in: Proc. of the 2nd European Conference on Mobile Robots (ECMR 2005), Ancona, Italy, pp. 224-229, stampalibri 2005 (ISBN 88-89177-187)

A NEW CONTROL ARCHITECTURE FOR MOBILE INTERACTION-ROBOTS *

C. Martin, A. Scheidig, T. Wilhelm, C. Schroeter, H.-J. Boehme, and H.-M. Gross

Ilmenau Technical University
Department of Neuroinformatics and Cognitive Robotics
christian.martin@tu-ilmenau.de

ABSTRACT

A very important component of each mobile robot is a good control architecture, which controls the whole robot. Besides the integration of modules for navigation and the Human-Robot Interaction skills, an often underestimated component is the application itself. Without an useful application the robot itself doesn't make sense. This paper describes a new architecture, which provides programming users a modular, extensible, transparent and portable systems and which allows a fast development of robotic applications even for non-programming users. Therefore we propose a four layer architecture, which strictly separates the application from the robot specific skill. We have implemented and successfully tested our proposed architecture on two totally different robots for to different applications.

1. INTRODUCTION

To develop an interactive mobile robot, each researcher has to deal with a lot of different classic robotic problems, like a precise localization, a collision-free navigation and a robust Human-Robot Interaction (HRI) interface. Furthermore, another very important component of an interactive mobile robot is the application itself. Without an useful application the robot itself doesn't make sense.

That means, that each developer has to design a specific application to give this robot an useful task. We will describe a new robot control architecture, which on the one side gives programming users a very flexible system, which is modular, extensible, transparent and portable for different robot platforms, and on the other side allows a fast development of robotic applications, and which is easy to understand and use for newcomers or non-programming users.

We have implemented and tested our proposed architecture on our robots HOROS and PERSES.

A standardized control architecture for mobile robots is still an open question today. But there are a wide range of academical and also commerical (like ERSP [1]) work.

This paper is organized as follows: In section 2 we will give a short overview over existing control architectures for interactive mobile robots. In section 3, our new architecture will be explained in detail. Section 4 is dedicated to two examples of applications, which were realizied with our architecture. The paper closes with a summary and conclusions in section 5.

2. STATE OF THE ART

The most modern control architectures are heterogeneous and are called *hybrid*. They contain *reactive* as well as *deliberative* components.

Over the last years, many different robot system architectures were developed and introduced. A good overview is given in [2]. On the one hand, there are some architectures consisting of up to three layers (like 3T, CLARAty [3] or OROCOS), which are basically designed to deal with navigation problems and on the other hand there are some low-level architectures (like Player/Stage [4]), which only describe a kind of a robotic programming interface. Furthermore, there are existing systems in-between (like CARMEN [5]), which consist of a simple architecture and a full programming interface.

The major problem of all theses systems is, that they are basically designed to connect and control the different modules and methods (the skills) of a robot, but they are typically not designed to build a concrete robotic application. In most shown examples, mainly a few modules for collision avoidance, path planning, self localization, and other navigation skills are implemented. As far as we know, none of theses systems was tested or used with methods for Human-Robot Interaction. Furthermore, none of theses systems provides a possibility to integrate modules, which are required to build a real interactive mobile robot, like elements for a graphical user interface or a dialog management system. The only way to create an application for an interactive mobile robot based on the systems mentioned above seems to be to create a step-by-step program, which controls the whole system. Typically this can exclusively be done by a developer or programmer, which knows the whole system and desired application in detail.

This way, in the last years, several interactive mobile robots with specific architectures were built and introduced.

*THIS WORK WAS SUPPORTED BY THURINGIAN MINISTRY OF SCIENCE, RESEARCH AND ARTS (GRANT NUMBER: B509-03007).

One of theses robots is GRACE (Graduate Robot Attending a ConferencE) [6], which was build to tackle the AAAI Robot Challenge in 2003. The robot's software consists of many programs communicating via IPC (Inter Process Communication) and CARMEN [5]. The different programs exchange messages, e.g. sensor information, commands, or events. Each program provides one skill of the robot. This distribution guarantees the reuseability and the modularity of this system. But if someone wants to realize a new application with such an architecture, he has to know the whole system and all programs in detail.

The humanoid robot HERMES [7] is controlled by a behavior-based system [8], which uses a situation module (situation assessment and behavior selection) as core of the whole system. Around this core a set of skills is used to control the robot. The skills and the situation module are communicating via events and messages. The architecture for HERMES was designed to be reusable, but as far as we know, it was never used for another robot.

The robot MOBSY [9] uses two levels for the software integration. The most abstract level is the task level, where the overall behavior of the system is determined. The second level contains the robot-specific methods, which provide the different skills of the robot. The skills will be used in the task level to execute the behaviors. The architecture of MOBSY seems to be specially designed for the appropriate task and is not reusable for other applications.

The Care-O-bot [10] uses a hybrid architecture. The four most important components of the robot are the Man-Machine-Interaction module, a symbolic planner, a fact manager, and an execution module. Furthermore, the robot uses a database which contains all necessary information about the environment. Although this architecture contains explicit a Man-Machine-Interaction module and is used for different (but similar) robots, it seems not to be easy to create new applications, because some important parts (like to symbolic planner and the fact manager) have to be modified for each new task.

3. CONTROL ARCHITECTURE FOR MOBILE INTERACTION-ROBOTS

In the following section, we will define our requirements to a control architecture for mobile interaction-robots and introduce our system in detail.

3.1. Requirements

A modern robot control architecture has to fulfill the following common demands:

- *Modularity*: The different modules of the architecture must be functional independent and exchangeable.
- *Extensibility*: The architecture must be easily extensible with new modules.
- *Transparency*: An exchange or a modification of a single module must be transparent to the other modules.
- *Portability*: The architecture should be able to run on different robot platforms.
- *Efficiency*: The architecture must be able to run in real-time on the underlying robot.

Based on the introduced problems with existing control architectures, we derived the following additional demands on a control architecture for a mobile interaction robot:

- *Rapid Application Development*: quickly generation of a new application
- *Customizability*: easy generation of new applications also by non-programming users and
- *Reusability*: easy reuse of a generated application by different robot systems with different hardware components.

3.2. Basic Structure of the Architecture

To build a real robotic application which can fulfill all these demands, it is necessary to separate the robot-specific methods and skills (e.g. collision avoidance or people detection) from the application itself (e.g. a robot as an office guide or a robot as a shopping assistant). Doing this way, the demands *Customizability* and *Reusability* can be easily fulfilled. To bring these different parts together, an abstraction layer in-between is necessary. As the result, our developed architecture consists of four layers (see figure 1 and 2.).

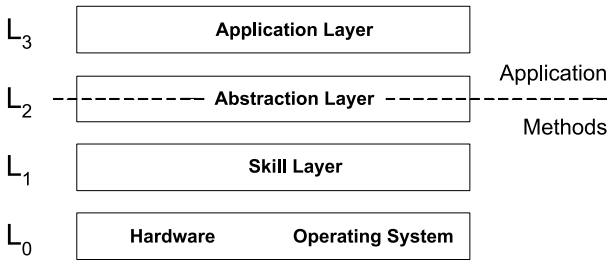

Fig. 1. **The four layers of the architecture.** The *Abstraction Layer* L_2 separates the methods from the application.

The layer L_0 (*Hardware Layer*) encloses the robot hardware (sensors and actuators), the operating system, and the low-level interface to the hardware. The low-level sensor information will be processed in the next higher level to provide different skills, which will be executed in L_0.

In the next layer L_1 (*Skill Layer*) all required classical robotic-specific methods are located. Typically, these are

modules for collision avoidance, localization and navigation, speech recognition, speech synthesis, a people- or object tracking and so on. These different robot-specific methods and skills are reusable for numerous different applications. The functionality and complexity of layer L_1 depends on the underlying robot.

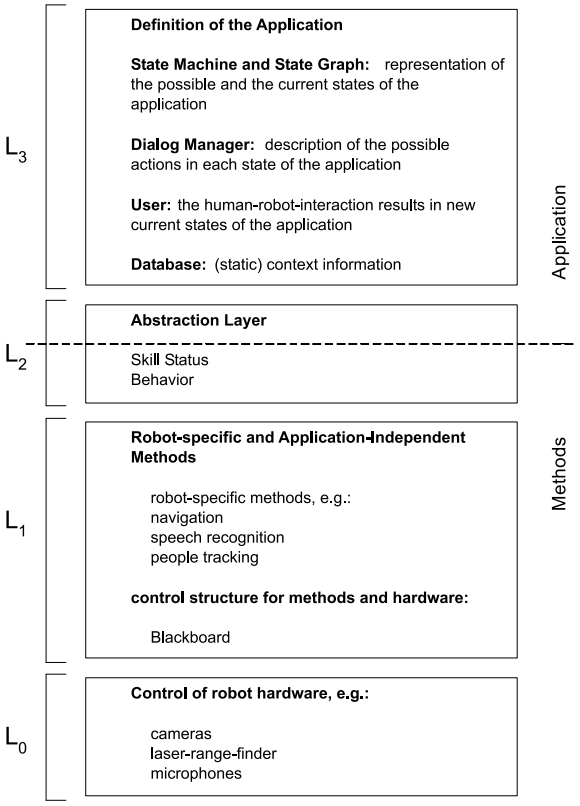

Fig. 2. **The main components of the architecture:** the method-level with robot-specific methods, the application-level with a robot-independent application and a robot-specific interface to adapt the current application to the currently used robot.

The layer L_2 (*Abstraction Layer*) generalizes from the robot-specific skills of L_1 and provides a high-level interface to the capabilities of the robot. The skills (like navigation and people tracking) are combined in this layer to a set of high-level *Behaviors* (like people guidance). To control the skills of L_1 and to get a feedback about the execution of the different skills the so called *Skill Status* is used.

The highest layer L_3 (*Application Layer*) provides elements, which are required for a specific application of a mobile interactive robot.

An important fact is, that all layers are only communicating with their immediate neighbours. This guarantees the transparency between the different layers.

By using this strict separation, for a new application or for the usage of an alternative robot system only the application layer L_3 or the abstraction layer L_2 have to be changed rather than the whole system. Thus, the introduced demands *Rapid-Application-Development*, *Portability* and *Reusability* can be fulfilled. Furthermore, to allow the generation of an application also by a non-programming user (*Customizability*), we use parameterizable elements in the application layer (the state graph and the dialog manager). In the following section, we discuss the specific components of the control architecture in detail.

3.3. Specific Elements of the Architecture

As shown in figures 1-3 our control architecture incorporates a *Skill Layer* L_1 and an *Application Layer* L_3, which are connected by an *Abstraction Layer* L_2.

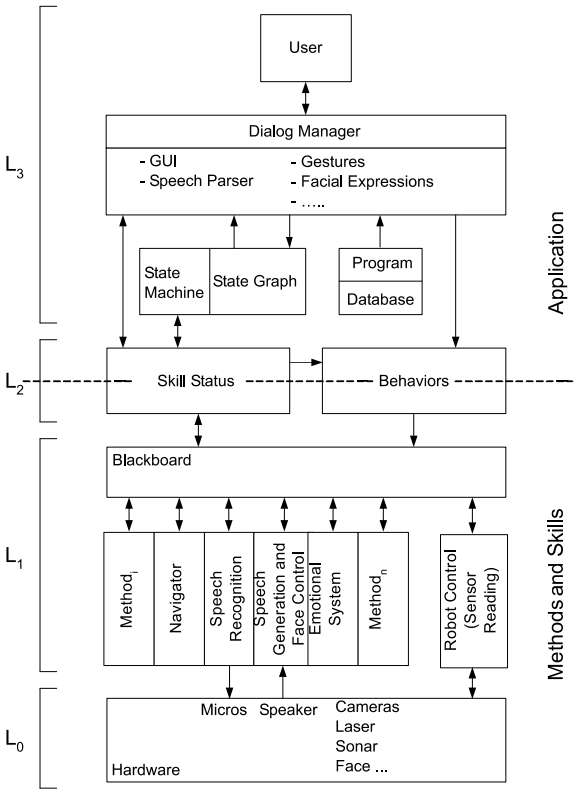

Fig. 3. **The four layers in detail.** Specific elements of the four layers of the control architecture.

The *Skill Layer* consists of two main components: The robot-specific and application-independent methods and a *Blackboard System*. In the context of a mobile interaction-robot the methods should include at least a navigation strategy, a people detection module, a speech recognition and a speech generation (see figure 3). The realization of such methods mainly depends on the hardware components (the *Hardware Layer*) of the underlying robot system. For instance, a robot which is equipped with sonar sensors will

use a different navigation method than a robot equipped with a laser-range-finder.

Furthermore, for the communication between all different methods we use a *Blackboard System* [11]. With the blackboard it is possible to share all required information between the different methods. Therefore, the blackboard can be considered as general shared data memory. The blackboard structure makes it easy for the programming users to integrate new modules or to make modifications on existing modules. The programmer only must ensure, that the interface from his module to the blackboard keeps consistent. Therefore, with a blackboard system the demands *Modularity*, *Extensibility* and *Transparency* can be easily fulfilled.

The available robot-specific methods of L_1 will be combined in the *Abstraction Layer* to a set of high-level macro-behaviors. Each of these behaviors uses a set of available skills of L_1. Typically, the available behaviors are exclusive, i.e., only one behavior can be executed at a time. As a result, the layer L_2 provides a set of high-level macro-behaviors, which can be executed without knowledge about the underlying skills of L_1. Thus, it is very easy for non-programming users to use the robot. For instance, in a state "Create Attention" the desired behavior could be defined as "Face the current user". Depending on the used robot a specific system could activate the (robot-specific) person tracker and track a person by moving the pan-and-tilt camera unit. Another system without a pan-and-tilt camera unit could also track the perceived person by moving the whole robot body. So the *Behaviors* and *Skill Status* translate the application-specific behaviors in the robot-specific methods. The *Skill Status* is also used to get a feedback of the execution of the underlying robot-specific method. For instance, if a navigation task can not be fulfilled (because the way is blocked), the state machine or the dialog manager can react in an application-depended way and solve the conflict.

Depending on the general application which has to be realized, the available behaviors of L_2 will be combined in the *Application Layer*. This means, creating a new application means to combine the available macro-behaviors in a new kind. In praxis, this process of combination is mostly large-scaled and requires a lot of new programming which can be typically done only by people knowing the underlying robot-specific methods and their realizations. In order to simplify this, we propose the separate application-level depicted in figure 3. The elements of the *Application Layer* include a state graph (interpreted by a state machine), a dialog manager, the user (as the interaction partner of the robot), and a database, which contains the information about the environment. Thereby, the state graph incorporates the principle states of the application. For instance, the application "information system" could be defined by the states "Wait" (when no user is perceivable), "Create Attention" (when a user is perceived) and "Dialog" (when the user wants to interact) (see section 4.2 for details). Further, each state also defines a principle behavior which will be executed if the system is in this specific state. Each state is connected with at least one other state. These connections define specific conditions that have to be fulfilled to get into a state. Thereby, the conditions result from the robot-specific methods (via the *Skill Status*) or from the user (via the dialog manager). The resulting whole definition of a state graph is based on a XML-file. So an application is more configurable than programmable, and in the consequence pretty easy to generate also by non-programming people. By defining only this state graph in the application-level, already a whole application can be generated. Therefore, the mentioned demands *Rapid-Application-Development*, *Customizability* and also *Reuseability* are fulfilled.

3.4. The State Machine and the Dialog Manager

In our proposed architecture, the robot will be controlled by the state machine and the dialog manager. The state machine makes a suggestion, which default behaviour should be executed in L_2 in the current state. The dialog manager can simply send this behavior to the Abstraction Layer or select another behavior. By doing so, the desired behaviors in each state can be be chosen more flexibly. For instance, in the state "Create Attention" the state graph defines a principle behavior which will be executed if no other behaviors are defined in the dialog manager. Thus, the dialog manager could also define, e.g., numerous different speech outputs to attract the user's attention in this state. Furthermore, the dialog manager can also realize, that in the context of the current application the most promising behavior among all possible behaviors in a specific state can be learned. In the next version of the control architecture, the dialog manager will also be configurable like the state graph. In the current version the dialog manager is still to program.

4. APPLICATIONS

We use the proposed control architecture for our mobile interaction-robot HOROS (see section 4.1), and we are also currently engaged to transfer this architecture to our robot PERSES, a shopping assistant robot (see section 4.3). Subsequently, we discuss one part of the task of HOROS, to be an office information and guiding system for employees, students, and guests of our institute, in the context of the proposed control architecture.

4.1. Robot System HOROS

The hardware platform for HOROS is a Pioneer-II-based robot from ActiveMedia. It integrates an on-board PC (Pentium M, 1.6 GHz, 512MB) and is equipped with a laser-range-finder and sonar sensors. For the purpose of HRI, this platform was extended with different modalities (see Figure 4).

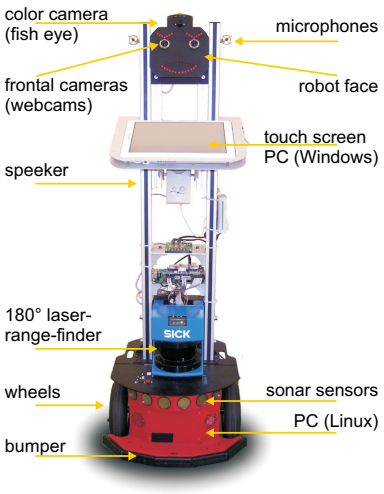

Fig. 4. **The HOme RObot System**. Sensory and motory modalities of the mobile interaction robot HOROS.

This includes a tablet PC (Pentium M, 1.1 GHz, 256MB) for touch-based interaction, speech recognition and speech generation. It was further extended by a robot face which includes an omnidirectional fisheye camera, two microphones and two frontal webcams for the visual analysis of dialog-relevant user features (e.g. age, gender, emotions).

4.2. The Control Architecture in the Context of a Survey Task

The office application of HOROS includes a survey task, which will be discussed in the context of the control architecture. Thereby, HOROS is standing in a hallway in our department. His task is to attract attention of people that came by. As soon as the system recognized a person near him, the robot addresses the visitor to come nearer. He then offers to participate in a survey about the desired future functionality of HOROS. Further, a people tracking module is used to detect break offs, thus if the user is leaving before finishing to survey, the robot tries to make them came back and finalize the survey. After the successful completion of the interaction or a defined time interval with no person coming back, the cycle begins again with HOROS waiting for the next interaction partner. The experiment was made in the absence of any visible staff members, so the people could interact more unbiased. The respective state graph as an element of the Application Layer L_3 is shown in figure 5.

Each state of the state graph has also some defined input conditions. So if the specific input conditions of a state are fulfilled the application will get into this state. In each state, all outgoing conditions must be consistent. Exemplary, the incoming and outgoing conditions for the state ”Create Attention” are depicted in figure 6. This state can be reached

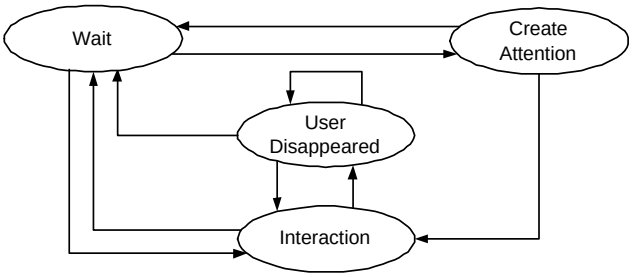

Fig. 5. **State graph for the survey task.** The state graph consists of only four states which already define the principle application.

only from the state ”Wait” (see figure 5) by the following incoming conditions: the application was longer than 30 seconds in the state ”Wait” (the last interaction partner of the robot left the surroundings of the robot) and at least one new user is perceived. In figure 6, two outgoing conditions are also depicted. These are as well the incoming conditions for the state ”Wait” (the perceived user left the surroundings without an interaction or the perceived user came not closer to the robot within 15 seconds) and for the state ”Interaction” (at least one perceived user came closer than 0.75 meters to the robot).

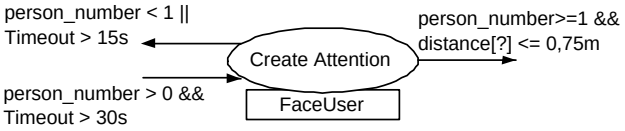

Fig. 6. **State ”Create Attention”.** There is one incoming condition and two outgoing conditions in the state ”Create Attention”. Further, when this state will be reached the general behavior ”Face User” will also be activated.

The state ”Create Attention” is also defined by a general behavior ”Face User”, that will be executed if the application reaches this state via fulfilled input conditions. This behavior will be routed to the dialog manager next. If there are no other defined behaviors for this state in the dialog manager, ”Face User” will be sent back to the Abstraction Layer L_2, then to the Skill Layer L_1 and subsequently will be executed by the Hardware L_0.

Simultaneously to these processes, the robot permanently perceives its environment in the Hardware Layer L_0. Using the respective sensor readings in the Skill Layer L_1 the used methods and Blackboard variables are updated. Consequently these Blackboard variables can also result in an updated Skill Status in the Skill Layer L_2 and subsequently in a newly activated state of the state graph. State transitions in the state graph can also be caused by user inputs, e.g. via the GUI in the dialog manager.

Another application of HOROS based on our control ar-

chitecture is a guidance function also in the context of our office scenario as described in [12]. The robot can guide visitors from the entrance of the building to the rooms of staff members and give information about the possible whereabouts of staff members.

4.3. The Control Architecture in Context of a Shopping Assistant

Another of our robots is PERSES, which works as an interactive mobile shopping assistant [13]. At the moment we are working on the *Abstraction Layer* for PERSES. As soon as this work is finished, it will be quite easy to create the shopping assistant application in our architecture, although PERSES (a B21r robot) is totally different to HOROS (a Pioneer-II-based robot).

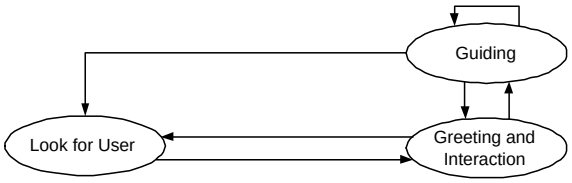

Fig. 7. Basic state graph for the shopping assistant.

Figure 7 shows the possible basic structure of a state graph, which is necessary for PERSES. Of course, this graph has to be completed with different sub-graphs and conditions in the individual states. Using such a state graph, a simple dialog manager and an appropriate database, PERSES will be able to work as a shopping assistant.

5. SUMMARY AND CONCLUSIONS

In this paper, we described a control architecture for mobile interactive robots. Our architecture fulfills the demands on modern architectures, like modularity, extensibility, portability and efficiency. Based on our architecture, it is very easy for programming users to integrate new modules or to modify existing parts. Furthermore, our architecture also is designed to allow non-programming users to develop robot applications. It guarantees the rapid application development and the reusability of existing applications. We illustrated the usability of this concept in two typically applications (the survey task and the shopping assistant).

In the future we will extend our architecture with a configurable and adaptive dialog manager, which will it make still easier for non-programming user to develop new robotic applications.

6. REFERENCES

[1] Evolution Robotics GmbH. Homepage: `http://www.evolution.com`.

[2] Coste-Maniere, E. and Simmons, R. Architecture, the Backbone of Robotic Systems. In *Proc. IEEE Internat. Conference on Robotics and Automation*, 2000.

[3] Nesnas, I.A., Wright, A., Bajracharya, M., Simmons, R., Estlin, T., and Kim, W. CLARAty: An Architecture for Reusable Robotic Software. In *SPIE Aerosense Conference*, 2003.

[4] Vaughan, R., Gerkey, B., and Howard, A. On device abstractions for portable, resuable robot code. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, 2003.

[5] Montemerlo, M., Roy, N., and Thrun, S. Perspectives on Standardization in Mobile Robot Programming: The Carnegie Mellon Navigation (CARMEN) Toolkit. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, volume 3, pages 2436–2441, 2003.

[6] Simmons, R., Goldberg, D., Goode, A., Montemerlo, M., Roy., N., Sellner, B., Urmson, C., Schultz, A., Abramson, M., Adams, W., Atrash, A., Bugajska, M., Coblenz, M., MacMahon, M., Perzanowski, D., Horswill, I., Zubek, R., Kortenkamp, D., Wolfe, B., Milam, T., and Maxwell, B. Grace: An autonomouse robot for AAAI robot challenge. In *AAAI Magazine, vol. 24, no. 2*, pages 51–72, 2003.

[7] Bischoff, R. Towards the Development of "Plug-and-Play" Personal Robots. In *IEEE-RAS International Conference on Humanoid Robots*, 2000.

[8] Arkin, R.C. *Behavior-Based Robotics*. MIT Press, Cambridge, MA, 1998.

[9] Zobel, M., Denzler, J., Heigl, B., Nöth, E., Paulus, D., Schmidt, J., and Stemmer, G. MOBSY: Integration of Vision and Dialogue in Service Robots. In *Computer Vision Systems, Proceedings Second International Workshop (ICVS)*, pages 50–62, 2001.

[10] Hans, M. and Baum, W. Concept of a Hybrid Architecture for Care-O-Bot. In *Proceedings of ROMAN-2001*, pages pp. 407–411, 2001.

[11] Nii, H. P. Blackboard Systems. *The Handbook of Articifical Intelligence 4*, pages 1–82, 1989.

[12] Martin, C., Boehme, H.-J., and Gross, H.-M. Conception and Realization of a Multi-Sensory Interactive Mobile Office Guide. In *Proc. IEEE Internat. Conf. on Systems, Man and Cybernetics (SMC)*, pages 5368–5373, 2004.

[13] Gross, H.-M. and Boehme, H.-J. PERSES - a Vision-based Interactive Mobile Shopping Assistant. In *Proc. IEEE Internat. Conf. on Systems, Man and Cybernetics (SMC 2000)*, pages pp. 80–85, 2000.