Acromovi Architecture: Implementation of the Upper Layer

Patricio Nebot and Enric Cervera
Robotic Intelligence Lab
Dept. of Computer Science and Engineering
Univeritat Jaume I
Campus de Riu Sec, E-12071 Castellón de la Plana, Spain
pnebot@icc.uji.es, ecervera@icc.uji.es

Multiagent systems are an important tool for the development of new robotic applications. They are the natural environment for the development of applications in which more than one robot takes part. And they make possible the fast implementation of powerful architectures for the specification and execution of the tasks by those teams of robots. The Acromovi architecture is a distributed agent-based architecture for programming and controlling a team of multiple heterogeneous mobile robots that are capable to cooperate between them and with people to achieve service tasks in daily environments. This architecture is divided into two layers. The purpose of this work is to describe the agents that form the upper layer of the Acromovi architecture. In this layer there is a set of agents addressed to supervise and control the access to the agents of the lower layer.

1. INTRODUCTION

Multiagent systems are an important tool for the development of new robotic applications. They are the natural environment for the development of applications in which take part more than one robot. And they make possible the fast implementation of powerful architectures for the specification and execution of the tasks of those teams of robots.

As has been shown in previous works and papers [9] [10], the Acromovi architecture was born from the idea that teamwork is an essential capability for a group of multiple mobile robots. Having one single robot with multiple capabilities may waste resources. Different robots, each one with its own configuration, are more flexible, robust and cost-effective. Moreover, the tasks to achieve may be too complex for one single robot, whereas they can be effectively performed by multiple robots.

Acromovi architecture is a distributed architecture for programming and controlling a team of multiple heterogeneous mobile robots that are capable to cooperate between them and with people to achieve service tasks in daily environments.

This architecture has some peculiarities. It allows the reuse of code, due to it uses native components, by means of agent wrappers, providing the programmer with a set of tools for mobile robots. Also, it allows the sharing of the robots' resources among the team and an easy access to the robots' elements by the applications.

Other two important features of the Acromovi architecture are the scalability and the facility of use. Regarding the scalability, Acromovi permits that once an application has been tested, it could be converted in a new agent of the global architecture, thus, it is possible to reuse this new agent to create more complex applications. And finally, concerning to the ease of use, it is important to remark that a new application can be developed in very short time.

Moreover, Acromovi architecture is a middleware layer between the robot architecture and the applications. This middleware is based on an agent-oriented approach. It has been developed by means of the JADE multiagent development tool, and it is is composed of multiple interacting agents.

This middleware level is logically divided into two layers. In the lower layer, there are a set of components that access to the physical parts of the robots, as the sonar, the base or the gripper; and other special components that offer services to the upper layer, as the vision, the navigation or the localization.

The purpose of this work is to add to the Acromovi architecture the agents that form the upper layer. In this layer there is a set of "generic" agents aimed at controlling and supervising the access to the agents of the lower layer.

These "generic" agents have been conceptually divided into two groups. On the one hand, there is a generic agent to control those agents of the lower layer that can serve data in a continuous way. On the other hand, there are two special agents in charge of controlling those agents of the lower layer that can have a problematic access, as can be concurrent problems, and so on. The candidate agents of the lower layer to use an agent of this type are those who move a physical part of the robot.

In the following section, the most important works related with control architectures and systems of multiple robots working cooperatively are mentioned. In section 3, the Acromovi architecture is described, firstly form the point of view of the design, and the implementation of this architecture is shown afterwards. In the fourth section, the upper layer of the Acromovi architecture is described, just as the agents it is composed of. Finally, in section 5, the most important conclusions are enumerated.

2. STATE OF THE ART

Although the field of cooperative robotics has received much attention recently, this section only wants to cite a few works focused on problems which are similar to the one addressed in this work.

MICRobES is an experiment of collective robotics that tries to study the adaptation of a micro-society of autonomous robots to an environment with humans. The robots also must cohabit with the people [11].

CEBOT is a hierarchical decentralized architecture able to dynamically reconfigure itself to try to adapt to the environment changes [4]. In this architecture there are a special "cells" that coordinate the subtasks.

ThinkingCap-II [1] is an architecture developed in a distributed platform based on agents for mobile robots.

SWARM is a distributed system made of a great number of autonomous robots. This project tries to show that a system with multiple non-intelligent robots can exhibit a collective intelligent behaviour. In this case, the architecture is homogeneous and the interaction is limited to the nearest neighbours [6].

Finally, Miro is a middleware to create applications for autonomous mobile robots. It is based on the construction and use of an object-oriented middleware to make easier and faster the development of applications and to promote the portability and the maintenance of the robot software [2]. This software also provides generic abstract services, which can be applied in different robotic platforms without introducing changes.

These works implement architectures and systems to teams of robots can make cooperative tasks. Acromovi architecture is an architecture of this type, but it is important to remark some aspects like the software reusability by means of embedded agents, the share of resources among the robots of the team, and the easy access to the physical elements of the robot by the applications

3. ACROMOVI ARCHITECTURE

The presented agent-based architecture, Acromovi, was born from the idea that teamworking is an essential capability for a group of multiple mobile robots [7] [8].

As it has been mentioned above, Acromovi architecture is a framework for application development based on embedding agents and interfacing agent code with native low-level code. It addresses the implementation of resources sharing among all the group of robots. Also, Acromovi architecture is a distributed architecture that works as a middleware of another global architecture for programming robots.

It has been implemented by means of the JADE (Java Agent DEvelopment Framework) [5], a tool for the development of multiagent systems, implemented in JAVA, that fulls with the FIPA specifications [3].

The embedded agents that constitute the Acromovi architecture work as interfaces between the applications and the physical elements of the robots. Some agents also make easier the handling of these elements and provide higher-level services.

Establishing mechanisms of cooperation between robots implies to consider a problem of design of cooperative behaviour given a group of robots, an environment and a task, how the cooperation must be carried out. Such problem implies several challenges, emphasizing among them the definition of the architecture of the group. The multiagent systems are the natural environment for such groups of robots, making possible the fast implementation of powerful architectures for the specification and execution of tasks.

3.1 Architecture Design

The mobile robot has already a programming architecture, with native (C/C++) libraries, constituted by two layers. ARIA, the lower layer, and Saphira, the upper layer. But also, the Acromovi architecture is able to subsume any other extra software, like the ACTS (a colour-tracking library) and the Vislib (a frame-grabbing library).

As it can be seen in Fig. 1, the Acromovi architecture is a middleware layer between the robot architecture and the applications that allows the collaboration and cooperation among the robots within the team and allows sharing the resources of each robot among all the team.

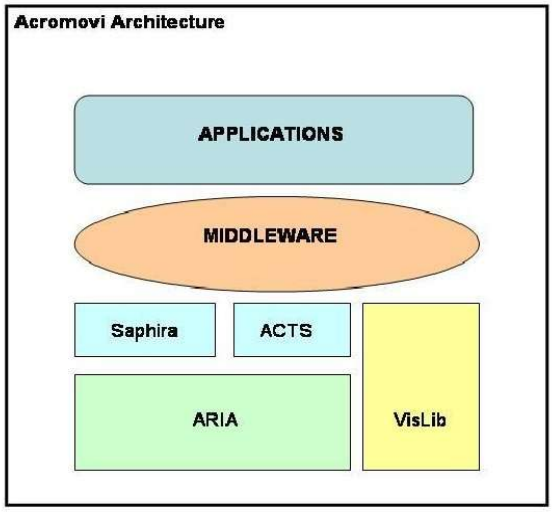

Fig. 1. General architecture diagram

This middleware is based on an agent-oriented approach. The applications layer is over the middleware layer. These applications, to access to the components of the robots, must communicate with the agents of the middleware, which access then to the bottom layer that controls the robot.

The middleware level has been divided into two layers. In the lower layer, that can be seen in the Fig. 2, there is a set of components that access to the physical parts of the robots, such as the sonar, the base or the gripper; and other special components that offer services to the upper layer, as the vision, the navigation or the localization. These components only perform two kinds of functions: requests processing and results sending.

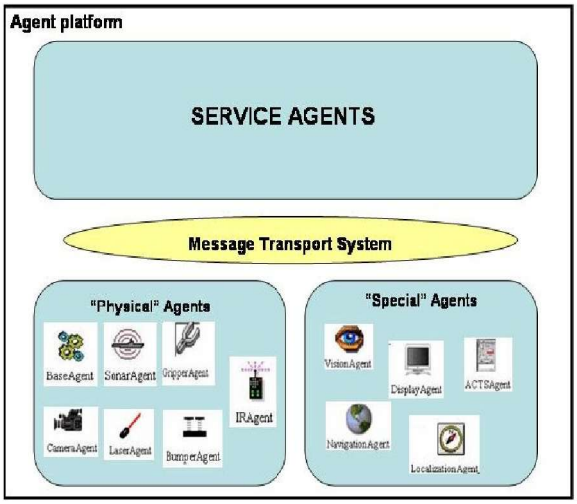

Fig. 2. The lower layer of the middleware

The upper layer, which is explained later, comprises a variety of embedded agents addressed to supervise and control the access to the agents of the lower layer, such as arbiter agents to manage the access to the different elements of the robot, filter agents to avoid conflictive actions of certain elements, and subscriber agents to serve data from one element in a continuous way. These agents also act as links between the applications and the agents that access the components of the robot.

Other important characteristic of the middleware is due to the fact that the team of robots is heterogeneous. Because of this characteristic, there are different middleware layers, depending on the configuration, for each robot of the team. These middleware layers are generated in execution time according to the elements that each robot has active at the moment of its activation.

Finally, another interesting characteristic of the overall architecture is the scalability. That is, once an application has been tested, and if it is useful for the entire architecture, it can be easily converted into a new agent of the upper layer of the middleware, increasing this way our system to make applications more difficult and more interesting, following a bottom-up design.

3.2 Architecture Implementation

Due to the fact that the architecture has been designed following an agent-oriented approach, for its implementation has been selected a multiagent systems programming tool. JADE (Java Agent DEvelopment Framework) is a framework to develop agent-based applications in compliance with the FIPA specifications for interoperable intelligent multiagent systems.

The JADE middleware implements an agent platform for execution and a development framework. And it provides a library of FIPA interaction protocols ready to be used.

Following the JADE specifications, each robot or PC involved in the architecture is a main container. In each of these containers, we have a group of agents. These agents are created in execution time depending on the robot configuration, as it has been mentioned above. Each of these agents represents one of the elements that are active in the robot in that moment. This can be seen in Fig. 3.

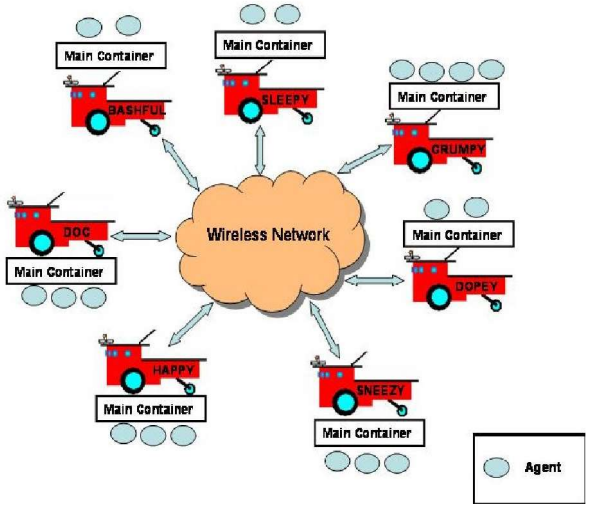

Fig. 3. Structure of the JADE implementation

The agents of the lower and the upper layers co-habit together in each of the containers. Their separation is logical, and in the level of implementation their differences lie in the agents that can communicate with them or the agents that they can communicate with.

It is important to note that in the lower layer the agents have been conceptually divided in three different groups, depending in which part of the robot the agent carries out its work. Thus, the body agents are the basic agents for the operation of the robot and manage its main elements. These agents are the Base, Gripper, Sonar, Bumper and IR agents. The laser agents communicate with the laser and implement the localization and navigation for the robot. The agents are the Laser, Localization and Navigation agents. At end, the vision agents are in charge to capture the images from the camera and process them in a correct way. These agents are the Camera, ACTS, Vision and Display agents.

In the upper layer the agents have also been divided into two groups, as it is explained below in detail. On the one hand, there is an agent in charge to control those agents of the lower layer that can serve information in a continuous way. On the other hand, there are two agents in charge to manage those agents of the lower layer that can have conflicts in its access. The agents of the lower layer that

can use these agents are those that can move physically any part of the robot, as the base, gripper o camera.

4. UPPER LAYER

The purpose of this section is to explain the configuration and the agents that form the upper layer of the Acromovi architecture explained before.

As it has been said before, in this layer there is a set of agents addressed to supervise and control the access to the agents of the lower layer. Arbiter agents to manage the access to the different elements of the robot. Filter agents to avoid conflictive actions of certain elements. And subscriber agents to serve data from one element in a continuous way.

Thus, in this layer, that can be seen in Fig. 4, only there are three types of agents, arbiter, filter and subscriber agents. It is important to remark that these agents are "generic", that is, their functionality varies depending on the agent of the lower layer that they serve. So, there can be more than one of these agents at the same time, but controlling different agents of the lower layer, in the overall architecture.

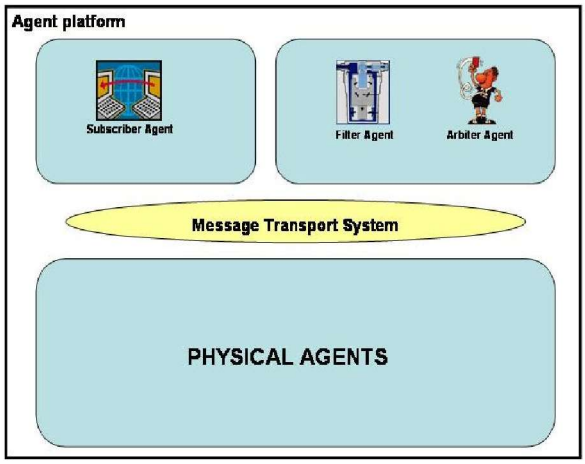

Fig. 4. The upper layer of the middleware

These three "generic" agents have been conceptually divided into two groups. The first group is formed by one generic agent in charge of controlling those agents of the lower layer that can serve data in a continuous way, such as the sonar, laser, vision, ... This generic agent is called Subscriber agent. The second group is formed by two special agents, the Filter and the Arbiter agents. These agents are designed to control the access to some «problematic» elements. Which are these «problematic» elements? They are all those elements that can have a wrong operation if one or more agents can to access and execute operations at the same time on them. These elements are the base, the gripper, the camera ... So, they try to resolve concurrency problems. Other important characteristic of these agents is that they also try to prevent the agents from performing operations with the «problematic» elements that can put in danger the robot's life. For example, if one agent wants to move the robot two meters, but there is a wall at one-meter distance.

In that way, for a robot with a laser system would have several agents in the upper layer. One Subscriber agent to manage the data coming from the Laser agent and another one to manage the data coming from the Sonar agent. Also, we have one Filter and one Arbiter agents to control the access to the Base agent, two more to control the access to the Gripper agent, two to control the access to the Localization agent, and finally two more to control the access to the Navigation agent.

The work the Subscriber generic agent is the following. When one application or agent wants to have a continuous flow of data from one of the agents of the lower layer, it has to make a subscription to the Subscriber agent. In the message of subscription, the agent must indicate how often it wants to receive that data. The frequency is specified by a number which indicates in which cycles of reading the agent wants the data. That is, if the agent sends 2, each 2 readings of the data, the Subscriber agent sends this data to the corresponding agent. Other agents can make a different petition, the Subscriber agent differentiates them and sends the data to each agent when corresponds.

The Subscribe agent is continuously requesting information to the agent which it is serving. So it always has the actual values of this agent. The rate in which Subscriber agent requests the information to the served agent is given by the real rate at which the agent can serve the information.

Finally, the work that perform the Arbiter and Filter agents is described. Firstly, the Filter agent should not allow an agent to realize an action that could be in danger the life of the robot or element that it must to care. That is, if one agent wants to move 3 meters but there is a wall in 2 meters, the Filter agent must to avoid this action. So, each action that an application or agent sends to one of the "problematic" agents must be seen by the Filter agent and if it can cause a danger in the robot, the Filter must avoid this action.

On the other hand, the Arbiter agent is in charge of distributing the access to the agents. This agent is in charge of giving permission to the other agents to access to the resources of the agent that it manages. That is, if one agent wants to use the services of the Base agent, it must ask the Arbiter agent in charge to manage the Base agent if it can do it. If the Arbiter gives it permission, the agent can access to the Base agent to make the corresponding operations. At the same moment that the Arbiter agent gives permission to a certain agent, it sends a message to the Filter agent so that it can control the messages that are sent by the agent with permission to the agent that controls the element. When the agent finishes its work, it informs of it to the Arbiter agent. When the Arbiter agent receives the notification, it can give permission to other agents.

5. CONCLUSIONS

In this article, it has been shown how the upper layer of the Acromovi architecture has been implemented and formed.

Also, it has briefly explained the Acromovi architecture, which is the support for all the programming and interaction of the agents that manages our team of robots.

The Acromovi architecture is basically an agent-based distributed architecture for the programming and controlling of teams of mobile robots. This architecture allows code reuse by means of the use of native components, providing the programmer with a set of ready-to-use of tested and efficient agents.

Also, the presented architecture allows the scalability of the system. That is, if one tested application is of interest for the whole system, it can easily converted in a new agent of the architecture, to serve as basis for new applications more complex.

Moreover, this architecture implements another concept, the resources sharing. This means that all the elements of one of the robots of the team can be easily accessed by all the other robots of the teams by means of the agents that control each of the other robots.

Regarding the upper layer, it is formed by a set of agents addressed to supervise and control the access to the agents of the lower layer.

The Subscriber agent is in charge of serving a continuous flow of data of a certain element of the robot to the rest of agents of the architecture. This agent also adds the possibility to indicate how often the provider agent wants to receive the data of interest for its operation. Also, the provider agent maintains a list with all the subscriber agents and their frequency of data, and serves to any agent when it corresponds.

The Filter and Arbiter agents allow to control the access to certain elements of the robot and filtering actions that can put in danger the robot. The Arbiter agent gives permission to access to a certain element of the robot to the petitioner agents in order of arrival of petitions. The Filter agent controls all the actions that the agents send to the element and avoid those that can put in danger the life of the robot.

With these new agents, it has been overcome some problems present on the Acromovi architecture. With the Subscriber agent, it has been solved the problem that appears when one agent needs a continuous flow of data from other agent. The Filter and Arbiter agents avoid a more important problem in the distributed systems, which is the concurrency problem in the access to a certain resource. The Arbiter agent solves that problem by giving the permission to the agent to access to a certain element. That is, it is insured that only one agent access to the element at each time.

6. ACKNOWLEDGMENTS

This work has been partly funded by project GV05/137 from the Generalitat Valenciana.

7. REFERENCES

[1] Cáceres, D. & Martínez, H. & Zamora, M. & Balibrea, L. (2003). A real-time framework for robotics software. International Conference on Computer Integrated Manufacturing (CIM-03).

[2] Enderle, S. & Utz, H. & Sablatng, S. & Simon, S. & Kraetzschmar, G. & Palm, G. (2001). Miro: Middleware for autonomous mobile robots. IFAC Conference on Telematics Applications in Automation and Robotics.

[3] FIPA, The Foundation for Intelligent Physical Agents (2005). Available from: http://www.fipa.org, Accessed: 2005-07-13.

[4] Fukuda, T. & Iritani, G. (1995). Construction mechanism of group behavior with cooperation. IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS).

[5] JADE, Java Agent DEvelopment Framework (2005). Available from: http://jade.cselt.it/, Accessed: 2005-07-10.

[6] Johnson, J. & Sugisaka, M. (2000). Complexity science for the design of swarm robot control systems. 26th Annual Conference of the IEEE Industrial Electronics Society (IECON).

[7] Jung, D. & Zelinsky, A. (1999) An architecture for distributed cooperative planning in a behaviour-based multi-robot system, Journal of Robots and Autonomous Systems.

[8] Mataric, M. J. (1998). New directions: Robotics: Coordination and learning in multirobot systems, IEEE Intelligent Systems.

[9] Nebot, P. & Cervera, E. (2005a). A Framework for the Development of Cooperative Robotic Applications, Proceedings of 12th International Conference on Advanced Robotics (ICAR 2005).

[10] Nebot, P. & Cervera, E. (2005b). Agent-based Application Framework for Multiple Mobile Robots Cooperation. Proceedings of IEEE International Conference on Robotics and Automation (ICRA 2005).

[11] Picault, S. & Drogoul, A. (2000). The microbes project, an experimental approach towards open collective robotics. Proceedings of the 5th International Symposium on Distributed Autonomous Robotic Systems (DARS'2000).