A Functional Architecture for a Team of Fully Autonomous Cooperative Robots*

Pedro Lima, Rodrigo Ventura, Pedro Aparício, and Luis Custódio

Instituto de Sistemas e Robótica
Instituto Superior Técnico
Av. Rovisco Pais, 1 — 1049-001 Lisboa
PORTUGAL
{pal, yoda, aparicio, lmmc}@isr.ist.utl.pt
http://lci.isr.ist.utl.pt/projects/mrob/socrob/

Abstract. A three-level functional architecture for a team of mobile robots is described in detail, including the definition of the role assigned to each level, the main concepts involved, and the corresponding implementation for each individual robot. The architecture is oriented towards teams of fully autonomous cooperative robots, able to carry out different types of cooperative tasks. Complexity is reduced by the decomposition of team strategies into individual behaviors, which in turn are composed of primitive tasks. Relationships among robots of the team are modeled upon the joint intentions framework. An application to Robotic Soccer and some of its preliminary results are presented.

1 Introduction and Motivation

Different functional architectures have been proposed in distributed artificial intelligence and intelligent control literature to handle the complexity of controlling a fully autonomous mobile robot or a team composed of such robots. A common concept among those approaches is the existence of atomic *primitive tasks* or *behaviors* which are the kernel of the architecture. Tasks executed by the robot result from the composition of those entities.

The main difference between the existing approaches concerns the interaction among the atomic entities. While some authors allow full flexibility, so that a team behavior emerges from a negotiation between running behaviors [3], others prescribe, with different flexibility levels, the task decomposition into primitive tasks [1], to an extent which may even forbid any direct communication between primitive tasks [7].

A three-level functional architecture for a team of mobile robots is introduced in this paper. The architecture is oriented towards teams of fully autonomous cooperative robots, able to carry out different types of cooperative tasks. The

* This work was supported by the Science Service of the Calouste Gulbenkian Foundation and by the Portuguese Foundation for Science and Technology (ISR/IST programmatic funding).

level splitting is inspired by the work of Drogoul and his co-workers [3], but there are important differences regarding the modeling of the relational level, which describes inter-agent negotiation and role assignment. The *joint intentions framework* [8, 5, 2] provides a solid foundation for teamwork modeling, and will be used in this work to support the implementation of the relational level.

Complexity is reduced by the decomposition of team *strategies* (i.e., what should be done) into individual behaviors, which in turn are composed of primitive tasks. A set whose elements are the behaviors assigned to each robot of the team is designated as the *tactics* (i.e., how to do it) for a given strategy. An application to Robotic Soccer and some of its preliminary results developed during and after the RoboCup'98 contest are presented.

The paper is organized as follows. Section 2 describes the team and individual architectures, with details of teamwork modeling at the relational level and of the foreseen/current implementation for the introduced concepts. Section 3 maps the concepts onto a robotic soccer team. Section 4 closes the paper with some preliminary conclusions and reference to future work.

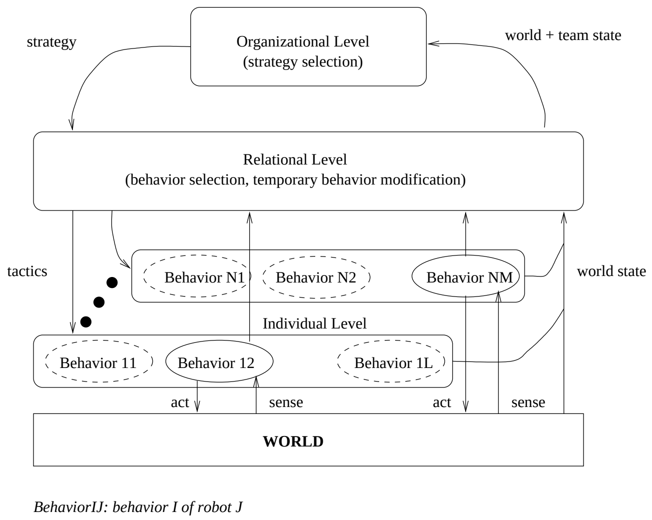


Fig. 1. The functional architecture of the ISocRob team.

2 The Functional Architecture

2.1 Team Architecture

The team architecture, based on the 3-level agent team architecture first proposed by Drogoul and his co-workers [4], is depicted in Fig. 1. Our interpretation of Drogoul's three-levels follows:

- **Organizational level**: establishes the strategy to be followed by the whole team, given the team and world states. The team state corresponds to the current set of behaviors under execution. The following examples, taken from the robotic soccer context, illustrate the concept of world (game) state, which is divided in two classes:
  1. game situations reached upon the application of RoboCup tournament rules (e.g., kickoff, end-of-game, penalty-for, penalty-against);
  2. team evaluation of current game status (e.g., losing & close to the end of the game, ball close to our goal).

  Strategies can be divided in, at least, two major categories:
  - pre-programmed scenarios for game situations in game state category 1 above;
  - dynamic strategies (e.g., defend, attack, counter-attack), corresponding to game state category 2 above.
- **Relational level**: at this level, relationships among robots are established. The robots negotiate and eventually come to an agreement about some team and/or individual goal. Moreover, behaviors are assigned to the individual robots, after a selection from within behavior sets representative of alternative tactics for the strategy selected by the organizational level. The selected behavior set depends on the current world plus team states. Behavior assignments may also be temporarily modified as a result of inter-robot negotiations.
- **Individual level**: encompasses all the available robot behaviors. Those include the primitive tasks (e.g., SeekBall, KickBall, RotateLeft) and their relations.

A *behavior* corresponds to a set of purposive (i.e., with a goal) primitive tasks sequentially and/or concurrently executed. A *primitive task* is a *sense-think-act loop* (STA loop), a generalization of a closed loop control system which may include motor, ball tracking or trajectory following control loops, to name a few.

STA loops are composed of the following key components:

- **goal**: the objective to be accomplished by the primitive task (e.g., moving to a given position plus orientation (pose) set-point, tracking the ball in the image);
- **sense**: sensor data required to accomplish the goal (e.g., distance to an object, object position in an image);
- **think**: the actual algorithm which, using the sensor data, does what is required to accomplish the goal (e.g., motion controller, ball visual servoing);
- **act**: the actions associated to the **think** algorithm (e.g., moving the wheel motors).

The sequence of primitive tasks is traversed as the logical conditions associated with the connections among them become true. The logical conditions are defined over a predicate set. There are two *predicate* classes:

- predicates which check the value of a given variable (e.g., the variable *goal* in lastseen(goal)=left);

– predicates which check the occurrence of a given event (e.g., the see(ball) predicate checks whether the ball became visible).

A *world model* is required to provide information to the relational and organizational levels regarding the world state. Since all computation is supposed to be distributed over the team members, with no external storage available, a distributed world model representation is required, containing all the relevant information for negotiation between agents, and in general the result of processing raw data, for primitive tasks usage. A *distributed blackboard* is proposed to implement the world model [9].

2.2 Individual Robots Architecture

Each individual robot is provided with all the three levels of the team functional architecture. However, the organizational level is only active in one of the robots, assigned as the team *captain*. The remaining robots have a dormant organization level, to ensure fault-tolerance: whenever the captain robot has a malfunction, the next robot in the list takes over as the captain. The list has no special order since, from the hardware standpoint, all robots of the team are currently homogeneous. In an non-homogeneous population, the potential captains (from a computational capacity standpoint) should be sorted according to their descending computational power.

An agent-based programming language has been specified and is currently under development [9, 10], to provide the team strategist (e.g., the coach, in robotic soccer) with the means to program the population in order to achieve the strategic objectives, embedded in the behaviors and in the primitive task STA loops.

Each of the above concepts will be implemented as follows:

– the *strategy* is determined at the organizational level by a *state-machine* whose transitions are traversed upon the matching of specific world states, and whose states define the current strategy. Therefore, strategies change when the world state (as perceived by the team) changes;
– *tactics* selection, including behavior selection, negotiation, and temporary behaviors modification, is implemented by *relational rules* at the relational level;
– a *behavior* consists of a *state-machine*, where each state corresponds to an *STA loop* and each transition has associated logical conditions defined over the predicate set described in subsection 2.1;

Team organization is necessarily a centralized operation. As such, decisions on strategies must be taken by a single agent, designated as the *captain*. Thus, the organizational state machine runs in the captain. To increase team robustness, whenever the current captain does not signal that it is alive for more than a timeout period, a new captain must take control of the team.

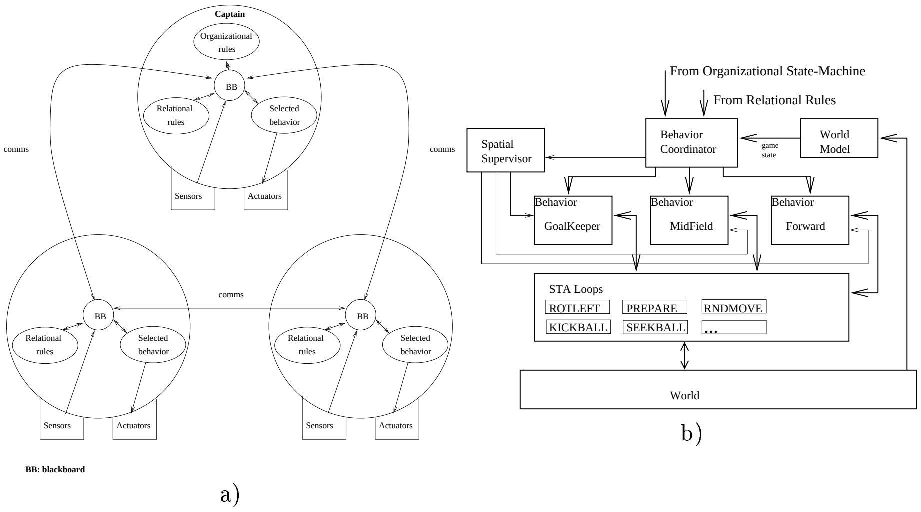


Fig. 2. Implementation of team functional architecture at each individual robot: a) Blackboard, relational rules, organizational state-machine and communications. b) Behavior selection and coordination.

A *blackboard* implements global shared memory and event-based[1] communication. The blackboard is the sole medium of communication between the agents, supporting the message exchange required for negotiation. One of the key factors will be the distribution of data per robot, which should minimize the need to communicate in order to obtain information (e.g., data obtained from processing an image should be stored in the robot where the image was acquired). The global team strategy is also stated in the blackboard, as a variable which triggers some of the relational rules.

The schematic block diagram of the implementation, at each individual robot, of the team architecture is depicted in Fig. 2 - (a). The behavior selection and spatial coordination are detailed in Fig. 2 - (b): the *behavior coordinator* selects the correct behavior for the robot, based on the world state and on the strategy and tactics provided by the organizational and relational levels of the team architecture. When a behavior is selected, the corresponding *spatial supervisor* is also activated. The *spatial supervisor* ensures that the robot always stays within the influence zone associated to its active behavior (see Section 3).

2.3 Relational Rules

Relationships among the team robots are established at the relational level of the team functional architecture. Given a strategy established by the organizational level, different tactics can be used to implement it. Tactics consist of behavior sets, whose elements are the behaviors assigned to each individual robot of the

[1] Event is interpreted here in the context of a computational model.

team. A tactics is chosen based on the current world state, but also on each agent's current internal state. The strategy must specify not only the goal to be attained (e.g., attack, defense) but also criteria to check how close to the goal the team is. Behaviors can be temporarily modified as a result of inter-robot negotiation, as part of the tactics to attain the goal.

An example is the situation where two teammates, both assigned a forward-like behavior, actively try to get the ball. In such a case, one of them should signal the other its intention. A negotiation process would follow, where the teammates would determine their distances to the ball, to decide which one should pursue it. After taking such a decision, the other player should temporarily modify its normal forward-like behavior.

The absence of such a relational mechanism leads to situations where team behavior is poor. Consider the case of two forward-like players with similar behaviors, that often conflict with each other while trying to reach the ball. The key to solve this problem is to endow the team members not only with individual goals, but also make them knowledgeable of the team goals. This is clearly related to concepts such as *joint persistent goal*, *joint intentions* and *joint commitment* [8, 5, 2]. Moreover, it requires *communication* between team members.

For instance, a joint persistent goal is defined in [5] as follows: *A team of agents has a joint persistent goal, relative to q, to achieve p iff: they all mutually believe that p is currently false; they all mutually believe that they all want p to be eventually true, and until they all come to mutually believe either that p is true, that p will never be true, or that q is false, they will continue to mutually believe that they each have p as a weak achievement goal relative to q.*

The example above can be interpreted under this definition. The strategy p (e.g., attack) is a weak achievement goal relative to the main goal q of scoring a goal. Suppose the two players both assumed the Forward behavior as part of the selected tactics. They will pursue the strong goal q (i.e., they will attempt to score a goal) by executing their Forward behaviors so as to attain p. A criterion to check whether p is attained is to determine whether the players are able to keep playing within their assigned influence zone. Both players will continue to work towards meeting this and the other criteria which define the attack strategy until they all come to mutually believe either that all the criteria were met, that the criteria will never be attained (e.g., after a timeout), or that 'scoring a goal' is no longer the main team endeavor (e.g., because the game is over). Working towards meeting the criteria includes temporarily modifying their behaviors to cope with the team goal (e.g., refraining from pursuing the ball). This distinguishes a group of non-cooperative agents whose individual goals just happen to be the same, from a group of cooperative agents which share a common aim. The latter exhibits cooperation and coordination, while in the former the individual agents compete when the resources are scarce [5].

The relational rules implement a *recipe* which is commonly agreed by all the agents of a team [5]. This recipe is embedded in the rules and may either be prescribed initially (i.e., before joint action is started) or evolve over time. We are currently looking at the possibility of changing the recipe over time using reinforcement learning techniques, based on a performance function which weights the reliability (i.e., the ability to meet specifications) and the cost (computational or other) of a given recipe [6]. Reinforcement learning should be able to determine the recipe (to achieve a joint intention) that best balances cost and reliability.

3 Application to a Soccer Robot Team

Some of the concepts described in the previous section will now be mapped onto a team of fully autonomous soccer robots.

Fig. 3 presents a functional division of the field in several regions. These are zones where robots try to locate themselves inside the field, according to their assigned behaviors, e.g., defenders should stay inside the D zone and Forward players should stay inside the F zone. This division helps the assignment of influence areas to players.

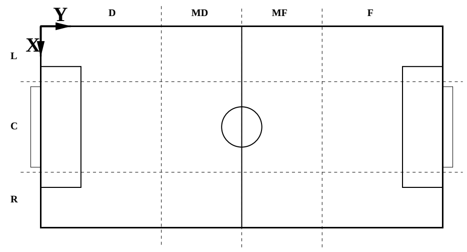


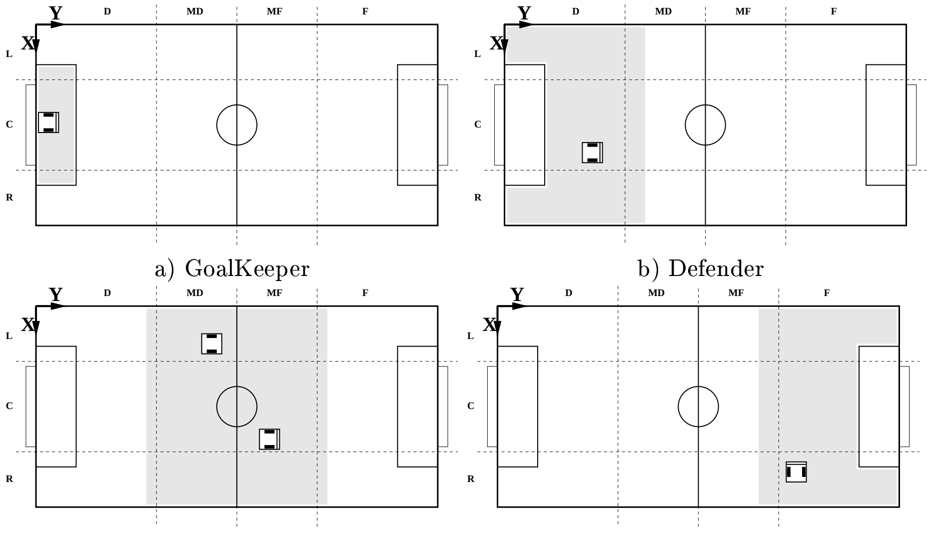
Fig. 3. The field division in actuation areas.

Besides Defense (D), MidDefense (MD), MidForward (MF) and Forward (F), further divisions are introduced to increase the field resolution. Along the field longitudinal axes, the field is divided in Left (L), Center (C) and Right (R) parts. This division is particularly useful when the team has more than one player acting in the same functional area (e.g., L and R defenders).

3.1 Player Behaviors

As explained before, behaviors are composed of primitive tasks sequentially or concurrently executed. A field influence zone is associated to each behavior. Several behaviors must be implemented in a robot soccer team. The most significant ones, whose influence zones are depicted in Fig. 4, are:

- GoalKeeper – Defends the goal. To do that, it continuously looks for the ball and, if necessary, leaves the goal area and kicks it away. The influence zone is defined by the goal area lines and is shown in Fig. 4 - a).



a) GoalKeeper b) Defender

c) MidFielder d) Forward

Fig. 4. Influence zones for individual behaviors.

- Defender – The defender mission is to move the ball from the vicinity of its team goal to the opponents field. If possible, it should try to move the ball to the vicinity of a MidFielder. It should return to its original position (D) when the ball is once again in the opponent's field (see Fig. 4 - b)).
- MidFielder – Such as in real soccer, the Midfielder is able to play in a variety of positions. Its influence zone lies within the MD and MF areas (see Fig. 4 - c)). This player natural ability is to receive the ball from its own team field and decide what to do, based on the other players availability. If a Forward is in the near vicinity of the opponents goal (F area), the MidFielder should try to pass it the ball.
- Forward – The Forward behavior induces the player to be in the F zone (see Fig. 4 - d)). If the ball goes into our field, the Forward's mission is to keep track of the ball, although it should not move out of its zone by its own initiative. When the ball moves into the F zone, it must try to take control over it and kick it into the opponents goal. Such a behavior is implemented by the state machine of Fig. 5 - a).
  An alternative implementation would consist of letting the Forward players move up and down the field, using the lateral L and R corridors.

3.2 Relational Behavior Modification

Individual behaviors can be temporarily modified to allow cooperative relations between teammates, as explained in Subsection 2.3. Fig. 5 - b) depicts the state machine which implements the Forward behavior endowed with states

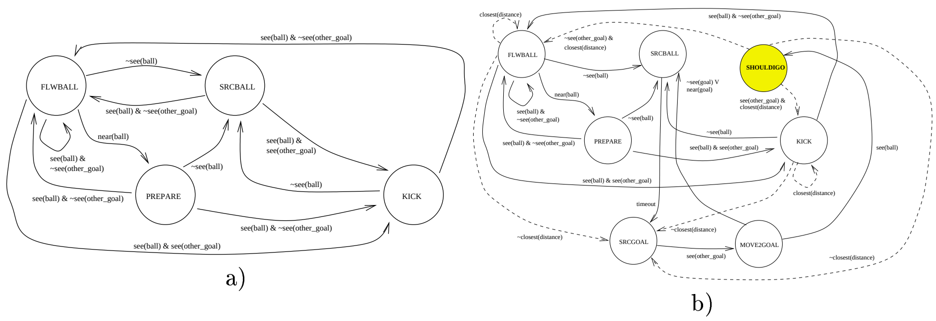


Fig. 5. State machines for the Forward behavior: a) With no cooperation. b) With cooperation. States and state transitions representing relational rules are filled and dashed in the diagram, respectively.

and state transitions representing relational rules. The negotiation implemented corresponds to the example in Subsection 2.3 concerning two Forward players who actively try to get the ball. The additional state ShouldIGo is entered by a Forward player upon its detection of the ball (predicate see(ball)). In this state, a message is broadcasted through the blackboard to all teamates stating that this Forward player saw the ball and also the estimate of its distance to the ball. The occurrence of messages from the the other teamates stating that they saw the ball and including the estimate of their distance to the ball is also checked in the ShouldIGo state. The distance to the ball of all the Forward players who saw the ball recently is continuously sorted. The sorted distances are checked by the closest(distance) predicate, associated to all the dashed arcs of the state machine in Fig. 5 - b), which are responsible by the temporary behavior modifications with respect to the non-relational Forward behavior of Fig. 5 - a). The most important modification consists of not immediately kicking (KickBall state) or following (FlwBall state) the ball upon its detection in state SrcBall, but rather moving to state ShouldIGo where distances to the ball are compared. Should the Forward be the closest the ball among all its Forward teammates (or the only one who sees the ball), the state machine execution proceeds as in the non-relational Forward behavior. Otherwise, the behavior is modified by making the player move to a location close to the other team's goal (states SrcGoal and Move2Goal).

3.3 Game State

The *game state* refers to either situations reached as a result of the application of RoboCup rules or to an evaluation of the current game status. State changes are induced by the time flow and teams actions during the game. Examples of game states are as follows:

Game situations

– game-start – This happens in the beginning of the game, after a goal or when the game restarts after a break;
– penalty-for, penalty-against;
– end-of-game – This is signaled by an external event (e.g., two whistle blows).

Evaluation of game status

– ball-our-ofield – One of our players has ball possession. The ball is in our field;
– ball-nour-otfield – None of our players has ball possession. The ball is in the other team field;
– losing & close to the end of the game;
– ball close to our goal.

3.4 Scenarios for Game Situations

Pre-defined scenarios are usually associated with the game states corresponding to game situations (see above). An example is the game-start situation, where the players must move to their pre-determined start positions (see Fig. 6). Self-location of the players must be accomplished at this state as they must be correctly positioned prior to the start of the game. After positioning, the players will wait for the external kickoff signal (e.g., a whistle blow) that signals the start of the game.

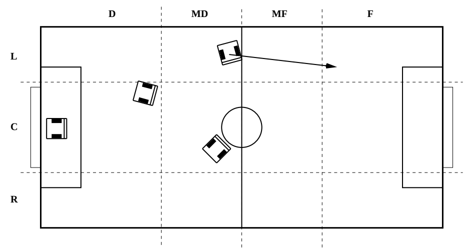
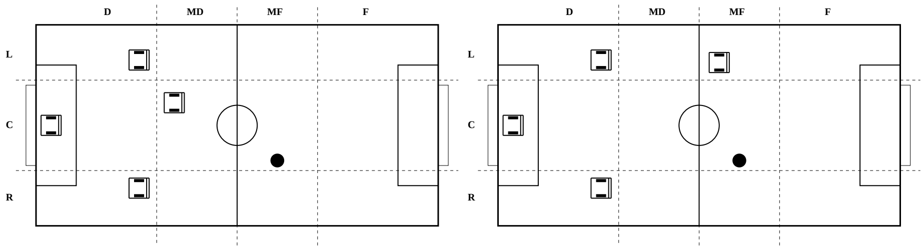


Fig. 6. Players position at game start-up.

3.5 Dynamic Strategies and Tactics

During the game the ball moves inside and outside of the team mid-field. Depending on factors such as the ball position and motion, the current game state, the current score, the number of available players and their behaviors, the opponents positions, the elapsed time, and the current strategy, the team strategy may change. This is inspired by real soccer. Possible strategies are:

– defense – The ball must be prevented from entering our field. Should that happen, it must be moved into the opponents field. Several defense tactics exist. Two examples of tactics for the Defense strategy are:
  - *Strong Defense* (SD) – This strategy points towards creating a continuous, physical barrier between the ball and our goal. It is aimed at avoiding opponent players from moving towards our goal. When re-positioning, the Defender players should try to avoid occluding the GoalKeeper visibility of the field, i.e., the DC zone should be free of players (see Fig. 7 - a)).



a) Strong defense initial positions. b) Medium Defense initial positions.

Fig. 7. Strong and Medium Defense tactics for the Defense strategy.

  - *Medium Defense* (MD) – Points towards a strong defense and a good recovery mechanism, essential to counter-attack. The concept is illustrated in Fig. 7 - b). The difference between the SD and the MD is that in MD not all players are moved into our field. This makes the transition to Counter-Attack easier, as one of the players stays in the opponent field.

– counter-attack – A counter-attack happens if the team is positioned to move the ball quickly into the opponent field and score a goal. It requires a Defender, to handle a possible interception of the ball by an opponent, a Mid-Fielder, to pass the ball into the Forward area, and a Forward player to kick the ball into the opponent goal.

– attack – Under this strategy, the whole team moves forward. Besides the GK, there is only one player in our field. This movement requires two MidFielders (one in the MD zone and another in MF zone), and one Forward. The idea is to have the ball passed from the M zone into the F zone, where a Forward player is to pick it up and kick at the goal.

4 Preliminary Conclusions and Future Work

Currently, our mid-size real robots are capable of simple but essential behaviors, composed of primitive tasks, such as following a ball, kicking a ball, scoring goals and defending the goal, using vision-based sensors (see Fig. 5). Current available behaviors include shooting at an empty goal starting from increasingly more difficult situations or defending the goal by permanently tracking the ball and kicking it out of the goal area as soon as it gets too close. One Forward *vs* one Defender and a GoalKeeper have also been successfully tested in live action.

Our current and future work is centered on four main topics:

- development of the self-localization system based on a vision camera and a mirror;
- update and tuning of the primitive tasks software;
- design and implementation of an agent-based programming language suitable for multi-agent systems;
- study and development of a teamwork model and its integration with the team functional architecture.

Among those, self-localization is perhaps the most essential. The functional architecture described in this paper relies on a (at least rough) awareness by each robot of its location in the field and, consequently, of the team current disposition in the field.

The work has been carried out in a bottom-up fashion, since we believe that many conceptual issues can be raised from and are strongly constrained by the actual implementation problems. Nevertheless, the basic framework described in the paper, concerning hardware, software and functional architectures, was designed in a top-down fashion in the beginning of the project and has been essentially kept unchanged so far.

References

1. J. S. Albus. Outline for a Theory of Intelligence. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(3), May/June 1991.
2. P. Cohen and H. Levesque. Teamwork. Technical Report 504, Center for the Study of Language and Information, SRI International, March 1991.
3. A. Drogoul and A. Collinot. *Autonomous Agents and Multi-Agent Systems*, chapter Applying an Agent-Oriented Methodology to the Design of Artificial Organizations: A Case Study in Robotic Soccer. Kluwer Academic Publ., 1998.
4. Alex Drogoul and C. Dubreuil. A distributed approach to n-puzzle solving. In *Proceedings of the Distributed Artificial Intelligence Workshop*, 1993.
5. N. R. Jennings. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence*, 75:195–240, 1995.
6. P. U. Lima and G. N. Saridis. *Design of Intelligent Control Systems Based on Hierarchical Stochastic Automata*. World Scientific Publ., 1996.
7. G. N. Saridis. Architectures of intelligent controls. In M. M. Gupta and N. Sinha, editors, *Intelligent Control Systems*. IEEE Press, Piscataway NJ, 1995.
8. M. Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7, 1997.
9. R. Ventura, P. Aparício, and P. Lima. Agent-based programming language for multi-agent teams. Technical Report RT-701-99, RT-401-99, Instituto de Sistemas e Robótica, IST-Torre Norte, March 1999.
10. R. Ventura, P. Aparicio, P. Lima, and L. Custódio. IsocRob — Intelligent Society of Robots. In *Collection of RoboCup99 Team Description Papers*, August 1999.