screen. This talking head, or `clone`, is developped by the Institut de la Communication Parlée (http://www.icp.inpg.fr). Their initial motivations were the communication for hard of hearing (thus well adapted in noisy environments!). In robot context, the hability to interact with an animated and human-like entity participate hightly to the quality and the richness of the communication.

The clone is based on a very accurate articulatory 3D model of the postures of a speaking locutor with realistic synthetic rendering thanks to 3D texture projection. From a given text, the speech synthetizer produces coordinated voice and facial movements (jaw, teeth, lips, etc.).

The directions of the head and of the eyes can be dynamically controlled. This capability is important as it allows to reinforce an interaction, looking towards the interlocutor face detected by `isy`, or to point out an object or a part of the exhibition currently mentioned by the robot.

The clone appears in front of the touchscreen each time the robot has to speak (see picture 5.

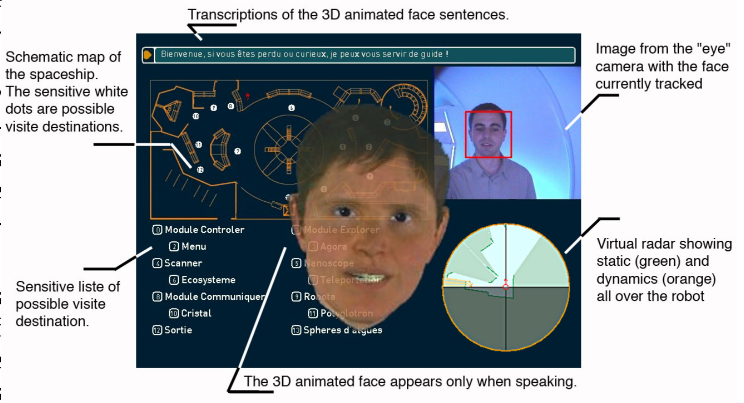
Transcriptions of the 3D animated face sentences.
Bienvenue, si vous êtes perdu ou curieux, je peux vous servir de guide !
Schematic map of the spaceship. The sensitive white dots are possible visite destinations.
Image from the "eye" camera with the face currently tracked
0 Module Controler
2 Menu
4 Scanner
6 Ecosysteme
8 Module Communiquer
10 Cristal
12 Sortie
1 Module Explorer
3 Agora
5 Nanoscope
7 Teleportation
9 Robota
11 Polyglotron
13 Spheres d algues
Sensitive liste of possible visite destination.
Virtual radar showing static (green) and dynamics (orange) all over the robot
The 3D animated face appears only when speaking.

Fig. 5. A view of the interface of the touchscreen.

The robot interface, written with java, is made of independant components or microGUI directly controlled by the supervisor through a dedicated communication channel.

The available microguis are :

- a map of the environment including the current robot position and trajectory
- the local "aspect" map displayed as a radar

- the image of the “eye” camera with the faces currently detected by isy
- the clone or talking head
- pop-up warning messages
- top messages
- localization window (init).

The data flow is directly coming from the posters of the modules after an automatic transcription by GenoM in an `xml` format of the data.

5) Controls of the functional level: This set of modules offers a good degree of redundancy for several functions as shown on figure below. It greatly helps in making the robot’s behavior robust, and provide tools for the supervisor to adapt itself to a large set of varying situations

IV. Supervision

A. General description

Rackham is used in a context where there is no need for a high level planner i.e. a system that synthesizes a partially ordered set of tasks to be performed to reach a given goal. Consequently, the highest level of decision is to select what task to achieve. Indeed, the robot is able to perform a number of tasks in a variety of contexts and depending on various conditions (availabilty of visitors, energy level. . .).

Hence, the role of the robot supervisor involves several aspects:

- task selection,
- context-based task refinement,
- adaptive task execution control.

In its current configuration, Rackham, as a tour-guide in the exhibition, has basically to deal with two different tasks: *the search for interaction* (where the robot, left alone in the exhibition, tries to attract a visitor in order to interact with him), *the mission* (where the robot, according to the visitor’s choice, brings him to a selected place).

Depending on the context and the level of abstraction, the task execution is based on three aspects:

1) the definition of a state space, an action space and the construction of a policy[5],
2) the construction of robot primitives (based on the action space definition),
3) the execution of the policy and the control of this execution.

Various schemes are proposed in the literature in order to refine and execute robot tasks in the presence of uncertainty. Mainly they partition the state space [16] , or the action space [13], or both [18].

We do not intend to discuss here the various means that may be used to build policies. What we want to stress is the importance of the models (variables, primitive actions and their parameters, primitive observations) and the ability to decompose them. One key aspect is to construct efficient and robust motion execution primitives that are able by themselves to deal with local contingencies. This allows to reduce the burden of the higher levels and limits the complexity of the associated policy.

[5]by policy we mean : “a solution that specifies what the agent should do for *any* state that the agent might reach” [14], no matter the way it has been computed or deduced

Our choice was to use relatively low level observation and action primitives in order to leave as much flexibility as possible at the policy level. Indeed, as we will see in the sequel, the performance of tasks in the vicinity and/or in interaction with humans is not compatible with a “blackbox” strategy.

Another interesting aspect on which we focus is how the task execution process is influenced by the need for human-robot interaction.

B. Executing tasks in presence of humans

When a task is given, our robot not only needs to execute it, but it also needs to be able to explain it (by exhibiting a legible behaviour or by displaying relevant information) and it should allow humans to act on the course of its actions during their execution.

For instance, during the *mission* task, Rackham should not only be moving toward its goal and avoiding obstables, it also has to maintain the interaction with the humans (waiting for possible inputs like abort or change the mission and displaying any relevant information that may be needed).

There are a number of speech-based or visualisation-based functions that allow to give feedback to the user mainly in terms of messages. Other information such as trajectory, robot position, etc, are displayed directly by the interface as soon as there are available nearby the modules (without direct intervention of the supervisor, we will go back to this point in section V-C.2).

Possible robot actions can be partitioned in two sets:

- navigation related actions: trajectory planning, trajectory following, motion trajectory suspend, motion trajectory resume, motion trajectory stop, change speed, re-initiliaze position manager, re-initiliaze position estimator, end, wait, error.
- interaction related actions: we do not explain here all the actions because there are many of them, but fundamentally they are of three kinds: actions explaining the current navigation action (one for each navigation action), actions explaining the state of the robot (“I’m lost.”, “We are blocked”,. . .) and actions trying to engage people with the robot (“follow me !”, “Please free the way before me to help me relocalize”).

The action selection is essentially performed on the basis of a common state space. Three state variables are needed:

- `localisation quality`: *good, average, bad*
- `trajectory`: *unavailable, available, error*
- `move`: *nothing, begin, moving, blocked, error, stopping, stop, ended*

This state space is an abstraction, for the supervision, of the modules feedbacks. For instance, if the `zone` module has already notified an entrance in the target zone, then the

TABLE I

EXAMPLE OF POLICIES UNROLLING FOR THE *mission* TASK

move	**trajectory**	**localisation**	**navigation action**	**interaction action**
nothing	unavailable	good	trajectory planning	explain action
nothing	available	good	trajectory following	wait
begin	available	good	wait	follow me !
moving	available	good	wait	wait
	(...)		(...)	(...)
moving	available	bad	motion trajectory stop	explain state
stopping	available	bad	wait	explain state
stop	unavailable	bad	re-init position manager	ask freeing the way
	(...)		(...)	(...)
moving	available	good	wait	wait
blocked	available	good	motion trajectory stop	explain state
stopping	available	good	wait	explain state
stop	unavailable	good	wait	explain state
nothing	unavailable	good	trajectory planning	explain action

notification of the end the motion by `ndd` module will switch the value of `move` to *ended*.

Considering that the two activities, navigation and interaction, are of different nature and that one can speak and move at the same time, we decided to separate navigation and interaction policy treatment. This seems also convenient in order to provide a “legible” robot behaviour (not all robot actions need to be displayed). Table I, shows an example of the policies unrolling.

Another key advantage of this separation is to build reusable policies. For instance, in the current system, the navigation policy can be used alone, when the robot has to perform a navigation task without interaction (when it is heading to its re-charging station) or in conjunction with a different human interaction policy.

C. *Executing tasks taking humans into account*

In a second step, we have extended the state space of the interaction policy. Indeed, there are a number of observation functions that are dedicated to the human-robot interaction (detection, tracking of human faces, detection of humans blocking the path...).

In order to take these observations into account, two state variables have been added to the interaction state space:

- `path state in front of the robot`: *free, blocking, blocked, freeing,*
- `face detection state`: *I see you, I don’t see you, I see you again.*

Notice that it explains in fact in which direction the things go: better or worse because it is the interesting information for the robot (and for the visitor to whom this will be shown).

Currently, we just use this information to give feedback to people with messages: are you there ?, here you are !, we are blocked, we can pass again...

In the future, they will be used to select actions that will influence the task execution itself (slowing down, suspending or even aborting execution, etc). This imposes to build task execution policies that are able to react to such inputs.

This kind of information will also be useful to choose the best media or the best conjunction of them to be used to convey messages to the humans.

V. RESULTS AND ANALYSIS

A. *Quantitative results*

Between march and october 2004, Rackam has spent eight weeks at the ”Cité de l’Espace” in four venues[6].

During the last two stays, the robot was completly entrusts to the organizers of Cité de l’Espace. During each experiment many figures and data are automatically logged for analysis purpose: all the requests to the modules and their reply, the covered distance, the visitors interactions, etc.

The results presented bellow are a synthesis of the data collected during the period from october 5th to october 17th. During this stay Rackham was put on mission by the organizers for a total of nearly 33 hours. The robot is then permanently in interaction and was requested for 902 visits to a place of interest. For 732 of them, the robot was able to reach the target zone. 168 were voluntarly interrupted by the visitors. 24 low levels errors detected by the supervisor that required the intervention of the organizer (eg, a critical data, like laser measurements, not updated in time).

The total covered distance is about 10km (the environment is about 25x10 square meters). The average speed seems slow but it integrates all the disturbances and jammings during the motion.

```
MISSIONS = 902
SUCCESS = 732  ANNULATION = 168
TMP-STOP = 137  FAILURES = 24
REQUESTS = 30022   (av. 33 / mission)
DISTANCE total   = 9439 m
         mission = 9109 m
         planned = 9952 m
DURATION total   = 32h 56mn 30sec
         on motion = 5h 4mn  42sec
SPEED on motion mission = 0.16 m/s
```

B. *Visitor behaviours*

1) Human robot interaction: It is striking to notice how the behavior of the visitors highly depends on their age.

Kids immediately identified Rackham as a robot (although it is very different from cartoons ones). They are not afraid at all and even often too effusive, catching the camera when it does not look at them or pushing the robot when it does not move fast enough.

Teenagers and young adults try to find out how it works or how to make the system fail, blocking its path or clicking on all the buttons. They are also very attracted by their own image displayed by the face tracker.

While adults are anxious to understand every thing (technological exhibitions serve to transmit knowledge), the elderly sometimes do not even imagine that this thing can move or that they can communicate through the (tactile) screen.

2) Interface misunderstanding: Among the various data displayed on the interface, we have implemented a “radar-like” representation of the local map and of the proximity data, in order to show how the robot models the dynamic

[6]see http://www.laas.fr/ sara/laasko.

Fig. 6. Head of Rackham emerging from a sea of kids.

obstacles (visitors), and static ones (from the map). Many visitors, looking at the robot, take that element for a joystick to make the robot move and stay disappointed on the robot inactivity.

3) How does it works ?: There is a difference between what people think the robot can do and what it really does. Generally, people do not understand that the robot is deaf, especially because it has an animated face that can speak.

Besides, they do not apprehend how the robot localizes itself or detects obstacles. They generally think that the robot uses the ultra-sonic sensors that they see (but that we do not use) or its cameras. They do not understand why the robot does not stop when they put their hands on them.

That brings out the difficulty for a robot to show the right code to be understand. People will gradually understand better how a robot works, but we have to take care to let the robot be “readable”.

C. Towards control and dataflow for interactivity

Rackham, with its currently functionality and limitation has been an attractive tour guide for many visitors. However, by observing the way people interact with it and by analysing the cases where it failed to accomplish its mission (either because of a software failure, or because the humans did not understand or follow what the robot was expecting from them – following it, or freeing the way when necessary), we are able to draw some conclusion and define future work.

1) Functional data relevance: A module often carries out complex computations and manipulates a large amount of data to accomplish the required functionalities. However the output result is generally simplified to be easily used and and interpreted. For example we would naturally expect a face detection module to return a flag indicating if there is somebody or not…and that’s what it does. But this binary information is a strong impoverishment of the result. Indeed in real systems results are rarely certain and most probably the algorithm has an idea of the confidence on its result. This confidence is a primordial information for a correct control. The problem is : can we measure and to express such an uncertainty in a coherent, standardized and comprehensible manner for all the data produced by the modules ?

Beyond that resulting confidence, other data, hidden within the modules, can be pertinent for the system control. For instance a module that localizes the robot using proximetrics data has an idea of the kind of environment encountered (eg, corridor, cluttered, etc.)

2) Interface limitations: Nor the supervisor nor the interface have access precisely to the other’s state. That brings limitations and complications.

First of all, it is difficult for the supervisor to treat messages given back by the interface.

More annoying for the interaction ability, interface is loaded at the beginning and although information display on it change along the time, it is not possible to put forward particular information when it’s necessary (or hide when it is not), to display elements bigger or not whether their current importance. For example, when the visitor have to choose his destination, the important things his the map and the possible destinations, not the face detection that distract him.

In fact, we missed the possibility to control directly what is displayed on the interface at a supervisor level.

We have now enhanced and redesigned this part of the system to allow better communication between the supervisor and the interface and better control from the supervisor on the interface.

VI. Conclusion and future works

In this paper we have proposed an instance of the LAAS architecture that is adapted to human/robot interaction. We’ve built a supervisor which represents explicitely the interaction tasks in addition to traditional navigation tasks and is able to explain its behaviour and to interact with the people present in the vicinity of the robot during its mission. We have combined this supervisor with a functional level (implementing localization, motion control with collision avoidance and interaction with users) that provide enough flexibility and redundancy to achieve a certain level of robustness in an relatively hostile environment.

Rackham has now solid foundations which allows it to navigate in a robust manner and to establish a simple interaction with people in a real world environment. It has been effectively used quite intensively and is considered as an attractive and succesful component of the overall exhibition.

But this is only the first page of the story.

We now work to enhance Rackham’s interaction and perception capabilities.

The integration of such capabilities will be done with the concern of developing a systematic manner to integrate more sophisticated context interpretation and to provide decision-making to synthesize and control interactive tasks at various levels.

Acknowledgments

The work described in this paper was conducted within the EU Integrated Project COGNIRON (”The Cognitive Companion") and was funded by the European Commission Division FP6-IST Future and Emerging Technologies under Contract FP6-002020 and by the French National Program ROBEA. Numerous persons have contributed to the project : the authors want to thanks: Aymeric Yvernes, Coline Lelong-Pantel, Florent Lanterna, Stéphane Plais, Guillaume Infantes, Maxime Cottret, Abdelatif Baba, Paulo Menezes, Ludovic Brethes, Frederic Lerasle; from the Space City Museum : Jean-Noël Plachez, Véronique Hallard, Maxime Randisi and Stephan Lauriol; and from ICP Grenoble : Frédéric Elisei.

REFERENCES

[1] R. Alami, R. Chatila, S. Fleury, M. Ghallab, F. Ingrand, "An architecture for autonomy", *International Journal of Robotic Research*, Vol.17, N°4, pp.315-337, Avril 1998.

[2] G. Bailly, M. Bérar, F. Elisei, and M. Odisio, "Audiovisual speech synthesis", *International Journal of Speech Technology*, 6:331-346, 2003.

[3] L. Brèthes, P. Menezes, F. Lerasle, and J. Hayet, "Face tracking and hand gesture recognition for human robot interaction," *International Conference on Robotics and Automation, New Orleans, May 27 - June 1 2004*.

[4] W. Burgard and A.B. Cremers and D. Fox and D. Häehnel and G. Lakemeyer and D. Shulz and W. Steiner and S. Thrun, "Experiences with an interactive museum tour-guide robot", *Artificial Intelligence* , 114 (1999) 3-55.

[5] S.Fleury, M.Herrb, R.Chatila, "GenoM: a Tool for the Specification and the Implementation of Operating Modules in a Distributed Robot Architecture", *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Grenoble, France, 1997.

[6] T. Fong, I. Nourbakhsh, K. Dautenhahn, "A survey of socially interactive robots," *Robotics and Autonomous Systems, Special issue on Socially Interactive Robots,* 42(3-4), 2003.

[7] B. Graf, M. Hans, and R. D. Schraft. "Care-O-bot II - Development of a next generation robotic home assistant",*Autonomous Robots*, 16(2):193-205, 2004.

[8] H. Huttenrauch and K. Severinson Eklundh. "Fetch-and-carry with CERO: Observations from a long-term user study with a service robot",*In Proc. IEEE Int. Workshop on Robot-Human Interactive Communication (ROMAN)*, pages 158-163. IEEE Press, 2002.

[9] B. J. A. Krose, J. M. Porta, A. J. N. van Breemen, K. Crucq, M. Nuttin, and E. Demeester. "Lino, the user-interface robot",*In European Symposium on Ambient Intelligence (EUSAI)*, pages 264ñ274, 2003.

[10] J. Minguez, L. Montano. "Nearness Diagram Navigation (ND): Collision Avoidance in Troublesome Scenarios",*IEEE Transactions on Robotics and Automation*, p 154, 2004.

[11] Illah Nourbakhsh, Judith Bobenage, Sebastien Grange, Ron Lutz, Roland Meyer and Alvaro Soto, "An Affective Mobile Educator with a Full-time Job," *Artificial Intelligence*, vol. 114,num. 1-2, pp. 95–124, 1999.

[12] J. Pineau, M. Montemerlo, M. Pollack, N. Roy, and S. Thrun, "Towards Robotic Assistants in Nursing Homes: Challenges and Results.," *Robotics and Autonomous Systems*, 42(3-4), 2003.

[13] J. Pineau and S. Thrun. "An integrated approach to hierarchy and abstraction for POMDPs," *Technical Report CMU-RI-TR-02-21* , Carnegie Mellon University, 2002.

[14] S. Russell and P. Norvig, *Artificial Intelligence : A Modern Approach,* Prentice Hall, 2003.

[15] Siegwart, R. and et al. (2003) "Robox at Expo.02: A Large Scale Installation of Personal Robots." *Special issue on Socially Interactive Robots, Robotics and Autonomous Systems* , 42 (2003) 203-222.

[16] Georgios Theocharous, Kevin Murphy, Leslie Kaelbling, ''Representing hierarchical POMDPs as DBNs for multi-scale robot localization," *ICRA'04 (Intl. Conf. on Robotics and Automation)* , New Orleans, LA, USA, 2004.

[17] S. Thrun and M. Beetz and M. Bennewitz and W. Burgard and A.B. Cremers and F. Dellaert and D. Fox and D. Haehnel and C. Rosenberg and N. Roy and J. Schulte and D. Schulz "Probabilistic Algorithms and the Interactive Museum Tour-Guide Robot Minerva", *Journal of Robotics Research* , vol. 19, num. 11, 2000.

[18] Turkett, Jr., W. and Rose, J. , "Planning With Agents: An Efficient Approach Using Hierarchical Dynamic Decision Networks," *Proceedings of the Fourth International Workshop on Engineering Societies in the Agent World* , London, England, 2003.

[19] B. Wrede, A. Haasch, N. Hofemann, S. Hohenner, S. Hüwel, M. Kleinehagenbrock, S. Lang, S. Li, I. Toptsis, G. A. Fink, J. Fritsch, and G. Sagerer. "Research issues for designing robot companions: BIRON as a case study", *In P. Drews, editor, Proc. IEEE Conf. on Mechatronics and Robotics*, volume 4, pages 1491-1496, Aachen, Germany, September 2004. Eysoldt-Verlag, Aachen.

A CORBA-Based Distributed Software Architecture for Control of Service Robots

Steffen Knoop, Stefan Vacek, Raoul Zöllner, Christian Au, Rüdiger Dillmann
Institute for Computer Design and Fault Tolerance (IRF)
Universität Karlsruhe (TH)
Germany
Email: {knoop,vacek,zoellner,au,dillmann}@ira.uka.de

***Abstract*— This paper presents the distributed robot control software architecture developed for the autonomous service robot Albert2. The development of this architecture is focused on two major issues: Modularity and the integration of learning aspects. Each module within the architecture is presented, as well as the underlying event-based communication framework. An approach for integration of learning capabilities is proposed.**

I. Introduction

The architecture of a robotic system strongly influences the functionality and usability of such a system. In the last decades, many architectures have been presented which reflect the different requirements a robotic system has to cope with. Many requirements have to be considered during the design of a robot architecture, e.g. reactivity, which means the ability to react spontaneously to unexpected events, or the need for an adequate type of control flow. Other requirements are robustness, programmability, extensibility and adaptability amongst many more.

Extensibility and adaptability become very important if the robot has to act in a dynamic and partially unknown environment. This applies only in a very limited way to industrial robots, in contrast service robots are faced permanently with unknown or partially known situations. Due to the fact that service robots are not restricted to a well-defined working area but work together with humans in a human-centered environment, they must be prepared to cope with unknown, i.e. new, situations as well as to acquire and integrate new knowledge.

Extensibility can be reached through learning of task knowledge. Learning can be done offline or online, respectively, and it is desirable to have the possibility to add newly acquired knowledge to the robot's existent knowledge. Adaptability is the robot's ability to adjust itself to dynamic or even new environments. Furthermore it should be able to use its task knowledge if applicable even if the knowledge was learned in a different environment.

In this paper we concentrate on integrating into the architecture the robot's ability to learn task knowledge. On the one hand, this knowledge will be acquired offline through single user demonstrations, and on the other hand, the system should be able to adapt and to extend its knowledge at runtime. It is clear that these abilities have to be taken into account while designing a robot's architecture.

Section II gives a short overview of existing approaches, section III introduces our robot system Albert2, and section IV explains the learning approach ”Programming by Demonstration”. The proposed software architecture is presented in section V. First results are shown in section VI.

II. State of the Art

As has been stated by many research groups, e.g. [1], architectures form the fundament of robotic systems. Typically the existing approaches can be divided mainly into three different classes:

Hierarchical architectures are based on a top-down approach. Communication and controlflow is only done vertically between the different layers. Higher levels delegate subgoals to lower levels to achieve superordinate goals. An advantage of this approach is that planning is straightforward because there is a superior view on the system. A drawback of these systems is their insufficient reactivity to dynamical environments. Examples for hierarchical systems can be found in [2] and [3].

Behavioral systems (e.g. [4], see [5] for an exhaustive discussion) consist of several modules each representing a specific behavior. These behaviors are not grouped in a hierarchical manner but are running concurrently. These systems are robust because they don't rely on the existence of specific functional units compared to hierarchical systems. Because of their modular style single behaviors can easily be added. The lack of a high-level control unit complicates the planning to achieve certain goals. Moreover safety considerations are not easy to integrate.

Hybrid architectures try to combine the well defined control flow of hierarchical systems with the advantages of behavior based systems which are highly reactive. These approaches have become very famous recently and there exists a vast number of architectures with different foci (e.g. [6], [7], [8]) and [9].

Another aspect which must be considered during the design of a robot architecture is the learning ability of the system. Zhang and Knoll [3] use a hierarchical approach to learn operation sequences of two arm manipulations. Bonasso et al. (cf. [10], [11]) use an architecture organized into three layers: skills, sequencing and planning. Learning is possible in each layer and through different layers. Other research work concentrates on learning a specific behavior [12] or mappings of sensor data [13].

III. Service Robot ALBERT2

The architecture which is presented in this paper has been realized on the service robot Albert2. Up to now it is mainly used in a kitchen environment and the focus of task learning and execution is laid on household environments.

Figure 1 shows the service robot Albert2 used for experiments in a household environment. For manipulation, it is equipped with 7 DoF arm and a three finger hand. A mobile platform serves for navigation. Built in sensors are a laser range sensor, a color stereo camera mounted on a pan tilt unit, and a force torque sensor mounted on the arm. For user interaction, the speech recognition system Janus [14] is used in combination with an external microphone, and loudspeakers on the robot provide speech output. Additionally, a touchscreen is attached to the front of the robot's upper body. It is used for displaying the robot's current internal state (e.g. waiting, working, idle) and for user interaction e.g by prompting questions on how to proceed (e.g. showing images of all graspable objects) to the user.

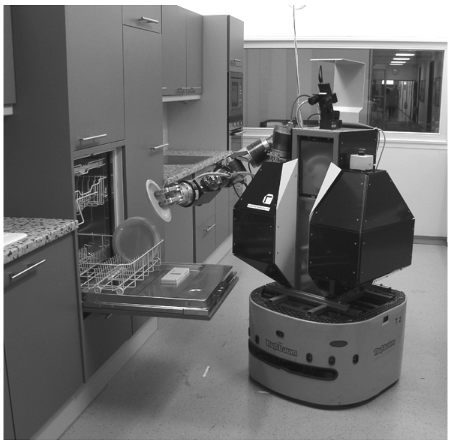

Fig. 1. Service robot Albert2

IV. Programming by Demonstration

In this section we shortly describe our approach for teaching a robot new task knowledge and how this knowledge is represented. A complete description of the learning mechanisms can be found e.g. in [15]. The approach is called *Programming by Demonstration* since the task is simply demonstrated by a human user in order to offer an easy to use interface for the unexperienced user. The focus lies on teaching higher level tasks since we assume that basic skills like specific grasps or basic motor control for movements of the arm already exist on the robotic system. We rely on one-shot-learning because users should not be forced to demonstrate the same task multiple times. The approach is basically composed of the following phases:

- Demonstration of the task
- Perception, data preprocessing and fusion
- Segmentation of the acquired data
- Generalization of the segmented data
- Simulation of the learned task and refinement of the task knowledge through user interaction
- Transfer of the newly acquired task knowledge onto the robotic system

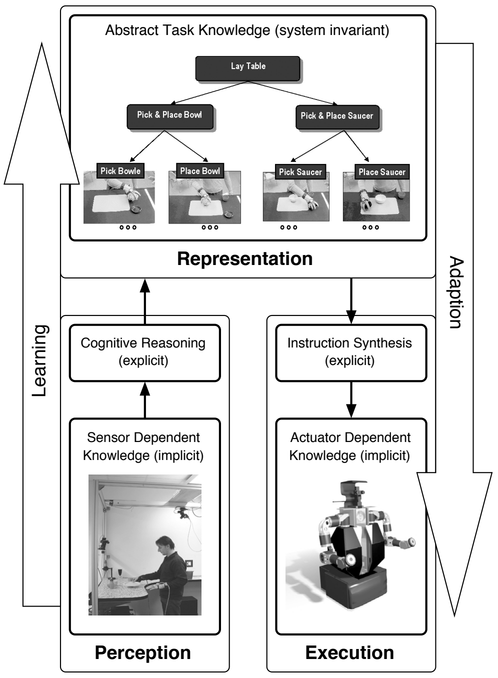
Abstract Task Knowledge (system invariant)
Lay Table
Pick & Place Bowl
Pick & Place Saucer
Pick Bowle
Place Bowl
Pick Saucer
Place Saucer
Representation
Learning
Adaption
Cognitive Reasoning (explicit)
Instruction Synthesis (explicit)
Sensor Dependent Knowledge (implicit)
Actuator Dependent Knowledge (implicit)
Perception
Execution

Fig. 2. Transfer of task knowledge of a human demonstration to a robotic target system.

The teaching starts with the user's demonstration in a specially designed training center where the performed task is perceived through different sensors like camera systems and data gloves. This center is neccessary because todays robot's sensors do not provide enough information with sufficiant precision. During the demonstration sensor data is merged with respect to the different sensor models. In the next stage the data is segmented depending on recognized grasps and movements. After that the data is analyzed and elementary operators which correspond to basic skills of the robot are identified. For generalization these basic operators are grouped hierarchically into so called macro-operators.

Finally the knowledge has to be transferred to the robotic system. Therefore we need a representation of the task knowledge for the robot which needs to be reflected by the architecture. Additionally a mapping of the macro-operators to the robot's task knowledge representations needs to be defined. The robot's representation must be extendable to be able to add new task knowledge and be adaptable in a way such that the existing knowledge could be refined either through reinforcement by the robot itself or by user interaction.

Figure 2 shows the learning process from user demonstration to generalized macro-operators on the left and the transfer of the task knowledge to the robot system on the right side. In the *Perception* and *Execution* phases the type of background knowlegde which is used for data processing is indicated, e.g. the implicit sensor models used during the perception of the user demonstration, which enables data fusion of different sensor sources. In contrary, macro-operators are abstract and thus do not rely on this background knowledge.

V. Software Architecture

A software architecture that is able to control a mobile robot basically has to cover three aspects to comprise all capabilities of the system: Control of the hardware, representation of the environment and integration of knowledge about skill and task execution. Including the demand for a lifelong learning system, these requirements have to be fulfilled in a way that the system remains extensible.

An expedient approach to design such an architecture is therefore to identify the functional components as hardware abstraction, environment model and skill and task execution. Our proposed software architecture consists consequently of the following components (see figure 3):

The *hardware agents* encapsulate all hardware specific functions. There is a hardware agent for each hardware in the robot system, e.g. one for the robot arm, one for the hand and one for the voice. Skill and task knowledge is represented by the *flexible programs*. Example skills are ”drive to position” and ”open door”, tasks can be ”transport an object” or ”put object on the table”. All information about the environment is stored in an *environment model*. This can hold all kinds of data: Coordinates, objects, relations, features, images or sounds.

Evidently, there needs to be a way of communication for these components. The proposed communication bus seen in figure 3 allows communication between all components, but nevertheless standardizes and restricts data exchange to a defined set of data structures. It is event-based and is able to incorporate internal as well as external (user triggered) events.

This approach is very much inspired by the way an operating system works. The components are in detail:

- **The communication infrastructure** consists of a notification distribution instance, where clients can subscribe for certain notification types. Notifications may be delivered by internal or external sources.
- **Hardware agents** (resources) represent real or virtual sensors and/or actuators. There is an agent for *laser scanner* and for *cameras*, but there may be also an agent for *detection of humans* which incorporates laser scanner and vision information. Hardware agents are also referred to as *resources*.
- **The agent manager** administrates all hardware agents and provides the resource management. Each notification that is passed to an agent is filtered by the agent manager. Thus, unauthorized commands (from instances which have not locked the called resource) to agents can be intercepted.
- **Flexible programs** contain the skill and task knowledge. These flexible programs (*FPs*) are the core of the proposed robot control architecture. Learning, within our context, means creation, extension and adaption of flexible programs.
- **The flexible program manager** administrates the flexible programs. All notifications addressed to flexible programs are filtered and delivered by the flexible program manager. It also holds a list of all currently existing FPs, including their type.
- **Domain controlling and supervision** as well as FP instantiation and priority control is done by the flexible program supervisor. Depending on the current context, FPs are created, prioritized or deleted. In later development, learning capabilities will be extended to this component.
- **The environment model** holds environmental data as well as the robot’s internal state. It is implemented as a blackboard. An intelligent xml data base, which is being developed at our research group, will soon be integrated and used for storage of task, skill and object feature knowledge.

The implementation of the proposed software architecture (see figure 3) consists of a set of CORBA object types, which communicate via a communication instance using a publish-subscribe mechanism (see also [16]).

Each public object’s interface is described in CORBA’s Interface Definition Language, which enables the use of different programming languages within the same framework. We use C++ and have base classes implemented for each object type, which encapsulate all communicational and infrastructural aspects. This simplifies usage and implementation of new elements, as users do not have to know any details about communication and functionality of other objects.

Instantiating each component as a separate CORBA object allows object distribution over several computers and different operating systems. It also decouples operation of caller and callee, which is essential in complex systems. By running different system parts on different machines, it is on the one hand possible to split up required computation power, and on the other hand it enables different users to use the same infrastructure.

As most of the communicated information is very high-level symbolic information, real-time constraints are not defined primarily by data transmission, but by hardware restrictions; nevertheless, the use of distributed objects holds the risk of data loss or speed reduction.

Some of these architecture components will now be described in detail.

A. Communication layer

The basis for every distributed architecture is formed by the communication layer. The proposed architecture uses a special message format (referred to as *notifications*) as a basic information container for communication.

Notifications are small data blocks, wherein most of the information is coded as plain text. This is important for debugging and readability.

Notifications are always of one of three types:

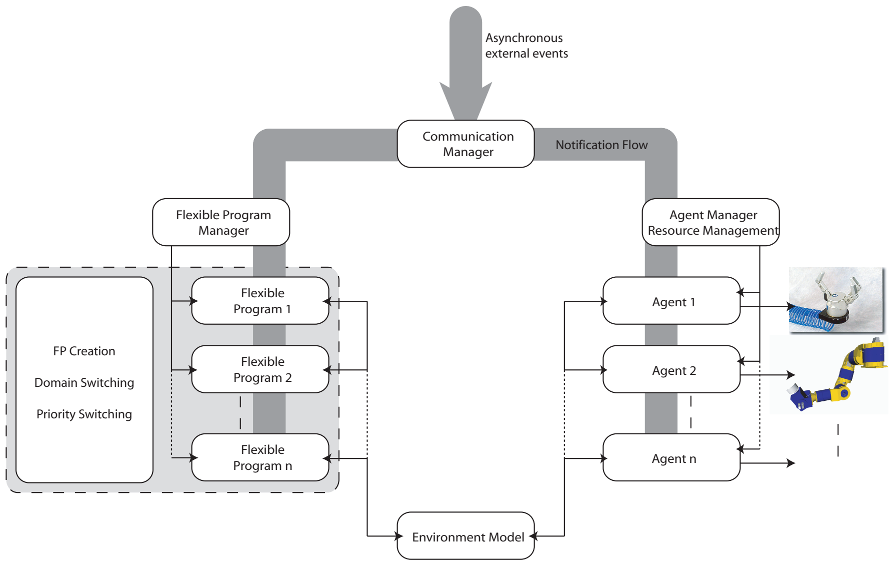

Fig. 3. CORBA-based software architecture with flexible programs, hardware agents and communication bus

- **Events** are always delivered to flexible programs. Events are generated either from hardware agents or from external sources like speech or gesture recognition. Events are data directed from sensors/actors to the control level.
- **Actions** are delivered to hardware agents. They are mostly generated by flexible programs. Actions correspond to data flow from the symbolic control level to the sensor/actor level.
- **Requests** are demands to the resource management and therefore delivered to the agent manager. Requests can be to lock, free, or try-lock resources.

TABLE I
NOTIFICATION DEFINITION

Header	
Type	Event, Action, Request
Time	Time and date of creation
Sender	Unique ID of sender instance
Receiver(s)	Unique IDs of receiver(s)
Body	
Name	Notification specification
key value list	Optional contents

A complete notification consists of a header and a body (see table I), the notification body again holds a name and a set of key value pairs. These have to be set by the sender and transmit the information. As every slot within a notification is defined in plain text, debugging and readability are simplified using a viewer and tools for manual creation and sending of notifications. Each notification name and the associated key value pairs have to be defined in advance. Thus, all inter-object communication is standardized and can be checked for consistency and conformity with the specification.

To control the data flow of notifications and to avoid storage of unattended messages, the timestamp in the notification header controls notification deletion from the delivery queues.

Notification distribution is performed by the communication manager using a publish-subscribe mechanism. Undelivered messages are not stored within the communication manager, but discarded directly. In contrast, the agent and flexible program manager possess a notification queue, wherein messages are stored until they are either delivered or expired.

B. Resource management

Real and virtual hardware agents are also referred to as resources. These resources have to be locked and unlocked before usage from any flexible program to avoid collisions. This resource management is done by the agent manager, which also administrates the hardware agents. Agent administration comprises keeping track of the agent's state (*running, sleeping, stopped*), of current locks, of safety aspects (stop/resume all agents in case of emergency stop) and also notification surveillance and distribution among hardware agents.

Agents can only be locked by one flexible program at a time, but these usage authorizations can be inherited.

C. Hardware agents

Each hardware agent is identified by a unique ID and a unique agent type. Depending on the agent type, an agent can accept different notifications: A vision agent is able e.g. to search objects, while an robot arm agent can move or switch move mode.

The capabilities of an agent are context independent and defined by the capabilities of the underlying (real or virtual) hardware. The hardware agent for the mobile platform e.g. accepts actions for geometric and topological navigation: *Rotate relative/absolute*, *drive relative/absolute*, *set position*, *stop driving* as well as *drive to node X*.

D. Interruption and exception handling

There are two cases, where running operations have to be canceled immediately:

- The goal has changed while an operation was running. This goal change can be caused by a human or by other external events. The goal change has to be carried out as ”smooth” as possible.
- An emergency situation occurs. This can either be an internal one (like collision detection, or some other conflict) or invoked by the user (e.g. saying ”stop” or pushing a soft stop button). It is very important that these exceptions trigger a stop of all hardware components immediately and that the system remains passive until an explicit resume command occurs.

While the latter is realized through direct stop calls (which are not handled by the standard communication and queuing software), to achieve the former, a continuous goal adjustment has to be performed within the hardware agents.

E. Flexible programs

Flexible programs always have the following properties:

- A unique ID, used to identify the FP within the whole system.
- A state (inactive/active, when active: FSM-like state description).
- A list of notifications that are accepted at the current state.
- A priority which can be used for the decision which FP gets a notification that is accepted by more than one FP.
- A list of currently owned resources.

These properties are used, set or read by the flexible program manager and the domain controller and are needed for communication.

Flexible programs encode the task knowledge. They generate agent commands from background knowledge, environmental data, and robot internal states. FPs also have to handle errors.

The flexible program description is independent from the robot’s kinematics, as all hardware specific algorithms are encapsulated in the hardware agent objects. Of course, to keep two robots exchangeable, they must possess similar hardware components.

F. Flexible program specification

Within our concept, the flexible programs hold the task knowledge that is included in the system. This knowledge is represented as the number, type, parameterization and order of the basic skills used. We propose to use a meta programming language. This task description language has to fulfill several requirements:

- It has to be powerful enough to describe all possible action sequences as well as dependencies, decisions and exceptions.
- On the other hand, this language should make as many restrictions as possible to ensure ease of use and debugging capabilities as well as to avoid ambiguities.
- It should be easily convertible to a standard data description format like xml. This enables use of standard tools for saving, searching, comparing and merging different flexible programs.
- To guarantee extendibility at runtime, the language must not need a compilation process. A program should be directly executable.

There are languages that fulfill some of these requirements (e.g. *ESL*, see [17], or *TDL*, see [18]), but in particular the program extendibility has not been an explicit demand.

G. Task knowledge representation

A flexible program holds task knowledge, which is encoded in its specific structure and parameters. The implementation of an FP which provides the interface and functionality described in section V-E is proposed as follows:

A flexible program is composed of functional blocks. These functional blocks define functionality, pre- and post-conditions in the same way FPs do; in fact, often FPs are used as functional blocks within other FPs.

One functional block consists of an input parameter list (coordinates, object names, persons, area of interest, etc), different result states (several success and failure states) and their output parameters (positions, objects, persons, etc), and the immanent functionality. This can consist of one or more calls to hardware agents or other flexible programs, including requests to the environment model to get or set required or perceived data and results. Such a functional block comprises manipulation and/or perception, the environment (including the robot itself) is always in the loop. It also contains information about interruptibility. If the current block is interruptible, there are additional suspend and resume methods.

Inside a flexible program, these blocks are then wired according to the task specification and depending on the respective results. As this sequence is a high level action chain, it is very easy to understand and debug.

It is important to note that there must not exist concurrent, asynchronous processes within one FP. This assumption has to be made to avoid collisions in resource management. This case has to be solved with different flexible programs.

H. Generation and extension of flexible programs

As flexible programs are defined on a high abstraction level using functional blocks, generation and extension is possible offline and online. Basically, there are three ways to integrate new knowledge into the system:

1) Manual FP definition: Flexible programs can be defined by a programmer. This can be done either using a GUI to build and connect functional blocks or by coding by hand.

2) FP derivation from user demonstrations: Flexible programs can be generated from macro operators (see section IV). Macro operators are set up by observation of a human demonstrating a task. This observation can either be done offline using extra hardware and software or online by the robot itself. Currently, a special demonstration environment is used with special sensors and software.

This transformation process will presumably be carried out semiautomatic; the operator will still need to correct and modify robot programs that are generated automatically.

3) FP extension by user interaction at runtime: Robot programs that are not complete, i.e. not every possible outcome situation is covered by the FP, can be extended at runtime. If such a situation occurs, the system can ask the user for a solution. Then, an empty functional block is constructed and filled by joining information given by the user with background and context knowledge. If this extension leads to a satisfying result, the FP is saved and used at the next time.

If e.g. the robot fails to move to the next room because the door is closed (and closed doors are not modeled as obstacles yet), the user can tell it to open the door and try again. This knowledge is then saved and reused whenever this situation occurs.

VI. Results

The architecture has been successfully implemented and tested on the service robot ”Albert2”. The robot is able to execute tasks like fetch-and-carry or pick-and-place operations successfully.

The required skills and time needed for implementation of additional functionality was drastically reduced, because the chosen architecture and knowledge representation supported the programmer by encapsulating most low level and infrastructural issues related to communication, timing, resource management, etc.

VII. Conclusions and future work

In this paper, we presented an approach for a service robot architecture. The way task knowledge is represented within the architecture allows for permanent extension of the knowledge base. Knowledge is perceived through user demonstrations, and can be extended at runtime by the robot itself or by user interaction. Taking these options into account, a suitable task knowledge representation has been proposed, where task knowledge is described by *flexible programs*.

The proposed learning capabilities will now be implemented and integrated in the presented framework. The architecture will be extended by a task planner, which will be incorporated into the learning process.

Acknowledgment

This work has been partially supported by the german collaborative research center “SFB Humanoid Robots” granted by Deutsche Forschungsgemeinschaft (see `http://www.sfb588.uni-karlsruhe.de`) and by the European Union via the Integrated project “COGNIRON (Cognitive Robot Companion)” funded under the Sixth Framework Programme (see `http://www.cogniron.org`).

References

[1] Ève Coste-Manière and R. Simmons, “Architecture, the backbone of robotic systems,” in *Proc. 2000 IEEE International Conference on Robotics and Automation*, San Francisco, CA, April 2000.

[2] J. Albus, R. Lumia, and H. McCain, “Hierarchical control of intelligent machines applied to space station telerobots,” in *Transactions on Aerospace and Electronic Systems*, vol. 24, September 1988, pp. 535–541.

[3] A. K. J. Zhang, “Control architecture and experiment of a situated robot system for interactive assembly,” in *Proc. 2002 IEEE International Conference on Robotics and Automation*, Washington, D.C., May 2002.

[4] S. A. Blum, “From a corba-based software framework to a component-based system architecture for controlling a mobile robot,” in *International Conference of Computer Vision Systems (ICVS 2003)*, Graz, Austria, April 2003, pp. 333–344.

[5] R. C. Arkin, *Bahvior-Based Robotics*. Cambridge, Massachusetts: The MIT Press, 1998.

[6] R. Alami, R. Chatila, S. Fleury, M. Herrb, F. Ingrand, M. Khatib, B. Morisset, P. Moutarlier, and T. Siméon, “Around the lab in 40 days ...” in *Proc. 2000 IEEE International Conference on Robotics and Automation*, San Francisco, CA, April 2000.

[7] K. H. Low, W. K. Keow, and M. H. A. Jr., “A hybrid mobile robot architecture with integrated planning and control,” in *Proc. of the First International Joint Congerence on Autonomous Agents and Multi-Agent Systems*, Bologna, Italy, July 2002, pp. 219–226.

[8] M. N. Nicolescu and M. J. Mataric, “A hierarchical architecture for behavior-based robots,” in *Proc. of the First International Joint Congerence on Autonomous Agents and Multi-Agent Systems*, Bologna, Italy, July 15-19 2002.

[9] P. Elinas, J. Hoey, D. Lahey, J. D. Montgomery, D. Murray, S. Se, and J. J. Little, “Waiting with josé, a vision-based mobile robot,” in *Proc. 2002 IEEE International Conference on Robotics and Automation*, Washington, D.C., May 2002.

[10] R. P. Bonasso and D. Kortenkamp, “An intelligent agent architecture in which to pursue robot learning.”

[11] R. P. Bonasso, J. Firby, E. Gat, D. Kortenkamp, D. P. Miller, and M. G. Slack, “Experiences with an architecture for intelligent, reactive agents,” *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 9, no. 2/3, pp. 237–256, April 1997.

[12] W. Paquier and R. Chatilla, “An architecture for robot learning,” in *7th International Conference on Intelligent Autonomous Systems*, Marina del Rey, California, USA, March 2002.

[13] M. Sahami, J. Lilly, and B. Rollins, “An autonomous mobile robot architecture using belief networks and neural networks,” in *working paper, Department of Computer Science, Standford University*, 1995.

[14] H. Soltau, F. Metze, C. Fügen, and A. Waibel, “A one-pass decoder based on polymorphic linguistic context assignment,” in *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU ’01)*, Madonna di Campiglio, Italy, 2001.

[15] M. Ehrenmann, R. Zöllner, O. Rogalla, S. Vacek, and R. Dillmann, “Observation in programming by demonstration: Training and execution environment,” in *IEEE International Conference on Humanoid Robots*, Karlsruhe, Germany, Octobre 2003.

[16] R. Dillmann, “Interactive natural programming of robots: Introductory overview,” in *DREH 2002*. Tsukuba, Ibaraki, Japan: Tsukuba Research Center, AIST, December 2002.

[17] R. P. Bonasso, J. Firby, E. Gat, D. Kortenkamp, D. P. Miller, and M. G. Slack, “Experiences with an architecture for intelligent, reactive agents,” vol. 9, no. 2/3, Apr. 1997, pp. 237–256. [Online]. Available: citeseer.ist.psu.edu/bonasso97experiences.html

[18] R. Simmons and D. Apfelbaum, “A task description language for robot control,” 1998. [Online]. Available: citeseer.ist.psu.edu/simmons98task.html