A Cooperative Architecture to Control Multi-Agent based Robots

R. Lafrenz, N. Oswald, M. Schulé, and P. Levi

University of Stuttgart, IPVR
Breitwiesenstr. 20-22, D-70565 Stuttgart, Germany
robocup@informatik.uni-stuttgart.de

Abstract. To control a group of cooperating autonomous mobile robots, we use a multi-agent architecture. The structure of a single robot consists of a set of concurrent elementary agents situated on different levels of abstraction. Their internal design and interplay are described. Then the system architecture for a group of cooperating robots is presented by an example of a concrete application. In this context the relation between behaviors, tasks, and strategies is discussed. Obviously, the latter have fundamental influence on cooperation.

1 Introduction

In robotics, two fundamentally architectures have emerged that support different aspects, functional and behavioral ones. The functional approach that is going back to [AML87] supports mainly the planning components. In the behavioral approach [Bro86] the reactive behavior of a system is emphasized. Recent approaches try to combine both approaches [vB95,BD94,CCC93].

The architecture suggested in [Lev89,LBLM98] has been successfully used in some real applications like deadlock avoidance in a traffic environment [Rau98], cooperative object recognition [OL97] or cooperative transport of a workpiece with several mobile robots [BML97]. Motivated by these successes we are now going to transfer the concept of multi-agent architecture to soccer playing robots.

In earlier applications with often natural scenarios, the level of abstraction in the sense of the architecture was not very high because of a narrow focus on solving the given task. In contrast, the soccer scenario with its artificial world offers a testbed for a completely developed architecture. Furthermore, it is a scenario that emphasizes both the reactive behavior and the planning components. In addition, there can be various aspects of cooperation at different levels of abstraction.

It is important to solve the tasks straight forward, but to do so, these tasks must be determined. Therefore a team of robots needs a strategy which in turn depends on the current situation of the game. To fulfil a strategy, it is divided into tasks, which in turn are solved by different behavior patterns.

In section 2 the building of a concrete system architecture for the soccer scenario consisting of several elementary agents is shown. The architecture of our elementary agents is presented in section 3. These elementary agents form a robot agent that is able to cooperate with other robots. Behavior patterns, tasks and roles for soccer playing robots are introduced in section 4. Finally in section 5 the results of the last two years work is presented.

2 Internal Structure of a Robot

In this section we show the system architecture of a single robot, which has to act as a team member within a group of autonomous robots. We chose robot soccer (RoboCup) as the application scenario, because it is a well-defined context with suitable complexity.

As mentioned before, we use a multi-agent-system to control a robot. The design of a single (elementary) agent is described in 3. Each elementary agent is responible for a hardware component or a task.

The overall architecture of a soccer player is shown in figure 1. The rectangles correspond to elementary agents. In the implementation, some agents are put together in one process. The details are explained in 5. Logically, the system in structured in three layers: strategical, tactical, and reflexive layer. These layers are defined by different response times. The assignment of an agent to a layer is not reflected in the curent software, but helps to understand and design the system. The agents in the refexive layer guarantee immediate response. They are responsible for the control of the robot hardware. The agents in the tactical layer have to make their decisions within parts of a second. A typical representative of an agent in this layer is the navigator, which is described later on.

The Reflexive Layer The elementary agents in the reflexive layer control the hardware of the robot and perform sensing and acting, e.g. *Camera* or *Vehicle Control*. Other kinds of *elementary agent*s in the reflexive layer fuse the data of different type or transform sensory data into information, e.g. the *Object Recognition*.

The *Pilot* has to carry out the orders given by the Navigator. As an agent, the Pilot is responsible for leading the robot to a given destination. It has to avoid collisions, and needs access to the Obstacle Detection. It is able to perform minor coorections to the path plenned by the Navigator.

To perform the driving, the *Vehicle Control* accepts commands from the Pilot. It is able to control the driving parameters like acceleration and velocity as well as the direction of the motion. It also provides dead reckoning by the means of wheel encoders yielding odometry information. This means, that the Vehicle Control agent is both, a sensor and an actuator.

To observe the scene and react properly to the situation, we need sensory information about the current scene. To get this, we provide three kinds of sensors: *Camera*(s), *Laser Range Finder* (LRF), and *Ultrasonics* (US). In addition, the Vehicle Control provides odometry information.

All these sensory agents control the underlying hardware, and perform the processing of the sensory data. This processing range from simple transformation

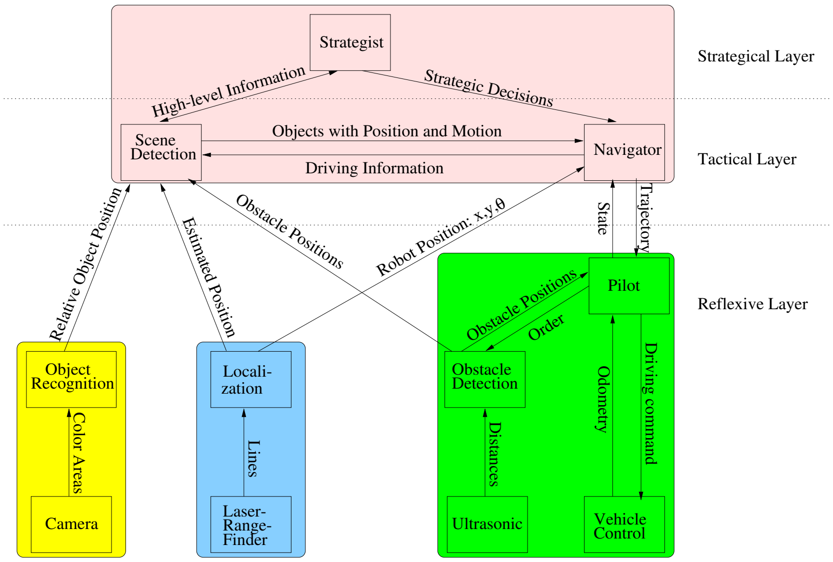

Fig. 1. System architecture

(e.g for odometry) up to complex algorithms, like line extraction and matching in the laser agent. One or more cameras are used to recognize the position of the ball and the heading direction by determining the colors of the goals. The *Object Recognition* delivers information about the ball, the opponents and proponents, and the goal by evaluating the Camera's output. This information is used to determine e.g. the ball position relative to the observing robot. The Laser Range Finder (LRF) is used for self-localization and for finding the position of obstacles (team-mates or opponents) in the field. The camera output is used by the object recognition, while the obstacle detection fuses the LRF and the US data. Ultrasonics are used for collision avoidance as well as for additional Environment modelling. In comparison to LRFs they enable the detection of obstacles in heights other than the very limited range of the LRF. The *Obstacle Detection* is responsible for detecting the position of the goal, the field and the players of each team. It gets its input from the LRF and the ultrasonics. The Localization uses odometry, laser scanner, and ultrasonics to determine the robot's position within the field. The Localization can derive the position just using a laser scan or can correct an inaccurate, odometry-based position estimation by using ultrasonics or the laser scanner. The Localization provides its information to the Pilot on request. Every new position information is given to the Scene Detection to enter it into the scene map.

The Tactical Layer In the tactical layer we find two agents, the *Scene Detection* and the *Navigator*. The Scene Detection is responsible for storing and updating the information about the current situation. As an agent, the Scene Detection collects all relevant information. It answers requests of the Navigator by using the knowledge stored in a scene map. Besides the given model like the field boundaries including the goals, positions and motion vectors of dynamic objects are stored in the scene map. These are the ball, opponents and proponents.

To generate knowledge, the Scene Detection deduces all incoming information and represents it in a global coordinate system. For example, the relative ball position, given by the Object Recogition, which is described later on, is transformed into global coordinates by using the position of the robot, given by the Localization. In case of incomplete or outdated information the Scene Detection asks all involved agents to deliver the most recent information or calculates a prediction.

Nevertheless, not all information about dynamic objects is available at any time. In order to complete the knowledge, further information is required. There are different ways to gain missing information. Depending on the source we distiguish between direct and indirect information. We call an information direct, if it can be derived from the robot's local sensory data only. The Scene Detection may also request information from the local maps of other robots which are stored in their corresponding Scene Detection. To do this, it communicates with the Scene Detection of other robots. The exchanged information is called indirect. For example, if a robot doesn't see the ball, it can use information about the ball position given by other robots' Scene Detection. Another advantage of communication is the fault-tolerance of the whole system. By exchanging the information, inconsistencies between the scene models are detected.

One of the basic requirements for that is the ability to distinguish between team-mates and oppenents. By assuming same shape and colors for robots of both teams, this distiction is impossible without cooperation, in other words, indirect information. But, by exchanging position information for all robots within the team, the recoginized objects can be identified as team-mates or opponents. In addition, the position estimation of a team-mate can be verified.

In order to act properly in the current situation, the Navigator asks the Scene Detection about the surroundings and the recognized objects. Depending on this knowledge, a behavior is chosen out of a set of possible behaviors, according to the strategy (see section 4). Each behavior implies a plan skeleton, which is instantiated using the knowledge provided by the *Scene Detection*. This instantiation considers all relevant knowledge, performs path planning, and generates a trajectory. This is given to the *Pilot* as an order. Trajectories consist of a set of coordinates ending in the destination. While the plan is executed, and thus the robot is driving on the given trajectory, the *Navigator* permanently analyses the scene to be able to correct the plan on execution or to stop plan execution and set up a new one.

The Strategical Layer In our concept, there is only one agent placed in the strategical layer: the *Strategist*. It reacts to infrequent external events (e.g. start of game, goals) and decides on the way of further playing. Thus it decides on the long-term strategies of the game, for example offensive or defensive playing mode. Therefore, it communicates with two other agents, with are placed in the tactical layer: the Scene Detection and the Navigator. In addition, the Strategist is able to communicate with the strategists of other players in the team. Sometimes a single player is not able to recognize the current situation. Therefore we provide communication beyond the boundaries of a single robot. In principle, any elementary agent is able to communicate with any other agent. Hence, very complex and powerful negotiations are possible.

3 Structure of elementary agents

The architecture of the *elementary agent* [Mus00] can be lead back to the *autonomic cyle* of [Lev89,Lev92]. The elementary agents are the autonomous function moduls of the system. They get their functionality from their plans. These plans control the interplay of the sensing and acting. The *body* of the elementry agent is supervised by the *decision unit*.

The behavior of an elementary agent is determined by the plans wich are executed in its body. These plans correspond to control loops for the sensors and actuators. In the body of the elementary agent a plan is executed in form of a program loop, which can be associated to one or more type of cycles.

1. The *react-cycle* is the simplest form of a control loop. It consists of predefined control parameters and one fixed control algorithm. The controller computes continuously the manipulated variables in such a manner that a controlled variable is adapted to the reference variable.
2. The *adapt cycle* enables the elementary agent to learn. It adapts the control parameters, which are managed in the component *model*.
3. The *behave cycle* is able to adapt to varying control paramters and to change the control algorithm. For this it selects plan skeletons out of the component *plan* and replaces it with its present control algorithm.
4. The *plan cycle* has the most distinct cognitive abilities. It is able to create new plan skeletons, i.e. new control algorithms.

The *decision unit* as *head* is the steering unit of the elementary agent. The task of the decision unit is to monitor the plans executed in the body. Therefore each plan collects information about its occupied and used resources. The decision unit coordinates the utilization of resources within the elementary agent, and is able to inhibit or release some cycles. It assigns the consumed resources (e.g. CPU-time). Beyond this it is responsible for the coordination with other elementary agents. It represents therefore one elementary agent as a set of services. This is a pre-condition for the takeover of a role in an interaction operation.

The set of services which describe the abilities of an elementary agent is defined through its plans. At runtime a role is in contact with the plans. This is done through its services. The plans set up the functionality of the services. The services enable a role to excert on execution of a plan.

Through takeover of a role and through delegation of decision competence to this role, it is posible to coordinate with other *elementary agents*. This means, that the decision competence is increased through the takeover of a role.

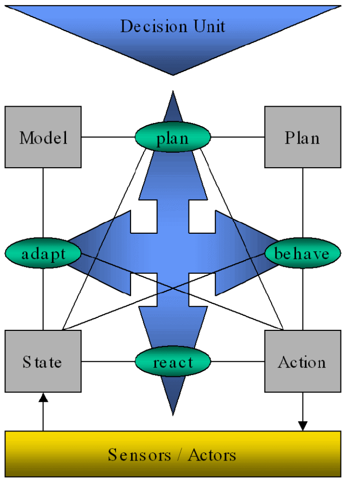

Fig. 2. Structure of an *elementary agent*

Fig. 2 shows the structure of an elementary agent. It consits of the following components:

- The *decision unit* is responsible for negotiation between the agents and for internal configuration.
- Data is transposed through seperate data channals. The configuration of this channals is done by the *decision unit*.
- *State* contains the current state of the elementary agent.
- *Model* administrates the statical or lernable knowledge.
- The internal cycles range from simple and fast react behavior (controller) to complex planning of each step using the knowledge of the *model*.
- *plan*
- *action*

4 Behavior patterns for soccer playing robots

In this section, we describe the application of the introduced architecture to a concrete example. We chose soccer playing, because it is a well-defined scenario with suitable complexity. Another advantage of this scenario is the possibility to directly compete with other approaches. Nevertheless, this architecture can also be applied to other scenarios like in [OL97,BML97,Rau98].

Roles for Teams of Robots In general, a team of robots is composed of n different types of players $P = \{p_1, \ldots, p_n\}$ like goalkeeper, forward, defender and spare man. To a type of player p_i a number m_i of different behaviors $B_i \subseteq b = \{b_1, \ldots, b_m\}$ is assigned. For a behavior b_j, a set of skills $S_j \subseteq S = \{s_1, \ldots, s_k\}$ is required. We call the set of behaviors for one type of player a role $r_i = \bigcup_j B_j \forall j$ with $s_j \in S_i$. With this formalization, a player type is equivalent to a role, and thus only determined by its behaviors.

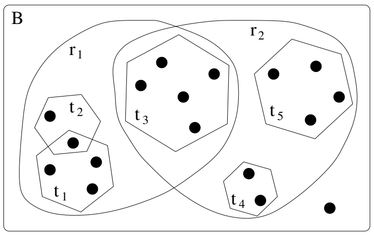

Fig. 3. Skills, behaviors, and roles

Instead of partitioning our team into limited number of types of players, we only have to define a set of general behaviors for all players. As a consequence of that, up to $m!$ roles can be defined. To visualize this description, consider fig. 3 containing 17 skills (black dots) and two roles, r_1 and r_2. The roles consist of the behaviors b_1 to b_5. Each behavior needs several skills, e.g. behavior b_3 needs five skills. In principle, a behavior or skill may be part of several roles. To create a new role, only a new set of behaviors and the possible sequences of them have to be selected.

As long as the hardware requirements are identical, a team can be created by the use of general-purpose players. In contrast to the field players, the goalie often has to move perpendicular to its sensing. To be able to react quickly, different hardware setup is required. The kicking device is also, unlike the others', mounted perpendicular to the driving direction. So we do have two types of robots for soccer playing: the fielders and the goalkeeper.

Behaviors for Soccer Playing Robots There are two kinds of behaviors, one for individual player, and one for a group of robots. A single player is able to play any role out of the set of roles R. A role contains several behaviors as listed in table 1, which could be considered to be offensive or defensive. Such a distinction facilitates the understanding and design of roles, but is not needed in principle.

Now we take a look at the behaviors for a group of robots. To achieve one of these behaviors, all players must transfer the selected group behavior into single behaviors. The set of single behaviors for each player depends on the current game situation and on the result of negotiations between the group members. These negotiations are necessary to coordinate the individual behaviors with respect to the overall goal. E.g. to build a wall for defence, the *Navigators* need to negotiate about the position each robot has to take.

offensive	defensive
control ball	take ball away
search ball	search ball
fetch ball	fetch ball
run into space	mark a robot
dribble ball	shoot ball away
score a goal	block ball

offensive	defensive
build wall for attack	build wall for defence
get space	pressing, space covering
play on wings	goal defence
powerplay	play for time

Table 1. Behaviors of a single player (left) and a team of player (right)

Strategies for Soccer Playing Robots The *Strategists* of all team-members negotiate about the future strategy s. Such a global strategy can be composed of several group behaviors. For example, an *attack on the wings* may consist of the group behaviors *build wall for attack* and *play on the wings*. Those group behaviors are possibly concurrent. The strategy specifies a set of roles R_s needed. Each *Strategist* generates its local strategy with respect to the global strategy. The assignment of a role r_i to a player i is subject to further negotiations between the *Strategists*.

Furthermore, the strategic layer has to decide how riskily a robot should play. This disposition to take risks is expressed by the markedness of a behavior. In other words, the strategic layer decides how this behavior is executed. E.g., if the team is leading there is no reason to take any risks and thus the robot will turn slowly in order not to loose the ball. Otherwise, if the team knows that it is down, it will risk a lot.

5 Experiments

During the last two years, the presented architecture was implemented in wide parts. The implemented system is shown in fig. 1. The original idea was to run each agent in an own process. In contrast, some agents were put together in one process in order to save communication and synchronization time.

Due to the fact, that on the used hardware the access to the ultrasonics and the wheels is via the same microcontroller board, the sonar-based obstacle avoidance, the vehicle control and the pilot, which reacts on the sonar data, are put in a single agent. Also, the Strategist and the Scene Detection have only minimal functionality in the current state of implementation and thus are integrated in the Navigator. The Scene Detection is responsible for exchanging information between the robots. Fig. 1 (a) shows a typical scene in a game, and the views of the scene seen from the robots one (Fig. 1 (c)) and two (Fig. 1 (d)) including extracted ball. By combining the camera and the object detection into one elementary agent, it is possible to find objects (namely ball and goals) within 40ms. A Kalman filter based approach leads to robust predictions of the ball position. The exchange of information between elementary agents takes place in the scene detection and is currently implemented for absolute positions of ball

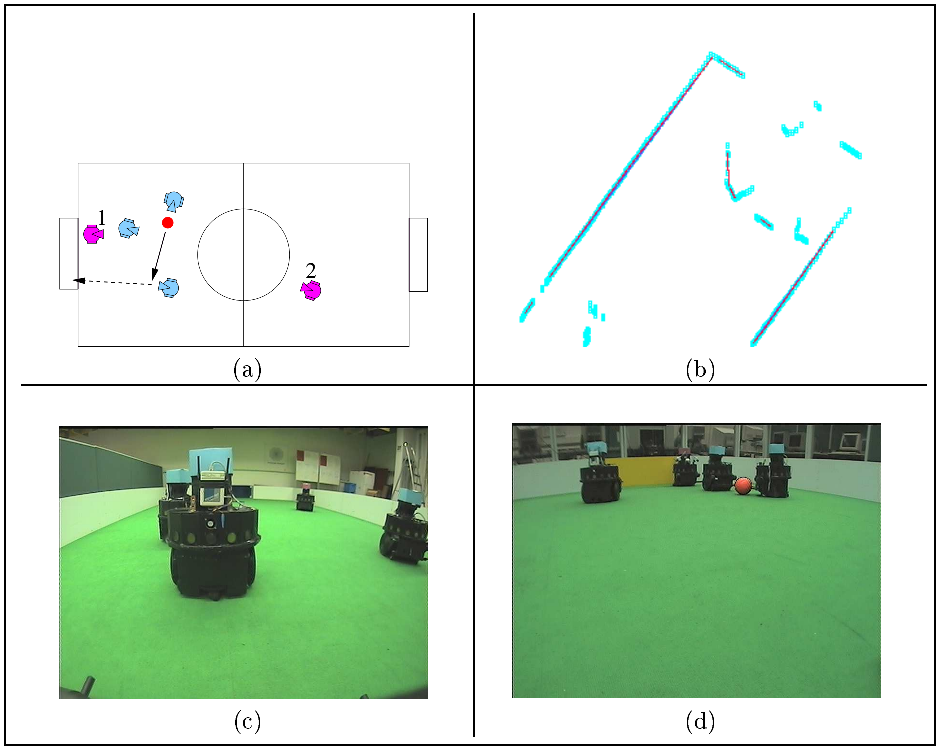

Fig. 4. A scene (a), laser scan of robot 3 (b), view of robot 1 (c) and robot 2 (d)

and robots. The communication between the elementary agents is realized using CORBA, which enables an easy handling mechanism for the message exchange. To keep track of the own position, the combined laser/localizer-agent evaluates every scan. By taking the physically possible motion between two consecutive scans into accout, the accuracy of the position estimation is highly increased. Fig. 1 (b) shows the localization result for robot three.

In the past, we had successfully shown that the presented architecture is suitable to control a team of autonomous robots. The CoPS Stuttgart team competed successfully at the *RoboCup 1999* in Stockholm and became Vice Champion at the *German Open Championship* 1999 in Stuttgart, where all German teams competed.

6 Conclusion and Outlook

To control a group of robots we have chosen a multi-agent approach, that enabled the adaption of the architecture to any kind of application. Each robot contains a strategical, tactical, and reflexive layer where concurrent *elementary agents* are located to fulfil miscellaneous tasks.

Then we demonstrated the application of the architecture to a concrete example, the soccer playing robots. It has shown that the presented architecture is suitable for this scenario. The transfer of this multi-agent architecture to other scenarios is quite easy, because only some agents of the tactical and strategical layer have to be modified of exchanged. We used this for experiments of fault-tolerant assignment problems ([LSS+00]). In the future we want to improve the negotiation mechanisms and team strategies so that the performance of the robot agents is enhanced.

References

[AML87] J.S. Albus, H.G. McCain, and R. Lumia. Nasa/ nbs standard reference model for telerobot control system architecture. *Technical Note 1235*, 1987.

[BD94] S. Bussmann and Y. Demazeau. An agent model combining reactive and cognitive capabilities. *IROS*, pages 2095–2102, 1994.

[BML97] M. Becht, M. Muscholl, and P. Levi. Ein framework fuer die kooperationsverfahren zwischen roboteragenten. *AMS*, pages 166–177, 1997.

[Bro86] R.A. Brooks. A layered control system for a mobile robot. *3rd Symposium. MIT Press*, pages 367–372, 1986.

[CCC93] R. Cattoni, T. Coianiz, and B. Caprile. Reactivity and planning in the mobile robot of maia. *ICAR*, pages 585–590, 1993.

[LBLM98] P. Levi, M. Becht, R. Lafrenz, and M. Muscholl. COMROS - A Multi-Agent Robot Architecture. In *DARS 3*, 1998.

[Lev89] P. Levi. Architectures of individual and distributed autonomous agents. *International Conference of Intelligent Autonomous System, IAS-2*, pages 315–324, 1989.

[Lev92] P. Levi. Architecture and restriction-based planning of autonomous traffic agents. *Journal for the Integrated Study of Artificial Intelligence Cognitive Science and Applied Epistemology (CC AI)*, 9(1):43–64, 1992.

[LSS+00] R. Lafrenz, M. Schulé, M. Schanz, P. Levi, P. Molnár, and J. Strake. Three-index assignment of robots to targest: An experimental verification. *to appear in: International Conference of Intelligent Autonomous System, IAS-6*, 2000.

[Mus00] K. M. Muscholl. *Interaktion und Koordination in Multiagentensystemen*. PhD thesis, University of Stuttgart, 2000.

[OL97] N. Oswald and P. Levi. Cooperative vision in a multi-agent architecture. *Lecture Notes in Computer Science*, 1310:709–716, 1997.

[Rau98] W.A. Rausch. *Interaktion autonomer Roboter*. Shaker Verlag, 1998.

[vB95] H. van Brussel. Navigation issues in intelligent autonomous systems. *IAS*, 4:42–52, 1995.