ELSEVIER

Robotics and Autonomous Systems 36 (2001) 67–86

Robotics and Autonomous Systems

www.elsevier.com/locate/robot

# Coordination in multi-agent RoboCup teams

Ciprian Candea [a], Huosheng Hu [b], Luca Iocchi [c], Daniele Nardi [c,*], Maurizio Piaggio [d]

[a] *Artificial Intelligence Research Group, 2A Reconstructiei, 2400 Sibiu, Romania*
[b] *Department of Computer Science, University of Essex, Wivenhoe Park, Colchester CO43SQ, UK*
[c] *Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Via Salaria 113, 00198 Rome, Italy*
[d] *Department of Communication, Computer and System Sciences, Via Opera Pia 13, 16145 Genova, Italy*

**Abstract**

In this paper, we focus on various aspects of coordination in the framework of the RoboCup competitions. In particular, we address both multi-agent systems that have been developed within the simulation league and multi-robot systems that have been realized in the middle-size league. From the multi-agent perspective we present a behavior-based technique for position selection and the so-called holonic approach. In the multi-robot domain, we address both communication and distributed coordination within a heterogeneous team of autonomous robots. © 2001 Elsevier Science B.V. All rights reserved.

*Keywords:* Coordination; Cooperation; RoboCup; Multi-agent; Multi-robot

## 1. Introduction

In this paper, we present four contributions on the issues arising in coordinating the behaviors of a team of autonomous agents playing soccer both in a virtual (simulated) environment and in a real (robotic) setting. The problem of coordination is one of the scientific challenges that have been put forward for the RoboCup competitions [1,2].

In particular, we address the challenge of improving the team performance by exploiting the ability to coordinate the actions of the players. The advantages of coordination in a soccer playing scenario can not only be measured in terms of the number of goals scored in the opponents' goal, but also in the achievement of subgoals such as defending own goal, gaining ball possession, properly covering a position in the field, and so on. These issues are dealt within the simulation and middle-size league of RoboCup.

The *simulation* league is played by computer programs. Each team is formed by 11 players, each of them implemented by a separated program. The simulation is run by a Soccer Server [3]. The server is responsible for generating "visual" and "auditory" information for the players in a form of text strings. It also takes the action commands and parameters from the player, such as power and direction for kicks. Information passed between server and client is subject to noise, a player may not kick the ball exactly as intended, and the view of the pitch will also contain errors. Each player has limited resources both in terms of sensing and in terms of motion capabilities. Communication between the server and the players provides the basis for the development of cooperative strategies.

In the simulation league all the constraints deriving from actual robot implementation are hidden from the designer, who can concentrate on agent modeling;

* Corresponding author.
*E-mail addresses:* ciprianc@airg.verena.ro (C. Candea), hhu@essex.ac.uk (H. Hu), iocchi@dis.uniroma1.it (L. Iocchi), nardi@dis.uniroma1.it (D. Nardi), piaggio@dist.unige.it (M. Piaggio).


PII: S0921-8890(01)00137-3

therefore, the relevant research issues are team work and multi-agent systems. In particular, the simulated environment is well suited for the application of learning techniques for the collaborative strategies.

The *middle-size* league is played by teams composed by one goal keeper and three middle-field players. All sensing devices must be on-board the robots, in particular, global vision as well as other external sensing devices are not available. The above constraints generally lead to the development of robots where computation is performed on-board. Communication among players is allowed, and it can be exploited to achieve coordination, but due to the frequent communication failures, the robots should not depend on communication, nor on information provided by other robots.

Obviously, when real robotic players are considered, issues such as uncertainty on the physical sensors and actuators arise, that are not dealt within the simulated environment. Moreover, coordinating a multi-agent system, where there is a fully distributed processing of the sensory input and thus there are many possible sources of errors, appears to be a rather challenging problem.

Summarizing, the two leagues, while being rather diverse in terms of the relationship to the environment, share a common assumption of full autonomy of the players leading to the development of distributed approaches to coordination. Indeed, in a highly dynamic and uncertain environment such as the one provided by RoboCup games, the centralized coordination of activities underlying much of the work in robotics does not seem to be adequate. In particular, the possible communication failures as well as the difficulty of constructing a global reliable view of the environment require full autonomy on each robot.

In the rest of the section, we provide a general introduction to the problems that are addressed in the contributions presented in this paper. We mainly refer to multi-agent systems (MAS) with regard to the simulated, software teams of players and to multi-robot systems (MRS) with regard to the real robot teams. In particular, we address the following issues:

- Coordinated behaviors in MAS.
- MAS architectures.
- Communication in MRS.
- Coordination in MRS.

The contributions presented in this paper fall in the above mentioned categories and will be presented in the subsequent sections.

### *1.1. Coordinated behaviors in MAS*

The concept of MAS is emerging as an important model for designing intelligent and complex software applications, exploiting diverse and distributed on-line information sources, and building sophisticated and distributed agent systems that work effectively in a group setting.

In general, to perform the specific tasks by cooperation of a team of agents three distinguishing features are needed. Firstly, the environment where the agents act is unpredictable and the tasks that need to be done are complex. Secondly, MAS should achieve effective coordination in terms of task allocation, resources sharing, team formation, etc. Thirdly, the behaviors in MAS are hierarchical in nature, including low-level emergent behaviors and high-level cooperative/collective behaviors.

Behavior coordination plays an important role in those cooperative MAS, which has been investigated by many researchers from different points of view, encompassing the entire spectrum of AI approaches. For instance, Decker and Sycara [4] developed a WARREN model to build intelligent agents with adaptive behavior coordination for doing portfolio management on the Internet. Parker [5] developed a fully distributed and behavior-based software architecture, ALLIANCE, that incorporates the use of mathematically modeled motivations such as impatience and acquiescence within a team of heterogeneous mobile agents to achieve adaptive action selection. Jennings et al. [6] developed an agent-based system for managing a British Telecom (BT) business process which is dynamic and unpredictable. Such a system has behavior laws to determine how the system's behavior depends upon its composition and on its component's behavior. Lesser [7] proposed an agent architecture for a large-scale multi-agent environment that was built on the generic partial global planning (GPGP) framework for coordination of small teams of agents.

There are many MAS approaches being pursued in the RoboCup simulation leagues, including teamwork among multiple agents [8], cooperative behaviors through implicit communication [9], a layered

approach to learning client behaviors [10], environment modeling [11], emergent cooperative behaviors [9], and the behavior-based position selection [12] which will be detailed in Section 2.

### 1.2. MAS architectures

One basic feature of MAS is the system architecture. The simplest architecture is a pure reactive one. The agents based on such an architecture perceive environment changes (as stimuli), interpret them, and react to them according to predefined procedures (starting from simple "if *stimulus* then *reaction*"); hence, they take decisions considering only information coming from current inputs [13].

Another widely used architecture is the belief–desire–intention architecture (BDI), which can be viewed as purely deliberative and typically involves two main processes: deciding which goals to pursue and then how these goals are going to be achieved [14].

However, an essential requirement for a real-world agent acting in a dynamic and uncertain environment is to be able to show reactive as well as deliberative behavior. The need to embody both kinds of behaviors has led to the so-called hybrid architectures.

In addition, task decomposition leads to subsystems that are arranged into hierarchical layers. Specifically, one can identify two types of control flow within the layered architectures:

- *Horizontally layered*: the layers are each directly connected to the sensory input and to the action output. In this way each layer acts like an agent, providing indications about what action to perform.
- *Vertically layered*: sensory input and action output are dealt with by dedicated layers, while one or more intermediate layers perform the processing from the inputs to the outputs.

For example, TOURING MACHINES [15] are horizontally layered architectures with three layers: one layer decides which actions the agent should perform, one enables for an immediate response to changes that occur in the environment and the planning layer provides the agent with pro-active behaviors [16].

The INTERRAP [17] is a vertically layered, two-pass agent architecture. INTERRAP contains three control layers, each of them associated with a knowledge base, a representation of the world at different level of abstraction. The highest-level knowledge base represents the plans and actions of other agents in the environment, the middle-level knowledge represents the plans and actions of the agent itself, and the lowest level represents raw information about the environment.

In RoboCup, one of the main challenges that have been pursued is finding a suitable balancing between reactivity and deliberation in agent architectures. In early stages of the competition almost all teams used the reactive model for their agents. After the teams gained some experience, they began to develop agents with hybrid architectures. Depending on the intended goal several architectures have been developed, often adapting architectures like INTERRAP or TOURING MACHINES, to the RoboCup environment. In Section 3, we present the holonic agent architecture that provides for a new coordination paradigm.

### 1.3. Communication in MRS

Every robot in an MRS, in order to successfully coordinate its action to achieve a common complex goal, must be aware, at least at a minimal level, of the subtask carried out by the other robots in the system. This problem is known as *the problem of action recognition* — the ability of a robot to observe and interpret the behavior of another robot [5]. The research in cooperative robotics has widely investigated on this issue leading to MRS that can be distinguished in two major categories:

- *MRS based on implicit communication*: in which communication is achieved throughout sensory feedback from the environment in which all robots are operating [9].
- *MRS based on explicit communication*: in which the explicit exchange of information between robots, typically based on messages, is exploited [5,18].

The former has the advantage of not requiring any type of common communication device and can lead to emergent, non-intentional, coordination. The latter has been more extensively exploited and has proven to be more efficient in different domains. For example, in [19], Balch and Arkin investigate the importance of communication in performing tasks related to grazing and foraging.

This trend is also confirmed in RoboCup, in the middle-size league, in which all the research groups that have specifically addressed the problem of team coordination, equipped the robots with suitable communicating devices and relied upon explicit message passing. However, most of the research activity was focused on the coordination and not so much on communication itself. In fact the devices used were off-the-shelf radio modems and wireless Ethernet cards and communication was based on the underlying common IP protocol, either TCP–IP [19] or UDP–IP [20,21], provided by the operating system or by commonly available add-on libraries. In Section 4 communication is addressed in detail: the properties and limitations of the above mentioned protocols in the RoboCup scenario are discussed together with the issues related to the computational load and impact on the real-time system.

### *1.4. Coordination in MRS*

The design of MRS has been pursued from quite different perspectives, by developing them from a system engineering standpoint to a biologically inspired approaches. From a system engineering viewpoint the idea is to use multiple robots for accomplishing a given task faster and more effectively [22]; multiple robots can be in different places at the same time, they can perform concurrent and cooperative actions and, in general, they allow for a decomposition of a complex task into simpler sub-tasks. In addition, the use of multiple robots can improve the robustness of the system; following [5], the MRS can be designed and implemented in such a way to guarantee two important features: adaptivity and fault tolerance.

In addition, several tasks have been used for MRS: foraging, multi-target observation, box pushing, exploration and soccer. While these domains are representative of a large number of issues which arise from the study of cooperative MRS, a classification of the proposed approaches needs to take into account additional features. In particular, an analysis of the work on MRS shows that different approaches to coordination are chosen according to the specific environment (see [23,24] for a survey). For instance, the studies in robotic scouting teams working in unstructured environments led to behavior-based approaches to achieve desired formations [22], while distributed planning is used when the robotic team has to compete for limited resources [25] or in structured and not very dynamic environments [26]. It has also been argued that robot teams working in highly dynamic environments get low benefit from complex plan negotiation. This is why some approaches propose to eliminate any kind of negotiation in coordination [27].

In the real robot leagues of RoboCup the dynamics of the environment and the presence of the opponent team provide for a challenging testbed of MRS approaches. As previously mentioned, coordination in the middle-size league relies on explicit communication, however, due to the frequent communication failures, the robots must not depend completely on communication, nor on information provided by other robots. Successful centralized approaches are possible when it is easy to reconstruct global information, for instance by means of laser scanners [28]. However, in most cases a distributed approach to coordination must be pursued because of the limitations on communication and of high uncertainty of the information acquired through the sensors. Furthermore, in the middle-size league there have been examples of heterogeneous robot players. In such a setting coordination needs to be achieved without laying down drastic prerequisites on the knowledge of the single players [29]. A distributed approach for coordination of heterogeneous players has been developed within the ART team and will be described in Section 5.

## 2. Position selection behaviors for coordination

This section describes the position selection behaviors (PSB), a set of the coordination behaviors in the simulation league, which were developed by the Essex Wizards team. The focus here is on how each PSB is selected for individual agents and on the architecture used to implement such position selection.

Selecting optimal positions for soccer robots to move in a robot football game is a challenging task since the environment and robot motion are dynamic and unpredictable. This section provides an overview of the PSB for the Essex Wizards simulated soccer team, a third prize in the RoboCup'99 simulator league at Stockholm and a third place in the 1st European RoboCup 2000 at Amsterdam. The focus is on how each PSB is selected for individual agents and

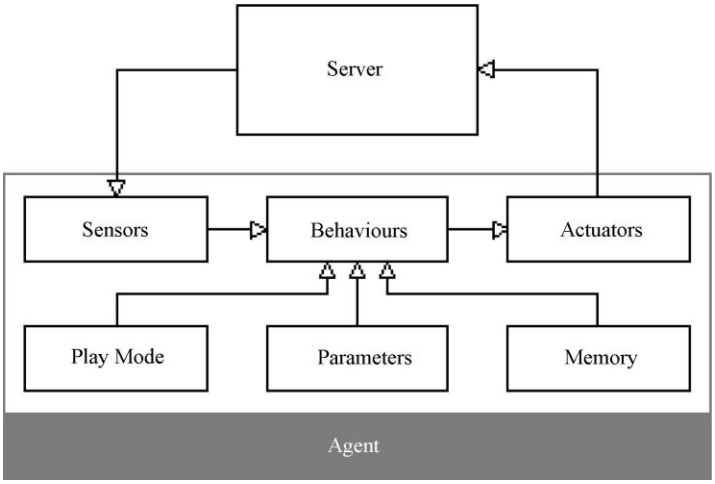


Fig. 1. Essex Wizards agent MAS.

what architecture is used to implement such position selection.

### 2.1. Essex Wizards MAS

The main objective for the Essex Wizards team is to build a firm research platform on which future work on multi-agent systems can be done. Currently, a multi-threaded approach is designed by Kostiadis [30] for the Essex Wizards team to achieve real-time performance, in which a modular architecture is adopted for each soccer agent. In such a design, there are six function modules: *Sensors*, *Actuators*, *Play mode*, *Parameters*, *Memory*, and *Behaviors*, as shown in Fig. 1.

The Behaviors module needs to carry out a number of tasks, including the "decision-making processes". It must provide both the basic behaviors such as moving, dribbling and kicking, and the cooperative behaviors such as selecting where to move to, which team-mate to pass to, and selecting between the behaviors.

#### 2.1.1. Low-level basic behaviors

At the lowest level any decisions made by the agent must be reduced to the core primitive actions provided by the server, i.e. "Kick", "Turn" and "Dash". These have been extended to extract useful information from the memory to centralize common tasks. For example, the "Turn" action turns the player by a certain degree, and is often useful to turn to face a particular direction, e.g., to face where the ball is expected to be. Other low-level behaviors include: *Advanced Kick*, that can deal with more complex situations by moving the ball in stages to a position where the desired kick can be made; *Move*, that mixes turns and dashes to reach the desired location; *Dribble* which involves intermixing kicks, dashes, and possibly turns to move the ball across the pitch, while keeping the ball under control; *Look*, which keeps track of the current ball location or scan for it by executing a number of turns if the ball is not in the view.

#### 2.1.2. High-level cooperative behaviors

The high-level cooperative behaviors are built on top of low-level primitive behaviors. These are currently implemented as a rule-based system available to each agent during the game. *Intercept* involves predicting the location of the ball at the time of intercepting and moving to that location, using moves and kicks behaviors. *Clear Ball* involves kicking the ball, using the advanced kick behavior, away from the goal, towards the side of the pitch and the opponent's end. *Send Ball* occurs when the agent is close to the opposing goal, and attempts to get the ball to a position from which a team-mate can score. *Pass Ball* analyzes, using a decision tree, the locations of team-mates and opponents on the field in order to decide for a good pass, and returns information about the most suitable team-mate to pass to, along with an estimate of the

success. *Position Selection* examines the current view of the pitch and uses it to suggest a good place to move to. Selection of such a position is a non-trivial task, requiring information about the current role of the agent and the state of the pitch. More details can be seen in the next section.

### 2.2. Position selection behaviors

The PSB is for the soccer agent to select the best position on the pitch to move to [12]. The factors being considered for such selection in a game include the position of team-mates and opponents, the location of the ball, and the abilities of the players. It is important to be in a good position to receive the ball from a team-mate or intercept the ball from opponent. Players are assigned a role, which in turn defines a home position for them. The PSBs currently implemented by Hunter et al. [12] are:

- *Fixed and fixed relative PSB*: The fixed PSB is the selected target point. The fixed relative PSB is similar, and returns a point at a fixed offset from the ball location.
- *Tracker PSB*: It protects the goal by selecting an appropriate position on the line between the home goal and an opponent player to be tracked.
- *Marker PSB*: It aims at positioning the player so that it can intercept a pass to the opponent, and intercept a shot on goal.
- *Ball tracker PSB*: It chooses a target location on the line between the ball and the goal. It can optionally fail if the ball is not in the player's home area, or if the ball-to-goal distance is less than a minimum threshold.
- *Offside trap PSB*: Offside trap is a tactic strategy to prevent an opponent player to pass the off-side line ahead of the ball. The PSB aims at selecting a point on the off-side line that shadows an opponent. For safety it will fail if the ball is too close to the off-side line.
- *General purpose PSB*: It is based on the strategic position by attraction and repulsion (SPAR) method [8], which works by maximizing the distance between the target point and other players on the pitch (both opponents and team-mates), and minimizing the distance between the target point and the opponent's goal, and between the target point and the ball. Four additional constraints are also applied: (i) Stay in an area near the player's home location. (ii) Stay within the pitch boundaries. (iii) Avoid being in an off-side position. (iv) Stay in a position in which it is possible to receive a pass.

### 2.3. Implementation

In order to present a simple interface to the core player, a library of PSBs has been developed. Positioned objects are created as instances of PSBs for certain circumstances. The core player then activates a specific PSB based on the current situation to find the position to move to. This brings consistency to the core player, in order to find the desired target location it simply activates the PSB for the current tactical situation recursively.

#### 2.3.1. Multiple players

Since cooperation among players is very important in a game, it has been taken into account in the design of the positioning strategy. Firstly, the basic indirect communication is used by certain PSBs. The observed positions of team-mates are used when selecting a suitable position. Secondly, although each player uses identical code, the configuration of each one, described in the next section, is different. This leads to different behaviors of individual players, ensuring that not all players decide to have the same role.

Similar reactive approaches have been used elsewhere, particularly in behavior-based robotics [13]. Subsequent work [31,32] used multiple robots capable of forming and maintaining formations, such as flocking. Such formations would be simple to implement using a suitable PSB.

#### 2.3.2. External configuration

Rather than passing a large number of static arguments between the core player and the PSB, most of the configuration parameters is extracted from an external configuration file at startup. The player extracts the information for the role to be played in each formation.

This makes a clear distinction between tactical and implementation details, and hides many implementation details completely. For example, each PSB has a different set of options. Accessing each PSB directly requires that the player know these options.

*2.3.3. Failsafe operation*

In certain situations a single PSB may not be able to select a good position due to the state of the pitch. For example, an off-side trap is no use if the ball is already beyond it. In this situation, the PSB could return some kind of error signals, but this would need the high-level behavior to either chose a default position itself, or redirect the request to an alternative PSB.

Alternatively, other high-level behaviors could guarantee never to call a PSB unless it will succeed. A PSB guarantees it will always return a valid position to the high-level behavior. By default if a PSB fails to generate a valid position internally, the home location of the player is returned. Constructing a stack of PSBs can change this situation. If the first one fails the next one is called, and so on down the stack until one succeeds.

*2.4. Competition results*

Each PSB described in the previous subsections has been tested extensively to ensure that it works as designed, and also implemented in a real game situation [12]. As an example, Fig. 2 was produced from the log file of the game between Essex Wizards and Rolling Brains at the RoboCup'99 competition in Sweden. Essex Wizards players are in dark gray circles and the lines represent the movement of players over time. The small white circle is the ball, and the black dashed line is the path taken by the ball.

The points A and D represent the positions of the Essex Wizards sweeper and the ball, respectively, at around cycle 3150. The ball is moving in to the other half of the pitch, and the sweeper moves forward to enforce the off-side trap. Before the sweeper arrives, the ball returns close to the centerline (E) and the sweeper (B) immediately begins ball tracking instead, attempting to reach a point between the ball and the goal. The sweeper currently moves more slowly than the opponent with the ball, as it needs to recheck the location of the ball periodically. Even at point C, the sweeper cannot see the ball at point F without stopping and turning.

## 3. Holonic architecture for coordination and control

In this section, we address agent architectures used in RoboCup simulation league. A critical feature of such architectures is the choice of the decision model: a hierarchical one, based on rigid rules, powerful but not very flexible, or a heterarchical one, very flexible but insufficiently robust.

In the AIRG team we adopt the holonic paradigm, used in holonic manufacturing system (HMS), based on the concept of *holon* defined as "autonomous and co-operative building block of a manufacturing system for transforming, transporting, storing and/or

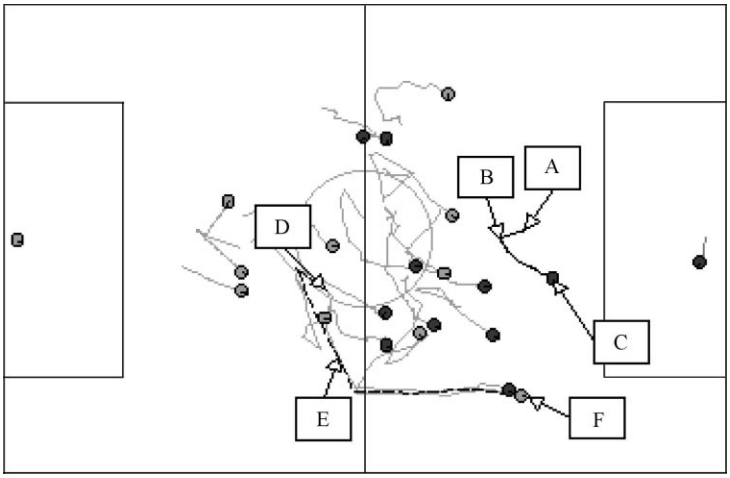


Fig. 2. Ball tracking in action.

validating information and physical objects" [33]. In the HMS framework, soccer can be viewed as a multilevel real-time system, requiring a multi-agent approach and exhibiting an enterprise-like structure suitable for applying a manufacturing paradigm.

### 3.1. Holon

The concept of holon has been proved to be very successful in the industrial process planning, increasing the flexibility of decisional systems. It was proposed by the Hungarian philosopher A. Koestler (1978) and explains the importance of the hierarchy of a system. Accordingly, a holarchy is a hierarchy of self-regulating control building blocks (holons), which operate: (a) as autonomous wholes in supra-ordination to their parts, (b) as dependent parts in subordination to controls on higher levels, (c) in coordination with their local environment [33]. The strength of a holonic organization, or holarchy, is that it enables the construction of very complex systems that are nonetheless efficient in the use of resources, highly resilient to disturbances (both internal and external), and adaptable to changes in the environment they act in.

### 3.2. Combining agents with holons

As above hinted the holonic paradigm needs to be suitably combined with a MAS approach to realize a team for RoboCup simulation league. There are several reasons justifying this view. A MAS approach is critical for soccer teams because, besides the individual goal of each agent, global objectives are established committing all or some agent groups to their completion [1]; MAS are the natural means to design and implement holonic software systems as well as a lot of related kinds of applications [34,35], while the holonic paradigm allows for better modeling of multilevel systems (including the player–team–coach ensemble) [17] and increases the flexibility of decisional systems (as both HMS and robotic teams) [36].

One of the most important characteristics of holarchies is the capability to modify themselves, i.e. to create temporary hierarchies; like modern industry, soccer is very dynamic: not only each team comes with its own style and game strategy, but also each game phase has some specificity [37]. Holarchies offer a balance between the two above mentioned decision models: the hierarchical control (fixed, static, pre-established) and the heterarchical one (autonomous, decentralized, flexible).

PMA/KUleuven developed a reference architecture for holonic manufacturing control systems [33]. The architecture is called PROSA: product–resource–order–staff architecture, which refers to the composition types of holons. The resource holon contains a physical part, namely a production resource of the manufacturing system, and an information processing part that controls the resource. The product holon holds the process and production knowledge to assure the correct product manufacturing, with adequate quality. The order holon represents a task in the manufacturing system. It is responsible for performing the assigned work correctly and on time. The staff holon is implemented to assist the other three holons in performing their work. Based on PROSA, an adapted approach can be applied to soccer teams (Table 1). The holarchy is structured on five levels (RoboCup, coach, team, player, component), each of them containing the specific holons for every category (product, resource, order, staff).

The generic holon architecture, shown in Fig. 3 (adapted from [17]), is conceptualized and implemented as a three-layer agent architecture (i.e. *holarchy layer*, deliberative layer and reactive layer). The holarchy layer contains mechanisms for devising joint plans with other holons/agents. To achieve the needed flexibility it has three sub-layers: (i) integration (for vertical collaboration with adjacent layers, i.e. for a holon/agent situated at the team layer to collaborate with the other holons/agents situated at the coach and/or player levels); (ii) cooperation (for horizontal integration of holons/agents situated at the same level); and (iii) monitoring (for modifying holarchy). The *deliberative layer* contains mechanisms able to deal with local plans and local goals. Finally, the *reactive layer* contains facts representing the world model of the holon/agent.

The programming mechanisms used to implement those features are intensive multi-threading and dynamic priorities. Threads represent the atomic, sequential, asynchronous, and dynamic entity of concurrent programming. In spite of their usefulness in the earlier development stages (conceptualization, design, and so

Table 1
Holonic approach for soccer teams

| | RoboCup | Coach | Team | Player | Component |
|---|---|---|---|---|---|
| Product | Workshop<br>Championship | Selection<br>Training<br>Strategies<br>Tactics | Game | Implement strategy<br>Implement tactics<br>Acquire skill<br>Acquire tactics | |
| Resource | | Strategy<br>Tactics<br>Memory<br>Rules | Time<br>Schemata | Skill<br>Schemata<br>Stamina<br>Components | Head<br>Right leg<br>Left leg |
| Order | Research<br>Entertainment | Building teams<br>Testing teams<br>Winning championships | Win game | Winning<br>Preservation | Execute |
| Staff | | Player<br>Teams | Spare player | | |

on), all concepts of functional entities (holon, agent, etc.) can be naturally implemented as threads. The negotiation between conceptual entities (agents, holons or even human decision-makers like the coach) has to be reflected in the multi-threading structure.

Dynamic priorities are vital in order to build and modify holarchies. For instance, the rank of the player holons in the holarchy is setup with two priority types: the static one is assigned to each player at game start based on his role in the team and/or game; the dynamic one is acquired online during the game, assigned by the team leader or by the coach. Indeed, priorities are not only an implementation mechanism (dedicated primarily to improve efficiency) but also play a conceptual role. In the context of RoboCup, priorities can be used not only to express holarchies, but also as heuristics. In addition, priorities are applied with a finer granularity and are often changed dynamically so that they become a convenient tool to fine-tune the low level implementation.

In the first European RoboCup we have adapted our 1999 AIRG team to work based on a holonic architecture. To evaluate this type of architecture, we tried to keep all 1999 implementation like skills, position-

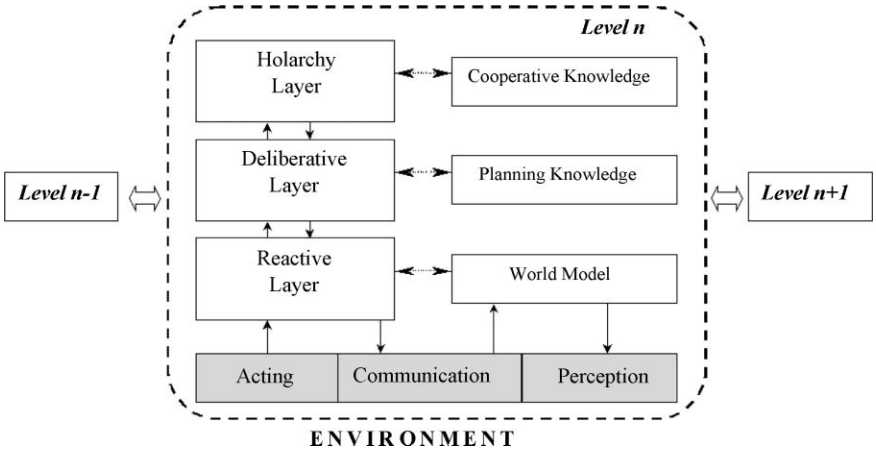


Fig. 3. Generic holon architecture.

ing algorithm and so on. The most important change was at the decisional level, how the agents form the decision. The time necessary to achieve the same task is improved with 8–10% in average, depending on the task. The decision accuracy also increases with 2–5% in average. At present, the weakness of our implementation is the algorithm for inter-holonic negotiation at all levels (inside the players and inside the team). In fact, we used a simple algorithm based on fixed priorities that are pre-defined; such an approach requires further development.

## 4. Communication and control in ETHNOS

In this section, we address the issue of communication in MRS in the RoboCup competitions where the software architecture of a robot soccer player must allow not only for a successful intra-robot integration of the different activities (spanning over many different types of representation) but also it must also guarantee a successful inter-robot integration by supporting communication and cooperation. In heterogeneous teams (the ART team — the Italian robot soccer team, e.g. [20]) these problems are particularly significant because successful coordination and cooperation between robots that differ both in software and in hardware must be accomplished. Here these communication problems are addressed by presenting ETHNOS IV — a programming environment for the design of multi-robot real-time systems. In particular, this paper will focus on the communication framework provided that allows for transparent and efficient communication between robots and supervising stations across different media (cable or wireless).

### *4.1. Communication*

Communication in our view is not an accessory component or module in the robotic system but needs to be fully integrated within the robot architecture. In this perspective we present ETHNOS IV (expert tribe in a hybrid network operating system): a programming environment for multi-autonomous robot systems. It is composed of:

- ETHNOS IV, a dedicated distributed real-time operating system, from which the overall environment takes its name, supporting different representation, communication, and execution requirements. These functionalities are built on top of a Posix compliant Linux RT kernel.
- A dedicated network communication protocol designed for both the single robot and the multi-robot environment, specifically taking noisy wireless communication into account.
- An object-oriented application programming interface (API) based on the C++ language (and a subset based on Java).
- A set of additional development tools (a robot simulator, a Java applet template, etc.).

The reference framework of an ETHNOS-based robot architecture is entirely based on the concept of expert, a concurrent agent responsible for a specific deliberative or reactive behavior. The experts in ETHNOS can be organized in groups, possibly distributed in a computer network, depending on their computational characteristics: the type of cognitive activity carried out, timing constraints, type of data managed, duration, etc. However, these groups do not have a hierarchical organization but, on the contrary, are executed in a flat model in such a way that the real-time constraints are not violated. This generality in the expert specification makes possible the use of ETHNOS with different architectural organizations without loosing real-time analysis, execution and inter-robot communication support. In this paper, we will focus only on the communication aspects. For further information please refer to [38].

#### *4.1.1. Inter-expert and inter-robot communication*

One of the goals of the RoboCup competition is to encourage the comparison and exchange of methodologies, techniques and algorithms within the robotics and artificial intelligence community. A common programming environment which, without imposing significant constraints on the single components, allows to easily put together the result of different researchers within the same group (or possibly also across different groups) is certainly a contribution in this direction.

According to the above goal, the ETHNOS programming environment allows the robots to be programmed in such a way that the different experts can be integrated, during development but also at run time, with little or no intervention in the software of the

expert itself, thus facilitating both rapid prototyping and dynamic architectural reconfiguration. The first property facilitates the development of a robot application (from a set of components or behaviors) even by non-highly specialized programmers (for industrial purposes but also for didactic activity in student projects, particularly relevant in RoboCup); the second property allows the system to easily scale-up to a robot capable of different complex activities and thus to be able to switch at run-time from a configuration in which it is performing a certain task (e.g., in which the robot is attacking and thus the active behaviors are responsible for ball pushing, path planning, and obstacle avoidance) to a different configuration (e.g., in which the robot is defending and the active behaviors are responsible of opponent-tracking, ball interception, etc.).

The above described properties are achieved by exploiting a suitable inter-expert message-based communication protocol (the EIEP — expert information exchange protocol [39]) fully integrated with the expert scheduler. The EIEP encourages the system developer to de-couple the different experts in execution, to reach, as close as possible, the limit situation in which the single expert is not aware of what and how many other experts are running. In this way an expert can be added, removed or modified at run-time without altering the other components of the system. Expert de-coupling is achieved by eliminating any static communication link. The EIEP is essentially an efficient implementation of a network distributed blackboard [40] based on a publish/subscribe protocol. Similar approaches have being exploited in more recent research in multi-agent systems [41] or have lead to the development of commercial products (NDDS by RTI).

The EIE protocol is primarily based on an asynchronous message-based method in which the single expert subscribes to the particular message types, known a priori. When another expert publishes any message of the subscribed types, the subscriber receives it transparently. In this way, an expert does not know, explicitly, who the sender of a message is or to which other experts, if any, its message is being distributed. Moreover, the same principles apply also if the application is distributed in a computer network: messages are distributed and received regardless of the particular machine on which they were produced or on which an expert subscribed [42]. Thus, from this point of view, the same application is programmed in the same way, whether it will run with all the experts executing on the same computer (robot soccer player) or with the experts executing on different computers connected in a network (soccer player and supervising station or remote high-level component responsible of non-time critical computationally intensive tasks).

ETHNOS also provides transparent inter-robot communication support. In fact, using the same EIEP principle, messages can be exchanged on the basis of their contents not only within the same robot but also among different robots (players within the same team). However, in inter-robot communication, it is often important to know exactly who the sender of a message is and to distinguish between internal messages (meaning messages to be distributed within the machines implementing a single player) and external messages (meaning messages to be sent from a player to another) to avoid the explosion of unnecessary network data communication.

In ETHNOS, the different experts are allowed to subscribe to communication clubs. For example, we may envisage a single club to which the different players belong or even different clubs, to which only players which are performing a cooperative activity belong (e.g., the one which carries the ball toward the goal and the one which is helping it in the action). Message subscription and publication can thus be distinguished in internal only, internal and in a specific club or internal and external in all clubs. Again, it is the responsibility of the system to dynamically transparently distribute the messages to the appropriate receivers.

In Fig. 4, we show an example configuration that we have tested. In particular, we are allowing the robots to communicate in a single club — the team club — (to which all of them have subscribed) and with an external supervisor (the coach) which monitors the activity of all the players.

It is worth noticing that, because of EIEP protocol, whenever we want to add/remove a player to/from the team, it is not necessary to explicitly inform each player about the modifications in the teams composition. In fact, the players just have to agree about the type of information they are ready to send and receive: it is ETHNOS which automatically updates the database of each club's members and consequently

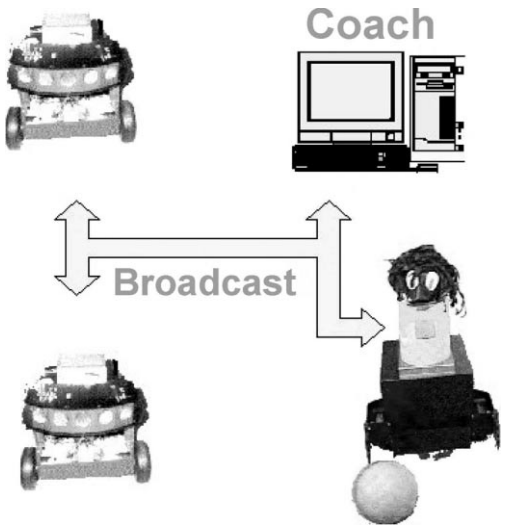


Fig. 4. Example of multi-robot ETHNOS configuration.

dynamically dispatches messages from senders to the appropriate receivers. This has been very important in ART in which there were more than four types of robots available, with different characteristics of play, and thus different team compositions were selected for the single matches and also modified during the matches to better contrast the opponent.

Moreover, since in the RoboCup (and in general in mobile robotics) network communication is often wireless (i.e. radio link, Wavelan©, etc.), due to noise or interference transmission packets are sometimes lost. In this context, both TCP–IP and UDP–IP-based communication cannot be used: the former because it is intrinsically not efficient in a noisy environment; the latter because it does not guarantee the arrival of a message, nor any information on whether it has arrived or not. For this reason, we have also designed a protocol for this type of applications, called EEUDP (ETHNOS Extended UDP) because, based on the UDP, it extends it with the necessary properties. The EEUDP allows for the transmission of messages with different priorities. The minimum priority corresponds to the basic UDP (there is no guarantee on the message arrival) and should be used for data of little importance or data that is frequently updated (e.g., the robot position in the environment that is periodically published). The maximum priority is similar to TCP because the message is sent until its reception is acknowledged. However, it differs because it does not guarantee that the order of arrival of the different messages is identical to the order in which they have been sent (irrelevant in ETHNOS applications because every message is independent of the others), which is the cause of the major TCP overhead. Different in-between priorities allow for the re-transmission of a message until its reception is acknowledged for different time periods (i.e. 5, 10 ms, etc.).

#### *4.1.2. Experimental results*

ETHNOS has been extensively tested and evaluated in RoboCup in the Italian team ART. In this section, we will describe the advantages of its use in this domain from the communication point of view addressed in this paper. From this perspective the support provided by ETHNOS is at two different levels: network communication and message distribution.

For network communication the system allocates a maximum guaranteed and dedicated time. This value is computed automatically on the basis of the schedulability analysis so that the real-time execution of the user experts is also guaranteed. Clearly, the value depends on the computational load of the experts in execution as well as the processor speed. In Fig. 5, the four graphs represent different machines corresponding to three robots (Relé, Homer and Bart) and the coach, connected using Wavelans. The difference in the processing power among the four different machines leads a difference in the schedulability analysis and thus a different amount of time that be safely dedicated to communication without missing the hard real-time deadlines. The top line in the graph indicates the calculated time available each 100 ms for communication purposes. This value decreases as the number of experts in execution increase (and so the computational load). The bottom line indicates the time spent in communication which also increases with the number of experts (this is because for this experiment we have assumed that the activity of every expert involves either transmission or reception of messages). In this way it is always possible to determine a priori whether the system is capable of both communicating information and executing in real time. For example, if we consider Relé, the limit situations in which the two lines converge is also the limit beyond which the schedulability analysis fails. Instead, if we consider Bart, real-time scheduling is

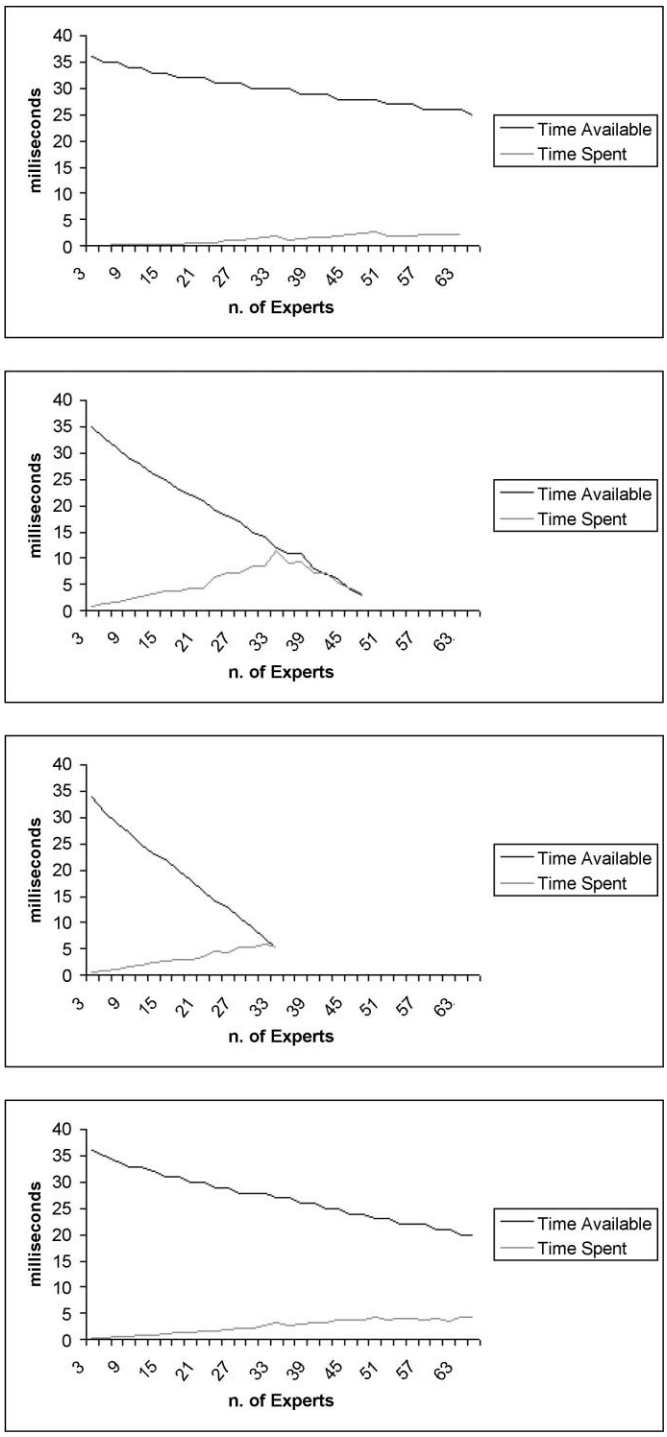


Fig. 5. Network communication in ETHNOS. From the top the graphs represent the robots: Relé, Homer and Bart, and the coach.

possible also beyond the intersecting point but only if we accept communication degradation.

## 5. Distributed coordination in heterogeneous robotic teams

In this section, we present the approach to coordination in heterogeneous MRS developed by the ART team [20] of middle-size league robotic soccer players. ART is composed by different robots developed in various Italian universities. Because of this kind of organization the players differ both in the hardware and in the software and thus distributed coordination is required (see also [43]). Moreover, it is useful to have a simple and flexible coordination protocol, that allows the coach to accommodate the various configurations that can arise by forming teams with different basic features.

The hypotheses underlying the coordination problem in our approach are:

- *Communication-based coordination*: exploiting the use of communication among the players to improve team performances, allowing the robots to acquire more information and to self-organize in a more reliable way.
- *Autonomy in coordination*: the players are capable to perform their task, possibly in a degraded way, even in case of lack of communication.
- *Heterogeneity in the multi-agent system*: the players are heterogeneous both from hardware and software viewpoints.

Besides the above constraints, coordination in ART has been designed to deal both with roles (defender, attacker, etc.) and with strategy (defensive, offensive). While the strategic level is currently demanded to an external selection (the human coach), roles are dynamically assigned to the various team elements during the game, depending on the configuration present on the field.

From a technical perspective the proposed approach is based on the explicit exchange of data about the status of the environment and is based on simple forms of negotiations. The distributed coordination method presented here has been successfully employed by all the members of the ART team during the 1999 and 2000 RoboCup competitions. The method is based on the ETHNOS communication layer that has been described in detail in the previous section. The effectiveness of the method has been proved by the fact that we were always ready to substitute any robot with another one, without affecting the performance of the overall team.

The goal of coordinating through a distributed protocol a multi-agent system, formed by heterogeneous components, not only has been achieved, but actually provided a substantial contribution to the overall performance of the ART team. While the effectiveness of coordination has been addressed in the RoboCup environment, the techniques and the tools can be successfully exploited in other multi-robot domains, where similar assumptions are verified.

A key step that made coordination successful has been the experimental work carried out in order to attain the desired coordinated behavior and the use of suitable tools for the analysis of data exchanged during the game.

### 5.1. Coordination

The major issues that we have addressed in the coordination protocol are the dynamic assignment of roles and the team strategy.

We adopted a *formation/role* system similar to the one described in [8,44]. A formation decomposes the task space defining a set of roles. Each robot has the knowledge necessary to play any role, therefore robots can switch roles on the fly, if needed. However, the implementation choices are different due to the difference in the application domain: in the simulation league the focus is on communication failures, while in the middle-size league the use of this kind of coordination is justified by the following experimental observations: (i) if more than one robot of the same team tries to perform the same action (e.g. to go to the ball) they very likely obstruct each other, worsening the team's performance; (ii) in order to win it is necessary to occupy properly various parts of the field. For instance not all the robots should attack: some must stay back to defend, otherwise they will concede many goals.

Therefore, the basic formation of our team requires that one robot takes the role of going to the ball, another that of defending and another that of supporting the attack. However, other formations are possible

depending on the kind of strategy adopted (offensive, defensive) and on the need to handle special situations such as the malfunctioning of the goal keeper.

The computation for the coordination protocol is distributed. Inside each robot, the coordination module processes the coordination messages and selects the formation that the team will adopt and its own role within the formation.

*5.1.1. Step 1: formation selection*

The robots have at their disposal a number of predefined formations and possess the rules to select the formation to adopt, on the basis of the environment configuration. Since each robot's data do not necessarily coincide with those of the others, the robots may choose different formations. Formation selection is thus accomplished by means of a voting scheme: every robot "votes" for the best team formation (according to its internal state); after a broadcast communication of these votes, the formation that gets the majority of votes will be chosen by all the robots in the team.

*5.1.2. Step 2: role assignment*

This step implements dynamic role assignment through explicit communication of the "utility" for the team that each robot will assume a certain role. This is achieved by the definition of a number of *utility functions* (specific for every role) that every robot evaluates given its current local information about the environment. By comparing these values, each member of the team is able to establish the same set of assignments (robot $\leftrightarrow$ role) to be immediately adopted.

More specifically, suppose we have $n$ robots $\{R_1, \ldots, R_n\}$ and $m$ roles $\{r_1, \ldots, r_m\}$. The roles are ordered with respect to importance, i.e. in the current formation assigning $r_i$ is more important than assigning $r_{i+1}$. This means that if $n < m$ then only the first $n$ roles will be assigned, while if $n > m$ then $m - n$ robots will not be assigned any task. In our scenario we always guarantee that $n \leqslant m$, so that every robot will always be assigned a role.

Let $f_j(i)$ be the value of the utility function, computed by robot $R_i$ for the role $r_j$ and $A(i) = j$ denote that the robot $R_i$ is assigned the role $r_j$.

The method for dynamic role assignment requires that each robot $R_p$ computes the following:

1. For each role $r_j$ compute and broadcast $f_j(p)$.
2. For each robot $R_i$ $(i \neq p)$ and for each role $r_j$, collect $f_j(i)$.
3. $L = \emptyset$ (empty the set of assigned robots).
4. For each role $r_j$ do
   4.1. $R_h$ is the robot such that $R_h \notin L$ and $f_j(h)$ has the highest value,
   4.2. if $h = p$ then $A(p) = j$ (my role is $r_j$),
   4.3. $L = L \cup \{R_h\}$.

It is easy to see that every role is assigned to only one robot and that every robot is assigned to only one role. The reason is that on every cycle of the algorithm a different assignment $A(i) = j$ is done: $j$ changes at each cycle and robots already included in the set $L$ of assigned robots cannot be chosen for further assignments.

In order to obtain an effective application of the above method, an important issue to be dealt with is the stability of decisions with respect to possible oscillations of the numerical parameters on which they depend upon (see also [45]).

We have chosen a method of stabilizing decisions by means of *hysteresis*, which amounts to smoothing the changes in the parameter values. This technique prevents a numerical parameter's oscillation from causing oscillations in high-level decisions. In the case of coordination, for instance, if at a certain instant robot $R_i$ covers role $r_j$, its utility function $f_j(i)$ for role $r_j$ returns a higher value.

Another critical factor for the correct operation of coordination is the capability of each element to realize a sudden difficulty in performing its task. For instance, a robot that is moving towards the ball can get stuck on its way. Once it has realized this circumstance, all its utility functions must return low values so that the role can be assigned to other robots.

Finally, if all the robots possess the same data (i.e. communications are working correctly), they will compute the same assignment, but in case of a great loss of transmitted data due to interferences, the robots may have slightly inconsistent data. Therefore, there could be roles temporarily assigned to more than one robot or not assigned at all. However, holes in data transmission last in general fractions of a second. So, if we assume that the values of the "utility functions" do not change sharply, the correct use of the hysteresis system described guarantees that the roles will be

correctly assigned almost always (as shown by the experimental data we have collected during the games that are discussed in the next section).

### 5.2. Experimentation

A successful coordination of the team depends on the correct implementation of the coordination protocol and on a suitable calibration of a certain number of numerical parameters, such as the hysteresis and the "utility functions" parameters. Calibration of these parameters requires ad hoc experimentation methods and tools that will be discussed in this section.

In addition, the heterogeneity of the ART robots makes the experimental phase particularly demanding, since the exchanged information are computed and interpreted differently by each element of the team. For example, consider the evaluation of reachability of the ball: each robot may have a mobile base of different capabilities and a set of behaviors with peculiar speed characteristics and, that notwithstanding, robots must calculate comparable numerical values.

The performance of the ART team at RoboCup 1999 [46] and at European RoboCup 2000 have provided substantial evidence that basic coordination among the team players has been successfully achieved. In fact, in several situations the players smoothly switched role and managed to get ball possession without obstructing each other and they have generally occupied the field in a satisfactory way. The evaluation of the coordination protocol has been carried out by analyzing a set of log files acquired during the matches in the European Championship 2000.

An analysis of the log files generated during the games is useful for highlighting possible situations in which coordination has not given good results. With this respect we made use of a 3D viewer for experimental data that allows for displaying several information about one or more robots. In Fig. 6, the position of three robots along the longitudinal direction of

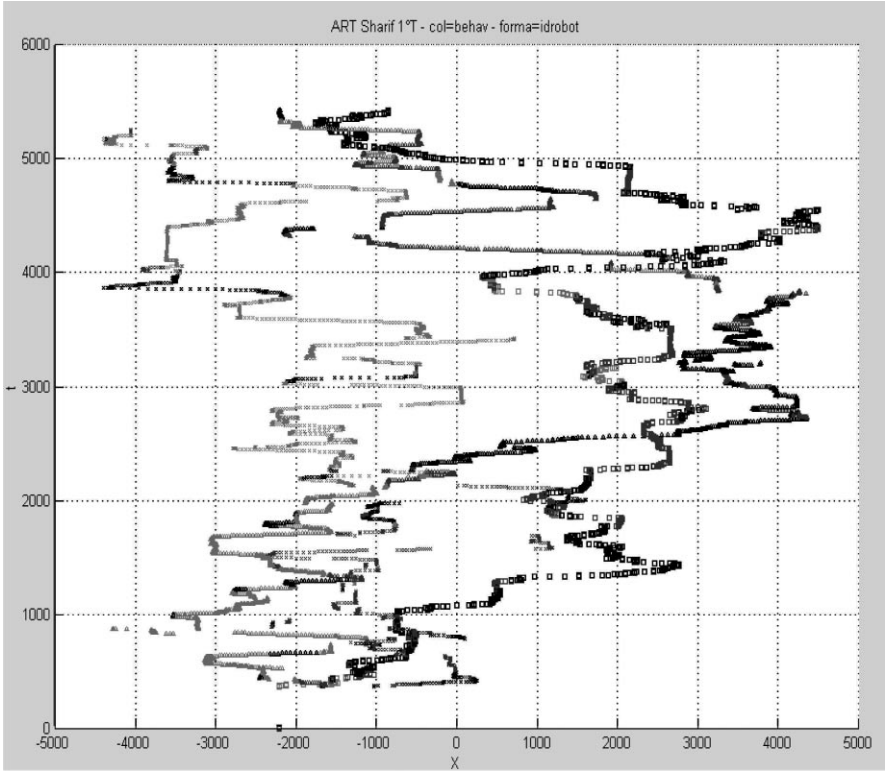


Fig. 6. Graphical display of log files.

the field (*X*-axis) over time (about 5 min of an official match during the European RoboCup Championship 2000) is displayed. Different colors are used for denoting the roles assigned to the robots. The graph visually shows that almost at every time each role is assigned exactly to one robot and that all the field (forward and backward) is properly covered.

Indeed aggregate results of this analysis shows that:

1. the three roles *GoToBall*, *Support*, and *Defend* have been assigned, respectively, for 87.6%, 68.3%, and 93.0% of the time;
2. the forward, middle and backward area of the field have been occupied, respectively, for 57.0%, 68.9%, and 97.5% of the time.

These data show two important features of our distributed approach:

1. with respect to a static assignment of roles, dynamic assignment provides a good distribution of the roles, but with the advantage of selecting the more appropriate robot for every role depending on the current situation of the environment;
2. the field has been properly occupied, even if there is not an explicit assignment of competence areas to the robots.

Finally, by a comparison between the logs and a visual review of the game, we have discovered that one relatively frequent source of failure in coordination arises from situations in which a robot is not actually able to correctly evaluate the utility function for a certain role. Suppose, e.g., that a problem in the vision system causes the robot to incorrectly see the ball very close to it. This robot will be more likely assigned to have the role corresponding to go to the ball, but it usually will take more time than what previously specified to accomplish its task. In these cases the overall performance of the team is not good, since a robot that is actually in a better position to go will not assume the correct role.

## 6. Conclusions

In this paper, we have addressed some issues arising in multi-agent and multi-robot coordination from a RoboCup perspective. Specifically, in the context of the simulation league, we have addressed the problem of determining the position of players and an approach to multi-agent control and coordination based on the notion of holon. In the context of the real robot mid-size league we have addressed the problem of communication and coordination in a heterogeneous system of robots. Although the above works have been specifically developed in the RoboCup framework they bring some general contributions to the fields of multi-agent and multi-robot systems.

In the realm of the simulation league, we have argued that coordinated behaviors play an important role in a cooperative MAS, and form the foundation for cooperative actions. In Section 2, the PSB framework was described for a team of simulated soccer agents to select the best positions in a football game autonomously. PSB aims to achieve optimal team formation via effective behavior coordination subject to dynamic and unpredictable conditions. The PSB framework adopts a modular approach and provides a flexible interface between high-level strategy planners and low-level reactive behaviors. The key concept in the PSB framework is the simplicity in software agent development and the direct connection from behavior coordination to team formation. This is not only very useful to the RoboCup scenario, but also many other real-world applications of MAS where an optimal solution could be achieved if there is an effective coordination among individual agents.

The holonic approach, initially developed for applications within rather controlled environments, has been adapted to the RoboCup framework. Promising results have been reported from industrial application domains (i.e. transportation and manufacturing) where decisional flow is somehow pre-established and holons typically deal with quantitative-based decisions. The application of the holonic approach to the RoboCup environment has required non-trivial adaptations in order to cope with the dynamics of the RoboCup games. Such an adaptation has shown that the holonic approach can be applied also in other domains with more dynamic scenarios.

For RoboCup-like environments, a holonic architecture provides for more operational adequacy regarding order, planning, monitoring, and stability along entire system life-cycle. In addition, the decision-making process is improved, since holons can follow competently all kind of rules in the various system components (i.e. RoboCup game rules, coach directives, and so on).

With regard to the middle-size league of real robots, it is worth emphasizing that the features of the RoboCup scenario make it a very challenging testbed for the comparison of existing methods, since the dynamics of the environment and the presence of the opponent team provide several advantages as compared with previous experimental work on MRS.

ETHNOS is a programming environment for real-time control of distributed multiple robotic systems, that has been described in Section 4. Its main feature, as compared with other architectures proposed in the literature, is the generality of the services it provides, allowing for its use within a variety of cognitive approaches and properly addressing the real-time aspects of communication.

The effectiveness of ETHNOS in the development of a heterogeneous team of robots has in fact been shown by the performance of the ART team in Stockholm and Amsterdam competitions. ETHNOS, implemented on a TRC Labmate and on GenovaRobot Staffetta, has also been successfully tested in human crowded real world environment, working continuously without performance degradation both in our laboratory and as an entertainer during social events, meeting so far approximately more than 5000 people (examples are Exhibition Sculture Gutenberghiane e Manifesto dell'Antilibro, Museo della Scienza e della Tecnica, Milano, 1997, Fair Salone Formula, Magazzini del Cotone Congress Center, Genova, 1998, Gaslini Hospital, Genova, 2000, Fair TecnoHotel, Genova, 2000).

Coordinating a team of heterogeneous robots by a distributed protocol is an important issue for MRS, but there are a few examples in the literature of applications in this scenario. The method presented in Section 5 has been successfully employed by all the members of the ART team during the official RoboCup competitions. The effectiveness of the method has been proved by the fact that the human coach of the ART team was always ready to substitute any robot with another one, without affecting the performance of the overall team. A key step that made coordination successful has been the experimental work carried out in order to attain the desired coordinated behavior and the use of suitable tools for the analysis of data exchanged during the game.

In conclusion, we believe that, in both simulated and robotics scenarios, a major issue in coordination is to find a suitable balance between the robot individual capabilities and the form of cooperation realized. The techniques and the tools presented in this paper give a few interesting solutions that can be exploited in applications where similar underlying assumptions are verified.

## Acknowledgements

The work reported in this paper has been developed in collaboration with: Boldur Barbat, Claudio Castelpietra, Matthew Hunter, Kostas Kostiadis, Alessandro Scalzo, Antonio Sgorbissa, Renato Zaccaria, Constantin Zamfirescu.

We thank the reviewers for their comments and suggestions to improve this paper.

## References

[1] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa, H. Matsubara, RoboCup: A challenge problem for AI, AI Magazine 18 (1) (1997) 73–85.

[2] M. Asada, The RoboCup physical agent challenge: Goals and protocols for Phase-I, in: H. Kitano (Ed.), RoboCup'97: Robot Soccer World Cup I, Springer, Berlin, 1998, pp. 42–61.

[3] I. Noda, H. Matsubara, K. Hiraki, I. Frank, Soccer server: A tool for research on multi-agent systems, Applied Artificial Intelligence 12 (2/3) (1998) 233–250.

[4] K. Decker, K. Sycara, Intelligent adaptive information agents, Journal Intelligent Information Systems 9 (1997) 239–261.

[5] L.E. Parker, ALLIANCE: An architecture for fault tolerant multirobot cooperation, IEEE Transactions on Robotics and Automation 14 (2) (1998) 220–240.

[6] N. Jennings, P. Faratin, M. Johnson, T. Norman, P. O'Brien, M. Wiegand, Agent-based business process management, Journal Cooperative Information Systems 5 (2–3) (1996) 105–130.

[7] V. Lesser, Reflections on the nature of multi-agent coordination and its implications for an agent architecture, Journal Autonomous Agents and Multi-Agent Systems 1 (1998) 89–111.

[8] P. Stone, M. Veloso, Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork, Artificial Intelligence 110 (2) (1999) 241–273.

[9] E. Pagello, A. D'Angelo, F. Montesello, F. Garelli, C. Ferrari, Cooperative behaviours in multi-robot systems through implicit communication, Robotics and Autonomous Systems 29 (1) (1999) 65–77.

[10] P. Stone, M. Veloso, A layered approach to learning client behaviours in the RoboCup soccer server, Journal of Applied Artificial Intelligence 12 (2–3) (1998) 165–188.

[11] K. Yokota, A. Ozaki, K. Matsumoto, H. Kawabata, H. Kaetsu, H. Asama, Modelling environment and task for cooperative team play, Journal of Distributed Autonomous Systems 3 (6) (1998) 361–370.
[12] M. Hunter, K. Kostiadis, H. Hu, A behaviour-based approach to position selection for simulated soccer agents, in: Proceedings of the RoboCup Euro 2000 Workshop, 2000.
[13] R. Brooks, A robust layered control system for a mobile robot, IEEE Journal of Robotics and Automation 2 (1986) 14–23.
[14] M. Bratman, Intention, Plans, and Practical Reason, Harvard University Press, Cambridge, MA, 1987.
[15] I. Ferguson, Touringmachines: An architecture for dynamic, rational, mobile agents, Ph.D. Thesis, University of Cambridge, Clare Hall, Cambridge, 1992.
[16] M. Wooldridge, Intelligent agents, in: G. Weiss (Ed.), Multiagent Systems, MIT Press, Cambridge, MA, 1999.
[17] K. Fischer, An agent-based approach to holonic manufacturing systems, in: Proceedings of the BASYS'98 Third IEEE/IFIP International Conference, 1998, pp. 3–12.
[18] M. Jamzad, A. Foroughnassiraei, E. Chiniforooshan, R. Ghorbani, M. Kazemi, H. Chitsaz, F. Mobasser, S.B. Sadjad, Arvand, in: M. Veloso, E. Pagello, H. Kitano (Eds.), Proceedings of the RoboCup'99: Robot Soccer World Cup III, Lecture Notes on Artificial Intelligence, Vol. 1856, Springer, Berlin, 1999, pp. 61–73.
[19] T. Balch, R.C. Arkin, Communication in reactive multiagent robotic systems, Autonomous Robots 1 (1) (1994) 27–52.
[20] D. Nardi, G. Adorni, A. Bonarini, A. Chella, G. Clemente, E. Pagello, M. Piaggio, ART'99: Azzurra robot team, in: M. Veloso, E. Pagello, H. Kitano (Eds.), RoboCup'99: Robot Soccer World Cup III, Lecture Notes on Artificial Intelligence, Vol. 1856, Springer, Berlin, 2000, pp. 695–698.
[21] M. Plagge, R. Gunther, J. Ihlenburg, D. Jung, A. Zell, The Attempto RoboCup robot team, in: M. Veloso, E. Pagello, H. Kitano (Eds.), RoboCup'99: Robot Soccer World Cup III, Lecture Notes on Artificial Intelligence, Vol. 1856, Springer, Berlin, 1999, pp. 424–433.
[22] R.C. Arkin, T. Balch, Cooperative multiagent robotic systems, in: D. Kortenkamp, R.P. Bonasso, R. Murphy (Eds.), AI-Based Mobile Robots: Case Studies of Successful Robot Systems, MIT Press, Cambridge, MA, 1998, pp. 277–296.
[23] D. Dudek, M. Jenkin, E. Milios, D. Wilkes, A taxonomy for multi-agent robotics, Autonomous Robots 3 (4) (1996) 375–397.
[24] Y.U. Cao, A. Fukunaga, A. Kahng, Cooperative mobile robotics: Antecedents and directions, Autonomous Robots 4 (1997) 1–23.
[25] R. Alami, F. Ingrad, S. Qutub, A scheme for coordinating multi-robot planning activities and plans execution, in: H. Prade (Ed.), Proceedings of the 13th European Conference on Artificial Intelligence (ECAI'98), Brighton, UK, 1998.
[26] E. Ephrati, J.S. Rosenschein, Divide and conquer in multi-agent planning, in: Proceedings of the 12th American Conference on Artificial Intelligence (AAAI'94), Seattle, WA, AAAI Press, Menlo Park, CA, 1994, pp. 375–380.
[27] B.B. Werger, Cooperation without deliberation: A minimal behavior-based approach to multi-robot teams, Artificial Intelligence 110 (2) (1999) 293–320.
[28] J.-S. Gutmann, T. Weigel, B. Nebel, Fast, accurate, and robust self-localization in the RoboCup environment, in: M. Veloso, E. Pagello, H. Kitano (Eds.), Proceedings of the RoboCup'99: Robot Soccer World Cup III, Lecture Notes on Artificial Intelligence, Vol. 1856, Springer, Berlin, 2000, pp. 304–317.
[29] K. Yokota, K. Ozaki, N. Watanabe, A. Matsumoto, D. Koyama, T. Ishikawa, K. Kawabata, H. Kaetsu, H. Asama, Uttori United: Cooperative team play based on communication, in: Proceedings of the RoboCup'98: Robot Soccer World Cup II, Lecture Notes on Artificial Intelligence, Vol. 1604, Springer, Berlin, 1999, pp. 479–484.
[30] K. Kostiadis, H. Hu, A multi-threaded approach to simulated soccer agents for the RoboCup competition, in: Proceedings of the 16th IJCAI RoboCup Workshop, IJCAI-99, Stockholm, Sweden, 1999, pp. 137–142.
[31] H. Hu, I. Kelly, D. Keating, D. Vinagre, Coordination of multiple mobile robots via communication, in: Proceedings of the SPIE'98, Mobile Robots XIII Conference, SPIE, 1998, pp. 94–103.
[32] M. Mataric, Interaction and intelligent behaviour, Technical Report AITR-1495, MIT AI Lab., Cambridge, MA, 1994.
[33] J. Wyms, H. Van Brussel, L. Bogaerts, Design pattern for integrating centralised scheduling in distributed holonic manufacturing control systems, in: Proceedings of the Second International Workshop on Intelligent Manufacturing Systems, 1999, pp. 75–82.
[34] B. Barbat, Holons, agents, and threads in anthropocentric systems, Studies in Informatics and Control 9 (3) (2000) 253–268.
[35] C. Candea, M. Staicu, B. Barbat, Holon-like approach for robotic soccer, in: Proceedings of the RoboCup Euro 2000 Workshop, 2000.
[36] R. Hino, T. Moriwaki, Decentralized scheduling in holonic manufacturing system, in: Proceedings of the Second International Workshop on Intelligent Manufacturing Systems, 1999, pp. 41–47.
[37] M. Giebels, H. Kals, H. Zijm, Building holarchies for concurrent manufacturing planning and control, in: Proceedings of the Second International Workshop on Intelligent Manufacturing Systems, 1999, pp. 49–56.
[38] M. Piaggio, A. Sgorbissa, R. Zaccaria, A programming environment for real-time control of distributed multiple robotic systems, Advanced Robotic Journal 14 (1) (2000) 75–86.
[39] M. Piaggio, R. Zaccaria, An information exchange protocol in a multi-layer distributed architecture, in: IEEE Proceedings of the Hawaii International Conference on Complex Systems, Vol. V, 1997, pp. 230–237.
[40] R. Engelmore, T. Morgan (Eds.), Blackboard Systems, Addison-Wesley, Reading, MA, 1988.
[41] B.P. Gerkey, M.J. Mataric, Murdoch: Publish/subscribe task allocation for heterogeneous agents, in: Proceedings of the Fourth International Conference on Autonomous Agents, ACM Press, Barcelona, 2000, pp. 203–204.
[42] M. Piaggio, R. Zaccaria, Distributing a robotic system on a network: the ethnos approach, Advanced Robotic Journal 12 (8) (1998) 743–758.

[43] C. Castelpietra, L. Iocchi, D. Nardi, M. Piaggio, A. Scalzo, A. Sgorbissa, Coordination among heterogenous robotic soccer players, in: Proceedings of the International Conference on Intelligent Robots and Systems (IROS'2000), IRS/IEEE Press, 2000.
[44] M. Veloso, P. Stone, Individual and collaborative behaviors in a team of homogeneous robotic soccer agents, in: Proceedings of the Third International Conference on Multi-agent Systems, 1998, pp. 309–320.
[45] M. Hannebauer, J. Wendler, P. Gugenberger, H. Burkhard, Emergent cooperation in a virtual soccer environment, in: Proceedings of the Distributed Autonomous Robotic Systems (DARS'98), Springer, Berlin, 1998.
[46] M. Asada, H. Kitano, M. Veloso, RoboCup: Today and tomorrow: What we have learned, Artificial Intelligence 110 (2) (1999) 193–214.

**Ciprian Candea** is a Researcher at Artificial Intelligence Research Group, Sibiu. His research interests include multi-agent architecture, human–computer interaction, information retrieval, and group decision support.

**Huosheng Hu** is a Senior Lecturer in the Department of Computer Science, University of Essex. His main research interests lie in the area of sensor integration, distributed computer architectures, intelligent control algorithms, path planning, behaviour and hybrid control architectures, communication and cooperation among multiple mobile robots, as well as service robots for home, office and hospitals. He is a Chartered Engineer, a senior member of IEEE, and a member of IEE, AAAI, IAS, IASTED and ACM.

**Luca Iocchi** is a Post-Doc at University of Rome "La Sapienza". His main research interests in the area of cognitive robotics and mobile robotics are action planning, self-localization, and coordination among multiple robots.

**Daniele Nardi** is a Full Professor of Computer Science at the Faculty of Engineering, Dipartimento di Informatica e Sistemistica, University of Rome "La Sapienza". His research interests include theoretical aspects of knowledge representation and reasoning, such as description logics and non-monotonic reasoning, as well as applications of knowledge-based systems in cognitive robotics and information integration. He is a member of IEEE, ACM, AIIA, AICA.

**Maurizio Piaggio** is a Research Assistant at the Department of Communication, Computer and System Sciences of the University of Genova, Italy. His research interests include autonomous robotics, service robotics applications, agent-based architectures and hybrid cognitive modeling. He is a member of the Italian Artificial Intelligence Society.