Task Decomposition, Dynamic Role Assignment, and Low-Bandwidth Communication for Real-Time Strategic Teamwork *

Peter Stone and Manuela Veloso
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
{pstone,veloso}@cs.cmu.edu
http://www.cs.cmu.edu/{~pstone,~mmv}

Artificial Intelligence, 110(2):241-273, June 1999.

Abstract

Multi-agent domains consisting of teams of agents that need to collaborate in an adversarial environment offer challenging research opportunities. In this article, we introduce *periodic team synchronization* (PTS) domains as time-critical environments in which agents act autonomously with low communication, but in which they can periodically synchronize in a full-communication setting. The two main contributions of this article are a flexible team agent structure and a method for inter-agent communication in domains with unreliable, single-channel, low-bandwidth communication. First, the novel team agent structure allows agents to capture and reason about team agreements. We achieve collaboration between agents through the introduction of *formations*. A formation decomposes the task space defining a set of *roles*. Homogeneous agents can flexibly switch roles within formations, and agents can change formations dynamically, according to pre-defined triggers to be evaluated at run-time. This flexibility increases the performance of the overall team. Our teamwork structure further includes pre-planning for frequent situations. Second, the novel communication method is designed for use during the low-communication periods in PTS domains. It overcomes the obstacles to inter-agent communication in multi-agent environments with unreliable, high-cost, low-bandwidth communication. We fully implemented both the flexible teamwork structure and the communication method in the domain of simulated robotic soccer, and conducted controlled empirical experiments to verify their effectiveness. In addition, our simulator team made it to the semi-finals of the RoboCup-97 competition, in which 29 teams participated. It achieved a total score of 67–9 over six different games, and successfully demonstrated its flexible teamwork structure and inter-agent communication.

1 Introduction

A multi-agent system which involves several agents that collaborate towards the achievement of a joint objective is viewed as a *team* of agents. Most proposed teamwork structures (e.g. joint intentions [7], shared plans [13]) rely on agents in a multi-agent system to negotiate and/or contract with each other in order to initiate team plans. However, in dynamic, real-time domains with limited communication, complex negotiation protocols may take too much time and/or be infeasible due to communication restrictions.

*This research is sponsored in part by the DARPA/RL Knowledge Based Planning and Scheduling Initiative under grant number F30602-95-1-0018. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies or endorsements, either expressed or implied, of the U. S. Government.

Our work has been focused in time-critical environments in which agents in a team alternate between periods of limited and unlimited communication. This focus motivates the introduction of *Periodic Team Synchronization* (PTS) domains. In PTS domains, during the limited communication periods, agents need to act autonomously, while still working towards a common team goal. Time-critical environments require real-time response and therefore eliminate the possibility of heavy communication among team agents. However, in PTS domains, agents can periodically synchronize in a safe, full-communication setting.

Our work is situated in an example of a PTS domain, simulated robotic soccer [16]. In robotic soccer, teams can plan strategies before the game, at halftime, or at other breakpoints, but during the course of the game, communication is limited. The *soccer server* [25] is a widely used robotic soccer simulator with a single, low-bandwidth, unreliable communication channel for all agents in the environment [1].

In this article, we describe our general team member agent architecture suitable for creating teams of agents in PTS domains. This architecture includes a mechanism for defining pre-determined multi-agent protocols accessible to the entire team, called *locker-room agreements*. Within this team member agent architecture and using the locker-room agreement, we define our flexible teamwork structure that allows for task decomposition and dynamic role assignment in PTS domains. In addition, we define a communication protocol in service of the locker-room agreement that is suitable for use during low-communication periods in a class of PTS domains: domains with low-bandwidth, single channel communication environments.

The team member agent architecture described in this article defines a complete agent, including perception, cognition, and action. It is fully implemented as a simulated robotic soccer team.

This article is organized as follows. Section 2 expands upon the introduction of PTS domains and the two main contributions of this article: a flexible teamwork structure and a low-bandwidth communication paradigm. Section 3 presents the general agent architecture within which both the flexible teamwork structure and the low-bandwidth communication paradigm are situated. Sections 4 and 5 formally present the teamwork structure and the communication paradigm respectively. Section 6 gives details of our full implementations of both main contributions of this article within the simulated robotic soccer domain. Section 7 presents extensive empirical results testing the effectiveness of these implementations. Section 8 is devoted to discussion and related work and Section 9 concludes.

2 PTS Domains

We define periodic team synchronization domains as domains with the following characteristics:

- There is a team of autonomous agents A that collaborate towards the achievement of a joint long-term goal G.
- Periodically, the team can synchronize with no restrictions on communication: the agents can in effect inform each other of their entire internal states and decision-making mechanisms with no adverse effects upon the achievement of G. These periods of full communication can be thought of as times at which the team is "off-line."
- In general (i.e., when the agents are "on-line"):
  - The domain is *dynamic* and *real-time* meaning that team performance is adversely affected if an agent ceases to act for a period of time: G is either less likely to be achieved, or likely to be achieved farther in the future. That is, consider agent a_i. Assume that all other agent behaviors are fixed and that were a_i to act optimally, G would be achieved with probability p at time t. If a_i stops acting for any period of time and then resumes acting optimally, either:
    - G will be achieved with probability p' at time t with $p' < p$; or

    * G will be achieved with probability p at time t' with $t' > t$.

- The domain has *unreliable communication*, either in terms of transmission reliability or bandwidth limits. In particular:
    * If an agent $a_i \in A$ sends a message m intended for agent $a_j \in A$, then m arrives with some probability $q < 1$; or
    * Agent a_i can only receive x messages every y time units.

In the extreme, if $q = 0$ or if $x = 0$, then the periods of full communication are interleaved with periods of *no* communication, requiring the agents to act completely autonomously. In all cases, there is a cost to *relying* on communication. If agent a_i cannot carry on with its action until receiving a message from a_j, then the team's performance could suffer. Because of the unreliable communication, the message might not get through on the first try. And because of the dynamic, real-time nature of the domain, the team's likelihood of or efficiency at achieving G is reduced.

Robotic soccer is a PTS domain since teams can plan strategies before the game, at halftime, or at other breakpoints; but during the course of the game, communication is limited. For example, the soccer server's communication protocol involves a single, low-bandwidth, unreliable communication channel for all 22 agents [1].

In PTS domains, teams are long-term entities so that it makes sense for them to have periodic, reliable, private synchronization opportunities in which they can form off-line agreements for future use in unreliable, time-critical environments. This view of teams is complementary to teams that form on the fly for a specific action and keep communicating throughout the execution of that action as in [7]. Instead, in PTS domains, teams define coordination protocols during the synchronization opportunity and then disperse into the environment, acting autonomously with limited or no communication possible.

It has been claimed that pre-determined team actions are not flexible or robust to failure [40]. In the context of PTS domains, a key contribution of our work is the demonstration that pre-determined multi-agent protocols can facilitate effective teamwork while retaining flexibility. We call these pre-determined protocols *locker-room agreements*. Formed during the periodic synchronization opportunities, locker-room agreements are remembered identically by all agents and allow them to coordinate efficiently.

In this article, we present the *team member agent architecture*, an agent architecture suited for team agents in PTS domains. The architecture allows for an agent to act collaboratively based on locker-room agreements.

A first approach to PTS domains is to break the task at hand into multiple rigid roles, assigning one agent to each role. Thus each component of the task is accomplished and there are no conflicts among agents in terms of how they should accomplish the team goal. However such an approach is subject to several problems: inflexibility to short-term changes (e.g. one robot is non-operational), inflexibility to long-term changes (e.g. a route is blocked), and a lack of facility for reassigning roles.

We introduce instead *formations* as a teamwork structure within the team member agent architecture. A formation decomposes the task space defining a set of roles with associated behaviors. In a general scenario with heterogeneous agents, subsets of homogeneous agents can flexibly switch roles within formations, and agents can change formations dynamically.

Within these PTS domains and our flexible teamwork structure, several challenges arise. Such challenges include:

- how to represent and follow locker-room agreements;
- how to determine the appropriate times for agents to change roles and/or formations;
- how to ensure that all agents are using the same formation; and

- how to ensure that all roles in a formation are filled: since the agents are autonomous and do not share memory, they could easily become uncoordinated.

Also within the team member agent architecture, we introduce a communication paradigm appropriate for agents in PTS domains with single-channel, low-bandwidth, unreliable communication during the dynamic, real-time (on-line) phases of operation. Not all PTS domains have such communication environments, but agents operating in those that do can implement this communication paradigm within their locker-room agreements.

In a nutshell, the contributions of this article are: the introduction of the concepts of PTS domains and locker-room agreements; the definition of a general team member agent architecture structure for defining a flexible teamwork structure; the facilitation of smooth transitions among roles and entire formations; a method for using roles to define pre-compiled multi-step, multi-agent plans; and techniques for dealing with the obstacles to inter-agent communication during the low-communication periods of PTS domains with single-channel, low-bandwidth, unreliable communication during the "on-line" periods.

In addition to simulated robotic soccer, there are several other examples of PTS domains, such as hospital/factory maintenance [9], multi-spacecraft missions [30], search and rescue, and battlefield combat [40]. There are also several other domains with similar communication requirements to the ones considered here. For example, aural communication in crowded settings is one. Both people and robots using aural sensors ([11]) must contend with multiple simultaneous audible streams. They also have a limit to the amount of sound they can process in a given amount of time, as well as to the range within which communication is possible. Another example of such a communication environment is arbitrarily expandable systems. If agents aren't aware of what other agents exist in the environment, then all agents must use a single universally-known communication channel, at least in order to initiate communication.

3 Architecture Overview

The team member agent architecture is suitable for PTS domains. Individual agents can capture locker-room agreements and respond to the environment, while acting autonomously. Based on a standard agent paradigm, our team member agent architecture allows agents to sense the environment, to reason about and select their actions, and to act in the real world. At team synchronization opportunities, the team also makes a locker-room agreement for use by all agents during periods of limited communication. Figure 1 shows the functional input/output model of the architecture.

The agent keeps track of three different types of state: the *world state*, the *locker-room agreement*, and the *internal state*. The agent also has two different types of behaviors: *internal behaviors* and *external behaviors*.

The world state reflects the agent's conception of the real world, both via its sensors and via the predicted effects of its actions. It is updated as a result of interpreted sensory information. It may also be updated according to the predicted effects of the external behavior module's chosen actions. The world state is directly accessible to both internal and external behaviors.

The locker-room agreement is set by the team when it is able to privately synchronize. It defines the flexible teamwork structure and the inter-agent communication protocols, if any. The locker-room agreement is accessible only to internal behaviors.

The internal state stores the agent's internal variables. It may reflect previous and current world states, possibly as specified by the locker-room agreement. For example, the agent's role within a team behavior could be stored as part of the internal state. A window or distribution of past world states

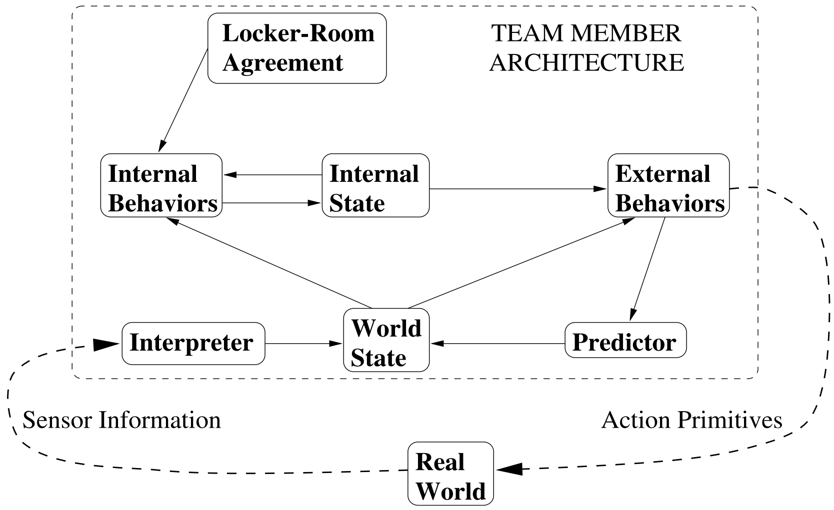

Figure 1: A functional input/output model of the team member agent architecture for PTS domains.

could also be stored as a part of the internal state. The agent updates its internal state via its internal behaviors.

The internal behaviors update the agent's internal state based on its current internal state, the world state, and the team's locker-room agreement.

The external behaviors reference the world and internal states, and select the actions to send to the actuators. The actions affect the real world, thus altering the agent's future percepts. External behaviors consider only the world and internal states, without direct access to the locker-room agreement.

Internal and external behaviors are similar in structure, as they are both sets of condition/action pairs where conditions are logical expressions over the inputs and actions are themselves behaviors as illustrated in Figure 2. In both cases, a behavior is a directed acyclic graph (DAG) of arbitrary depth. The leaves of the DAGs are the behavior types' respective outputs: internal state changes for internal behaviors and action primitives for external behaviors. One leaf is illustrated in Figure 2.

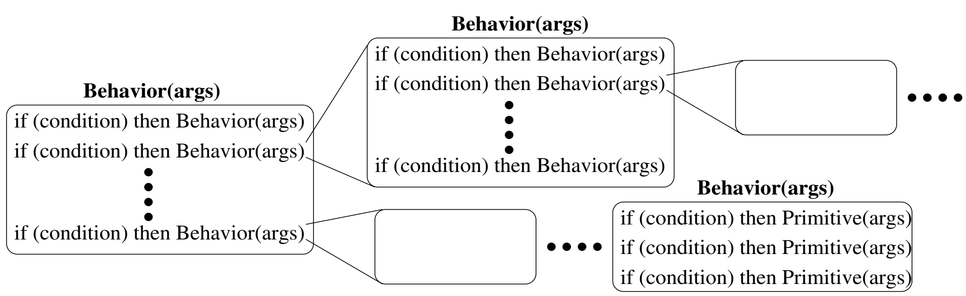

Figure 2: Behaviors in the team member agent architecture. Both internal and external behaviors are organized as directed acyclic graphs.

This notion of behavior is consistent with that laid out in [22]. In particular, behaviors can be nested at different levels: selection among lower-level behaviors can be considered a higher-level behavior, with the overall agent behavior considered a single "do-the-task" behavior. There is one such *top-level* internal behavior and one top-level external behavior; they are called when it is time to update the internal state or act in the world, respectively.

The following section introduces the teamwork structure that builds upon this team member agent architecture. The teamwork structure is designed for use in PTS domains. It exploits the locker-room agreement and the behavior definitions of the team member agent architecture.

4 Teamwork Structure

Common to all players, the locker-room agreement includes the team structure while team members are acting in a time-critical environment with limited or no communication. In this section, we present our teamwork structure. It defines:

1. Flexible agent *roles* with protocols for switching among them;
2. Collections of roles built into team *formations*; and
3. Multi-step, multi-agent plans for execution in specific situations: *set-plays*.

The teamwork structure indirectly affects the agent external behaviors by changing the agents' internal states via internal behaviors.

4.1 Roles

A *role*, r, consists of a specification of an agent's internal and external behaviors. The conditions and arguments of any behavior can depend on the agent's current role, which is a function of its internal state. At the extreme, a top-level external behavior could be a switch, calling an entirely different behavior subgraph for each possible role. However, the role can affect the agent's overall behavior at any level of its behavior graph: it could affect just the arguments of a behavior deeply embedded in the behavior graph.

Roles may be *rigid*, completely specifying an agent's behavior. Or they may be *flexible*, leaving a certain degree of autonomy to the agent filling the role. For example, consider an agent that has access to a clock and that can blow a whistle. Role r could rigidly specify that the agent filling it must blow a whistle on the hour every hour. On the other hand, role r could leave some flexibility to the agent that fills it, specifying that no fewer than 25% but no more than 75% of the times that the hour changes, the agent must blow a whistle. In this case, the agent must stay within a parametric range in order to successfully fill the role, but on every given hour change, it can choose for itself what to do. By specifying ranges of parameters or behavior options, the agent filling role r can be given an arbitrary amount of flexibility.

A role in the robotic soccer domain, can be a position such as a midfielder. In the hospital maintenance domain, a role could specify the wing of the hospital whose floors the appropriate agent should keep clean, while in the web search domain, it could specify a server to search.

4.2 Formations

We achieve collaboration between agents through the introduction of *formations*. A formation decomposes the task space defining a set of roles. Formations include as many roles as there are agents in the team, so that each role is filled by one agent. In addition, formations can specify sub-formations, or *units*, that do not involve the whole team. A unit consists of a subset of roles from the formation, a *captain*, and intra-unit interactions among the roles.

For a team of n agents $A = \{a_1, a_2, \ldots, a_n\}$, any formation is of the form

$$F = \{R, \{U_1, U_2, \ldots, U_k\}\}$$

where R is a set of roles $R = \{r_1, r_2, \ldots, r_n\}$ such that $i \neq j \Rightarrow r_i \neq r_j$. Note that there are the same number of roles as there are agents. However, it is possible to define redundant roles such that the behavior specification of r_i is equivalent to that of r_j ($i \neq j$). Each unit U_i is a subset of R: $U_i = \{r_{i1}, r_{i2}, \ldots, r_{ik}\}$ such that $r_{ia} \in R$, $a \neq b \Rightarrow r_{ia} \neq r_{ib}$ and r_{i1} is the *captain*, or unit leader. The map $A \mapsto R$ is not fixed: roles can be filled by different homogeneous agents. A single role may be a part of any number of units and formations.

Units are used to deal with local problem solving issues. Rather than involving the entire team in a sub-problem, the roles that address it are organized into a unit. Captains are unit-members with special privileges in terms of directing the other unit members.

Roles and formations are introduced independently from the agents that are to fill them. The locker-room agreement specifies an initial formation; an initial map from agents to roles; and run-time triggers for dynamic changing of formations. At any given time, each agent has an opinion as to what formation the team is currently using. Agents keep mappings $A \mapsto R$ from teammates to roles in the current formation. All this team structuring information is stored in the agent's internal state. It can be altered via the agent's internal behaviors.

Since agents are autonomous and operating in a PTS domain, during the periods of limited communication there is no guarantee that they will all think that the team is using the same formation, nor that they have accurate maps $A \mapsto R$. In fact, the only guarantee is that each agent knows its own current role. Thus, in our implementation of the teamwork structure, we create robust behaviors for team agents which do not depend upon having correct, up-to-date knowledge of teammates' internal states: they degrade gracefully. When limited communication is available, efficient low-bandwidth communication protocols can allow agents to inform each other of their roles periodically. Figure 3 illustrates a team of agents smoothly switching roles and formations over time.

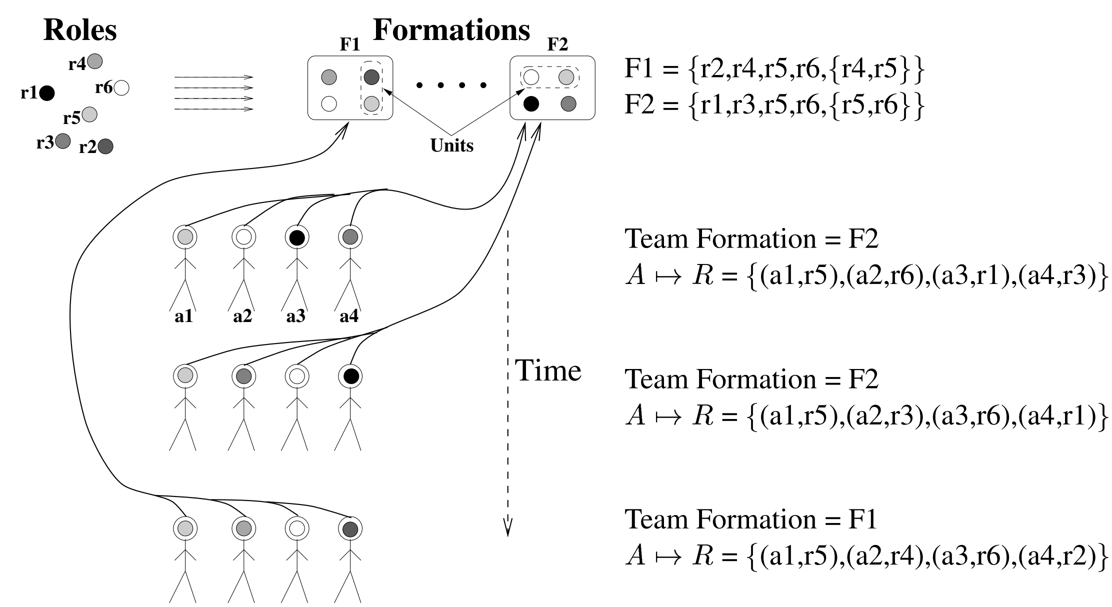

Figure 3: A team of agents smoothly switching roles and formations over time. Different roles are represented as differently shaded circles. Formations are possibly overlapping collections of roles. Units within the formations are indicated within a dotted enclosure. The definitions of all roles, formations, and units are known to all agents. An agent's current role is indicated by the shaded circle in its head and its current formation is indicated by an arrow to the formation. The agents first switch roles while staying in the same formation; then they switch to an entirely new formation.

4.3 Set-Plays

As a part of the locker-room agreement, the team can define multi-step multi-agent plans to be executed at appropriate times. Particularly if there are certain situations that occur repeatedly, it makes sense for the team to devise plans for those situations ahead of time. We define a set-play as the combination of:

- A *trigger* condition indicating the set of states in which the set-play is activated; and
- A set of *set-play roles* $R_{sp} = \{spr_1, \ldots, spr_m\}$, $m \leq n$ defining the actions to be taken by the participants in the set-play. Each set-play role spr_i includes:
  - A *set-play behavior* to be executed; and
  - A *termination* condition indicating the set of states in which an agent should cease filling the set-play role and resume its normal behavior.

The set-plays are defined in the locker-room agreement so that they are known to all agents on the team. Note that a set-play need not involve the entire team: $m \leq n$. The locker-room agreement also includes a general function to map roles in a formation to roles in a set-play: $F \mapsto R_{sp}$. Thus set-play roles are not assigned to pre-determined agents; instead they are filled by whichever agent is filling the appropriate role in the team's current formation.

5 Communication Paradigm

The teamwork structure defined in Section 4 is designed to be appropriate for all PTS domains. In the subclass of PTS domains with single-channel, low-bandwidth, unreliable communication during the periods of limited communication, such as the soccer server, the communication paradigm defined in this section is also appropriate. The communication paradigm further illustrates the use of the locker-room agreement within the team member agent architecture. Recall that PTS domains may have no communication possible during the "on-line" periods. In those cases, of course, the communication paradigm presented here does not apply.

Domains with single-channel, low-bandwidth, unreliable communication raise several challenges for inter-agent communication. The locker-room agreement can be used to make inter-agent communication more efficient and reliable. The five challenges addressed by our communication approach are:

1. Team members need some method of identifying which messages on the single channel are intended for which agent.
2. Since there is a single communication channel, agents must be prepared for active interference by hostile agents. A hostile agent could mimic messages it has previously heard at random times.
3. Since the communication channel has low bandwidth, the team must prevent itself from "talking all at once." Many communication utterances call for responses from all team members. However, if all team members respond simultaneously, few of the responses will get through.
4. Since communication is unreliable, agents must be robust to lost messages: their behaviors cannot depend upon receiving communications from a teammate.
5. Teams must determine how to maximize the chances that they are using the same team strategy (formation) despite the facts that each is acting autonomously and that communication is unreliable.

Communication Environment	**Challenges**
• Many agents, teams • Single-channel • Low-bandwidth • Unreliable	• Message targeting and distinguishing • Robustness to active interference • Multiple simultaneous responses • Robustness to lost messages • Team coordination

Table 1: The characteristics and challenges of the type of communication environment considered in this article.

The characteristics and challenges of this communication environment are summarized in Table 1.

In order to meet these challenges, a team uses messages with the following fields, all of whose syntax and semantics are defined within the locker-room agreement:

- The <team-identifier> identifies messages from within the team as opposed to another team in an adversarial environment.
- The <unique-team-member-ID> is a different sequential integer assigned to each team member.
- The <encoded-time-stamp> is a security code that can be used to verify a message's authenticity.
- The <time-stamped-team-strategy> indicates the current formation that the sender believes the team is using.
- The <selected-internal-state> can contain portions of the sender's internal state.
- The <message-type> and <message-data> contain the semantic content of the individual message. The messages can use any syntactic and semantic codes (KQML [10] and KIF [12] for example). The locker-room agreement also includes a mapping from message type to response requirements.
- The <target> indicates the intended recipient(s) of the message. It could be intended for a single team member identified either by <unique-team-member-ID> or by role within the team's current formation; for a unit of the current formation; or for all team members.

In addition to the protocol defined within the locker-room agreement, some internal state variables need to be devoted to communication. When an agent hears a message, it interprets it and updates the world state to reflect any information transmitted by the message. It also stores the content of the message as a special variable `last-message`. Furthermore, based on the locker-room agreement, an internal behavior then updates the internal state. If the message requires a response, three variables in the internal state are manipulated by an internal behavior: `response`, `response-flag`, and `communicate-delay`. `response` is the actual response that should be given by the agent as determined in part by the locker-room agreement. All three of these variables are then referenced by an external behavior to determine when a response should be given. For example one condition-action pair of the top-level external behavior might be: `if (response-flag set and communicate-delay=0) then say(response).`

The remainder of this section details how these particular message fields and internal state variables can be used to meet the challenges summarized in Table 1.

5.1 Message Targeting and Distinguishing

Since there is a single communication channel, agent a_i hears messages sent by all agents whether or not they are intended for it. Messages sent by agents from another team are completely ignored. Messages sent by a teammate are identified by the <team-identifier> field. Since all team members know the locker-room agreement, agents monitor all messages from teammates to determine their teammates' internal states, even if the content of the message is intended for another teammate.

Agents can distinguish messages that are intended for them by checking the <team-identifier> and <target> fields. An agent a_i pays attention to a message from a member of the same team that is targeted to a_i, to the entire team, or to some subset of the team that includes a_i. The <target> field could identify an individual agent either by its unique ID number or by the role that it is currently playing. Thus, a message could be sent to the agent playing a particular role without knowing which agent that is. Similarly, a message could be targeted towards all agents in a unit of the team's current formation.

5.2 Robustness to Active Interference

The only further difficulty related to an agent distinguishing which messages are intended for it arises in the presence of active interference. Consider a hostile agent h which hears a message that is directed to a_i at time t. h has full access to the message since all agents use the same communication channel. Thus if h remembers the message and sends an identical message at time u, agent a_i will mistakenly believe that the message is from a teammate. Although the message was appropriate at time t, it may be obsolete at time u and it could potentially confuse a_i as h intends.

This potential difficulty is avoided with the <encoded-time-stamp> field. Even a simple time stamp is likely to safeguard against interference since h is not privy to the locker-room agreement: it does not necessarily know which field is the time stamp. However, if h discovers which field is the time stamp by noticing that it always matches the time of the message, it could alter the field based on the time elapsed between times t and u. Indeed, if there is a globally accessible clock, h would simply have to replace t with u in the message. However, the team can safeguard against such interference techniques by encoding the time-stamp using an injective function chosen as a part of the locker-room agreement. This function can use any of the other message fields as arguments in order to make decryption as difficult as possible. The only requirement is that a teammate receiving the message can invert the function to determine the time at which the message was sent. If the time at which it was sent is either too far in the past or in the future (according to the locker-room agreement), then the message can be safely ignored. In particular, the locker-room agreement has a variable `message-lag-tolerance` encoding this time. If a message sent at time t arrives at time u with $u - t >$ `message-lag-tolerance`, then the message is ignored.

By observing enough messages and comparing them with the actual time, it is theoretically possible for hostile agents to crack simple codes and alter the <encoded-time-stamp> field appropriately before sending a false message. However, the function can be made arbitrarily complex so that such a feat is intractable within the context of the domain. If secrecy is critical and computation unconstrained, a theoretically safe encryption scheme can be used. The degree of complexity necessary depends upon the number of messages that will be sent after the locker-room agreement. With few enough messages, even a simple linear combination of the numerical message fields may suffice.

5.3 Multiple Simultaneous Responses

The next challenge to meet is that of messages that require responses from several teammates. However, not all messages are of this type. For example, a message meaning "where are you?" requires a response, while "look out behind you" does not. Therefore it is first necessary for agents to classify messages in terms of whether or not they require responses as a function of the <message-type> field. Since the low-bandwidth channel prevents multiple simultaneous responses, the agents must also reason about the number of intended recipients as indicated by the <target> field. Taking these two factors into account, there are six types of messages, indicated here as a1,a2,a3,b1,b2, and b3:

	Response requested	
Message Target	no	yes
Single agent	a1	b1
Whole team	a2	b2
Part of team	a3	b3

When hearing any message, the agent updates its internal beliefs of the other agent's status as indicated by the <time-stamped-team-strategy> and <selected-internal-state> fields. However, only when the message is intended for it does it consider the content of the message. Then it uses the following algorithm in response to the message:

1. If the message requires no response (types a1–3), the agent simply updates its internal state.
2. If the message requires a response then set `response` to the appropriate response message, `response-flag` = 1 and
   - If the agent was the only target (type b1), respond immediately: `communicate-delay` = 0;
   - If the message is sent to more than one target (types b2 and b3), set `communicate-delay` based on the difference between the <unique-team-member-ID> of the message sender and that of the receiver. Thus each teammate responds at a different time, leaving time for teammate messages to go through.

An internal behavior keeps decrementing `communicate-delay` as time passes. An external behavior uses the communication condition-action pair presented above:

```
if (response-flag set and communicate-delay=0) then say(response)
```

where say is an actuator primitive. Players also set the `communicate-delay` variable in the event that they need to send multiple messages to the same agent in a short time. This communication paradigm allows agents to continue real-time acting while reasoning about the appropriate time to communicate.

5.4 Robustness to Lost Messages

In order to meet the challenge raised by unreliable communication leading to lost messages, agents must not depend on communication to act. Communication is structured so that it helps agents update their world and internal states. But agents do not stop acting while waiting for communications from teammates. As brought up in [39], such a case could cause infinite looping if a critical teammate fails to respond for any reason. As well as continuing to act while waiting for `communicate-delay` to expire, agents ensure that they do not rely on inter-agent communication by continuing to act while waiting for responses from teammates. They also maintain world and internal states without help from teammates. Communication can improve the reliability of an agent's world state by elucidating some of an agent's hidden state; but communication is not necessary for an agent to maintain a reasonable approximation of the world state.

5.5 Team Coordination

Finally, team coordination is difficult to achieve in the face of the possibility that autonomous team members may not agree on the <time-stamped-team-strategy> or the mapping from teammates to roles within the team strategy. Again, there are no disastrous results should team members temporarily adopt different strategies; however they are more likely to achieve their goal G if they can stay coordinated.

One method of coordination is via the locker-room agreement. Agents agree on globally accessible environmental cues as triggers for switches in team strategy. Another method of coordination which complements this first approach is via the time stamp. When hearing a message from a teammate indicating that the team strategy is different from the agent's current idea of the team strategy, the agent adopts the more recent team strategy: if the received message's team strategy has a time-stamp that is more recent than that on the agent's current team strategy, it switches; otherwise it keeps the same team strategy and informs its teammate of the change. Thus changes in team strategy can quickly propagate through the team.

In particular, suppose that agent a_i heard at time t that the team formation is F_1. It then hears a message from agent a_j indicating that the team formation was set to F_2 at time u. If $t < u$, then F_2 is a more recent team decision and it updates its notion of the team's formation to F_2. However, if $u < t$, it is agent a_j that has an obsolete view of the formation. a_i then sends a message to a_j indicating in the <time-stamped-team-strategy> field that the formation was set to F_1 at time t, thus causing a_j to update its notion of the team's formation. In the rare even that $t = u$, the locker-room agreement must specify an order of precedence among roles in order for the agents to determine which agent's idea of the current formation to regard as correct.

Depending on the available bandwidth in the particular application, the <selected-internal-state> can also be used to facilitate team coordination by helping to keep the team members up-to-date regarding the mapping $A \mapsto R$, and perhaps regarding object locations that might be hidden to individual agents.

6 Implementation in the Robotic Soccer Domain

Robotic soccer is a very good example of a PTS domain: teams can coordinate before the game, at half-time, and at other break points, but communication is limited during play [17]. Introduced by Mackworth [20], the robotic soccer domain has been gaining popularity as an AI and robotics test-bed, with systems having been recently developed both in simulation and with real robots [16]. Robotic soccer is a multi-agent domain with heterogeneous (at least two types of agents even if each team is homogeneous), communicating agents. For a survey of research issues arising in this type of domain, see [33]. The research presented in this article was first developed in simulation and some of it has also been successfully used on our real robot team [42].

The soccer server [25, 1], which serves as the substrate simulator for the research reported in this article, captures enough real-world complexities to be a very challenging domain. This simulator is realistic in many ways: (i) the players' vision is limited; (ii) the players can communicate by posting to a blackboard that is visible (but not necessarily intelligible) to all players; (iii) each player is controlled by a separate process; (iv) each team has 11 members; (v) players have limited stamina; (vi) actuators and sensors are noisy; (vii) dynamics and kinematics are modeled; and (viii) play occurs in *real time*: the agents must react to their sensory inputs at roughly the same speed as human or robotic soccer players. The soccer server was successfully used as the basis for the RoboCup-97 simulator competition in which 29 teams participated [17].

As presented in [1] The soccer server models a communication environment appropriate in a time-pressured, crowded environment. All 22 agents (11 on each team) use a single, unreliable communication channel. When one agent speaks, agents on both teams can hear the message immediately along with the (relative) direction from which it came. The speaker is not inherently known. Agents have a limited communication range, hearing only messages spoken from within a certain distance. They also have a limited communication capacity, hearing a maximum of 1 message every 200 msec (actions are possible every 100 msec, so if all other agents are speaking as fast as they can, only 1 of every 42 messages will be heard). Thus communication is extremely unreliable. We use version 3 of this simulator for the research reported in this article.

This section details the full implementations of both main contributions of this article: the flexible

• All 22 agents (including adversaries) on same channel • Limited communication range and capacity • No guarantee of sounds getting through • Instantaneous communication

Table 2: Characteristics of the soccer server communication model.

teamwork structure and the novel communication paradigm. Both implementations are unified within the same robotic soccer system: the CMUnited-97 simulator team [34].

6.1 Teamwork Structure Implementation

One approach to task decomposition in the soccer server is to assign fixed positions to agents.[1] Such an approach leads to several problems: i) short-term inflexibility in that the players cannot adapt their positions to the ball's location on the field; ii) long-term inflexibility in that the team cannot adapt to opponent strategy; and iii) local inefficiency in that players often get tired running across the field back to their positions after chasing the ball. Our formations allow for flexible teamwork and combat these problems. (As the term "position" is often used to denote the concept of "role" in the soccer domain, in this section we use the two terms interchangeably.)

This section describes the CMUnited-97 simulator team implementation of the teamwork structure presented in Section 4. In the context of the team member agent architecture in Section 3, it covers the locker-room agreement, the internal behaviors, and the internal state.

6.1.1 Domain Instantiations of Roles and Formations

Figure 4 shows a simplified top-level external behavior used by a team agent. The agent's top priority is to locate the ball. If the ball's location is known, it moves towards the ball or goes to its position (i.e., to assume its role), depending on its internal state. It also responds to any requested communications from teammates.

External Behavior: Play Soccer()

If (Ball Lost)	Face Ball()
If (Ball known AND Chasing)	Handle Ball(args1)
If (Ball known AND Not Chasing)	Passive Offense(args2)
If (Commuincate Flag Set)	Communicate()

Figure 4: An example of a simplified top-level external behavior for a robotic soccer player.

The referenced "handle ball" and "passive offense" behaviors may be affected by the agent's current role and/or formation. Such effects are realized by references to the internal state either at the level of function arguments (args1, args2), or within sub-behaviors. None of the actions in the condition-action pairs here are action primitives; rather, they are calls to lower level behaviors.

The definition of a position includes *home coordinates*, a *home range*, and a *maximum range*, as illustrated in Figure 5. The position's home coordinates are the default location to which the agent should go. However, the agent has some flexibility, being able to set its actual home position anywhere within the home range. When moving outside of the max range, the agent is no longer considered to be in the position. The home and max ranges of different positions can overlap, even if they are part of the same formations.

[1] One of the teams in Pre-RoboCup-96 [15] used and depended upon these assignments: the players would pass to the fixed positions regardless of whether there was a player there.

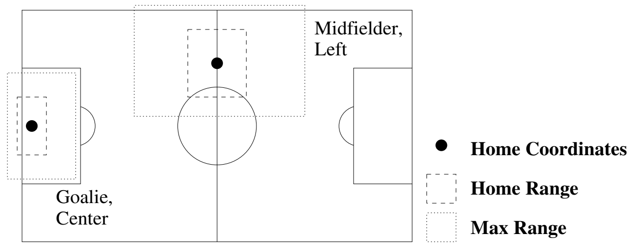

Figure 5: Different positions with home coordinates and home and max ranges.

A formation consists of a set of positions and a set of units (as defined in Section 4.2). The formation and each of the units can also specify inter-position behavior specifications for the member positions, as illustrated in Figure 6(a). In this case, the formations specify inter-role interactions, namely the positions to which a player should consider passing the ball. We use decision tree learning to help players decide where to pass [37]. Figure 6(b) illustrates the units, the roles involved, and their captains. Here, the units contain defenders, midfielders, forwards, left players, center players, and right players.

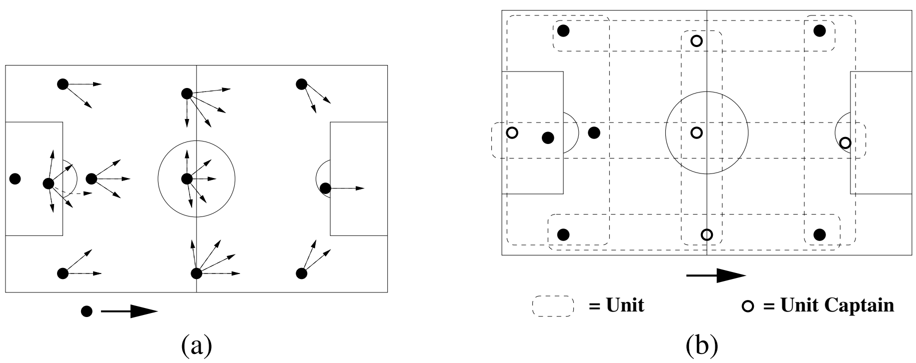

Figure 6: (a) A possible formation (4-3-3) for a team of 11 players. Arrows represent passing options. (b) Positions can belong to more than one unit.

Since the players are all autonomous, in addition to knowing its own role, each one has its own belief of the team's current formation along with the time at which that formation was adopted, and a map of teammates to positions. Ideally, the players have consistent beliefs as to the team's state, but this condition cannot be guaranteed between synchronization opportunities.

For example, consider the passive offense behavior in Figure 4. This external behavior references the agent's internal state via a series of function calls. Specifically, the agent is to move to some location within the home range of its current position in the team's current formation:

$$\texttt{TargetLocation} \in \texttt{HomeRange}(\texttt{MyPosition}(\texttt{CurrentFormation}())) \qquad (1)$$

where `HomeRange(p)` returns the home range of position p; `MyPosition(f)` returns a player's own current position in formation f; and `CurrentFormation()` returns the player's own opinion of the team's current formation. Thus the internal behaviors that determine the player's current position and formation affect its external behavior. Notice that by specifying a range of possible locations, the role leaves some flexibility to the autonomous player: it can choose which specific `TargetLocation` to move to within the range.

Different roles can also have entirely different external behaviors. As presented in Section 4.1, each role could have an entirely different external behavior subgraph.

Our teamwork structure for PTS domains allows for several significant features in our simulated soccer team. These features are: (i) the definition of switching among multiple formations with units; (ii) flexible position adjustment and position switching; (iii) and pre-defined special purpose plays (set-plays).

6.1.2 Dynamic Switching of Formations

We implemented several different formations, ranging from very defensive (8-2-0) to very offensive (2-4-4).[2] The full definitions of all of the formations are a part of the locker-room agreement. Therefore, they are all known to all teammates. However during the periods of full autonomy and limited communication, it is not necessarily known what formation the rest of the teammates are using. Three approaches can be taken to address this problem:

Static formation: the formation is set by the locker-room agreement and never changes;

Run-time formation switch: during team synchronization opportunities, the team sets globally accessible run-time evaluation metrics as formation-changing indicators.

Communication-triggered formation switch: one team member decides that the team should switch formations and communicates the decision to teammates.

Both run-time formation switches and communication-triggered formation switches are internal behaviors. The run-time triggers and communication protocols are defined in the locker-room agreement. When a run-time evaluation metric indicates that the formation should change, or when a heard communication triggers a formation change, an internal behavior changes the player's opinion of the team's formation in its internal state.

This change in internal state can then affect external behaviors. For example, a switch in formations changes the output of the `CurrentFormation()` function in Equation 1. The outputs of `MyPosition()` and `HomeRange()` are also altered: the new formation consists of a different collection of roles with different home ranges. Thus the passive offense external behavior changes as a result of the formation switch.

The CMUnited-97 simulator team uses run-time formation switches. Based on the amount of time left relative to the difference in score: the team switches to an offensive formation if it was losing near the end of the game and a defensive formation if it was winning. Specifically, the team starts out in a 4-4-2 formation. If $\Delta Minutes$ is the number of minutes left in the game, and $\Delta Score$ is the difference in score ($\Delta Score > 0$ if the team is winning; $\Delta Score < 0$ if the team is losing), then the team uses the following run-time algorithm:

- If $\Delta Score < 0$ and $-\Delta Score \geq \Delta Minutes$, then switch to a 3-3-4 formation;
- If $\Delta Score > 0$ and $\Delta Score \geq \Delta Minutes$, then switch to a 8-2-0 formation;
- Otherwise switch to (or stay in) a 4-4-2 formation.

Since each agent is able to independently keep track of the score and time, the agents are always able to switch formations simultaneously.

Communication-triggered formation switches have also been implemented and tested. Details are presented in the context of the communication paradigm implementation (Section 6.2).

[2] Soccer formations are typically described as the X-Y-Z where X, Y, and Z are the number of defenders, midfielders, and forwards respectively. It is assumed that the eleventh player is the goaltender [18].

6.1.3 Flexible Positions

As emphasized throughout, homogeneous agents can play different positions. But such a capability raises the challenging issue of when the players should change positions. In addition, with teammates switching positions, a player's internal player-position map $A \mapsto R$ could become incorrect and/or incomplete. The locker-room agreement provides procedures to the team that allow for coordinated role changing. In CMUnited's case, the locker-room agreement designates an order of precedence for the positions within each unit. Unless their pursuit of the ball leads them from their position, players only switch into a more important position than their current position.

By switching positions within a formation, the overall joint performance of the team can be improved. Position-switching has the potential to save player energy and to allow them to respond more quickly to the ball. However, switching positions can also cause increased player movement if a player has to move across the field to occupy its new position. Players must weight the possible costs and benefits before deciding to switch positions.

Like switching formations, switching positions can change external behaviors via their references to the internal state. In Equation 1, switching positions changes the value of returned by `MyPosition()`, thus also affecting the value of `HomeRange()`: the player executing the passive offense external behavior chooses its location from a different range of possible positions.

In addition to having the flexibility to switch to a different position, CMUnited-97 agents also have flexibility *within* their positions. That is, the external behavior references the internal state to determine a range of possible locations that are determined by the player's current position. However, within this range, the role does not specify the player's precise location. For example, in the passive offense external behavior (equation 1), the player must choose its `TargetLocation` from within the home range of its current position.

In the CMUnited multi-agent approach, the player positions itself flexibly such that it *anticipates* that it will be useful to the team, either offensively or defensively. The agents can exercise this flexibility within its external behaviors in two ways:

- Opponent marking;
- Ball-dependent positioning;

When marking opponents, agents move next to a given opponent rather than staying at the default position home. The opponent to mark can be chosen by the player (e.g., the closest opponent), or by the unit captain which can ensure that all opponents are marked, following a preset algorithm as part of the locker-room agreement.

When using ball-dependent positioning, the agent adjusts its location within its range based on the instantaneous position of the ball. For example, when the ball is on the same side of the field as the agent, the agent tries to move to a point on the line defined by its own goal and the ball. When the ball is on the other side of the field, the player adjusts its position back towards its own goal.

6.1.4 Pre-Planned Set-Plays

The final implemented improvement facilitated by our flexible teamwork structure is the introduction of set-plays, or pre-defined special purpose plays. As a part of the locker-room agreement, the team can define multi-step, multi-agent plans to be executed at appropriate times. Particularly if there are certain situations that occur repeatedly, it makes sense for the team to devise plans for those situations.

In the robotic soccer domain, certain situations occur repeatedly. For example, after every goal, there is a kickoff from the center spot. When the ball goes out of bounds, there is a goal-kick, a corner-kick, or a kick-in. In each of these situations, the referee informs the team of the situations. Thus all the players know to execute the appropriate set-play. A particular referee's message is the trigger condition for each set-play. Associated with each set-play role is a set-play behavior indicating a location on the field as well as an action to execute when the ball arrives. The player in a given role might pass to the player filling another role, shoot at the goal, or kick the ball to some other location. The termination condition for each role is either the successful execution of the prescribed action or the passage of a specified amount of time from the beginning of the set-play.

The locker-room agreement specifies that the roles in the current team formation are mapped to the set-play roles in the way requiring the least movement of agents from their position homes. That is $F \mapsto R_{sp}$ is chosen to minimize $\sum_{spr \in R_{sp}} Dist(r, spr)$ where $Dist(r, spr)$ is the distance from the home location of role r to the home location of its associated set-play role spr. This assignment of roles to set-play roles is part of each agent's internal behavior.

For example, Figure 7 illustrates a sample corner-kick set-play. The set-play designates five set-play roles, each with a specific location, which should be filled before the ball is put back into play. Based on the home positions of the current formation, each individual agent can determine the best mapping from positions to set-play locations, i.e. the mapping that requires the least total displacement of the 5 players. If there is no player filling one of the necessary formation roles, then there must be two players filling the same role, one of which must move to the vacant role. In the event that no agent chooses to do so, the set-play can proceed with any single set-play role unfilled. The only exception is that some player must fill the set-play role responsible for kicking the ball back into play. A special-purpose protocol is incorporated into the set-play behaviors to guarantee such a condition.

Once the set-play roles are filled, each player executes the action associated with its set-play role as an external behavior. As illustrated by the player starting the corner-kick in Figure 7, a player could choose among possible actions, perhaps based on the opponent positions at the time of execution. No individual player is guaranteed of participating in the play. For example, the uppermost set-play position is there just in case one of the other players misses a pass or shoots wide of the goal: no player will pass directly to it. Each player leaves its set-play role to resume its former role either after successfully kicking the ball, or after a pre-specified, role-specific amount of time.

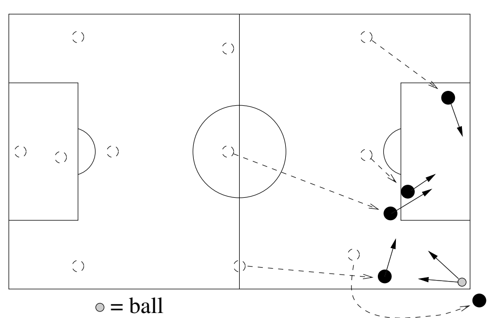

Figure 7: A sample corner-kick set-play. The dashed circles show the positions in the team's current formation and dashed arrows indicate the locations of the set-play roles—black circles—that they would fill. Solid arrows indicate the direction the ball is to be kicked as part of each set-play role.

The set-plays significantly improved CMUnited's performance. During the RoboCup-97 simulator competition, several goals were scored as a direct result of set-plays.

6.2 Communication Paradigm Implementation

In our teamwork structure, players are organized into team formations with each player filling a unique role. However players can switch among roles and the entire team can change formations. Both formations and roles are defined as part of the locker-room agreement, and each player is given a unique ID number. It is a significant challenge for players to remain coordinated, both by all believing that they are using the same formation and by filling all the roles in the formation. Since agents are all completely autonomous, such coordination is not guaranteed.

In PTS domains with limited communication (as opposed to no communication) possible during the dynamic, real-time periods, inter-agent communication can help the team stay coordinated. Communication protocols defined in the locker-room agreement combine with heard messages to trigger internal behaviors that alter the agent's internal state.

This section describes the CMUnited-97 simulator team implementation of the communication paradigm presented in Section 5. All of the agent messages are of the format:

> (CMUnited <Uniform-number> <Encoded-stamp> <Formation-number> <Formation-set-time> <Position-number> <target> <Message-type> [<Message-data>])

For example, player 8 might want to pass to player 6 but not know precisely where player 6 is at the moment. In this case, it could send the message `(CMUnited 8 312 1 0 7 ----> 6 Where are you?)`. "`CMUnited 8`" is the sender's team and number; "`312`" is the <Encoded-stamp>, in this case an agreed-upon linear combination of the current time, the formation number, and the sender's position number; "`1 0`" is the team formation player 8 is using followed by the time at which it started using it; "`7`" is player 8's current position; "`----> 6`" indicates that the message is for player 6; and "`Where are you?`" is a message type indicating that a particular response is requested: the recipient's coordinate location. In this case, there is no message data.

All teammates that hear such a message update their internal states to indicate that player 8 is playing position 7. However only player 6 sets its `response` and `response-flag` internal state variables. In this case, since the target is a single player, the `communicate-delay` flag remains at 0. Were the message targeted towards the whole team or to a subset of the team, then `communicate-delay` would equal:

- IF (my number $>$ sender number)
  ((my number $-$ sender number $-$ 1)*2)*`communicate-interval`
- ELSE (((sender number $-$ my number $-$ 1)*2)+1)*`communicate-interval`

where `communicate-interval` is the time between audible messages for a given agent (200 msec in the default soccer server). Thus, assuming no further interference, player 8 would be able to hear responses from all targets.

Once player 6 is ready to respond, it might send back the message `(CMUnited 6 342 1 0 5 ----> all I'm at 4.1 -24.5)`. Notice that player 6 is using the same team formation but playing a different position from player 8: position 5. Since this message doesn't require a response (as indicated by the "`I'm at`" message type), the message is accessible to the whole team ("`----> all`"): all teammates who hear the message update their world states to reflect the message data. In this case, player 6 is at coordinate position $(4.1, -24.5)$.

Were player 8 not to receive a response from player 6 before passing, it could still pass to its best estimate of player 6's location: should the message fail to get through, no disaster would result. Such is the nature of most communication in this domain. Should there be a situation which absolutely requires that a message get through, the sending agent could repeat the message periodically until hearing confirmation from the recipient that the message has arrived. However, such a technique consumes the single communication channel and should be used sparingly.

Notice that in the two example messages above, both players are using the same team-formation. However, such is not always the case. Even if they use common environmental cues to trigger formation changes, one player might miss the cue. In order to combat such a case, players update the team formation if a teammate is using a different formation that was set a later time as detailed in Section 5. For example, if player 6's message had begun "`(CMUnited 6 342 3 50 ...`" indicating that it had been using team formation 3 since time 50, an internal behavior in player 8 would have changed its internal state to indicate the new team strategy.

Other examples of message types used in our implementation of simulated robotic soccer players include:

- Request/respond ball location
- Request/respond teammate location
- Inform pass destination
- Inform going to the ball
- Inform taking/leaving position

7 Results

Although the flexible teamwork and communication paradigm implementations are merged into a single robotic soccer system, we are able to isolate the effects of each contribution through controlled testing. This section presents empirical results demonstrating the effectiveness of both main contributions of this article: the flexible teamwork structure and the low-bandwidth communication paradigm. In addition, the results of the CMUnited-97 simulator team at the RoboCup-97 competition are reported[3].

7.1 Teamwork Structure Results

The flexible teamwork structure improves over a rigid structure by way of three characteristics: flexible positioning within roles, set-plays, and changeable formations. We tested the benefits of the first two characteristics by playing a team with flexible, changeable positions and set-plays against a "default team" with rigid positions and no set-plays. The behaviors of the players on the two teams are otherwise identical. The advantage of being able to change formations—the third characteristic—depends on the formation being used by the opponent. Therefore, we tested teams using each defined formation against each other.

Standard games in the soccer server system last 10 minutes. However, due to the large amount of noise, game results vary greatly. All reported results are cumulative over several games. Compiled statistics include the number of 10-minute games won, the total cumulative goals scored by each team, average goals per game, and the percentage of time that the ball was in each half of the field. The last statistic gives a rough estimate of the degree to which each team was able to control the ball.

[3] The teamwork structure and communication paradigm were subsequently implemented in the CMUnited-98 simulator team as well. The CMUnited-98 simulator team competed in the RoboCup-98 competition, winning all 8 of its games by a combined score of 66–0 to become the 1998 world champion simulator team [38, 31].

7.1.1 Flexible Positions and Set-Plays

In order to test the flexible teamwork structure, we ran a team using ball-dependent flexible positions with set-plays against one using rigid positions and no set-plays. Both teams used a 4-4-2 formation. As shown in Table 3, the flexible team significantly outperformed the default team over the course of 38 games.

(Game = 10 min.)	Flexible and Set-Plays	Default
Games won	34	1
Total goals	223	82
Avg. goals	5.87	2.16
Ball in own half	43.8%	56.2%

Table 3: Results when a flexible team plays against a rigid team. The flexible team won 34 out of 38 games with 3 ties.

Further experimentation showed that both aspects of the flexible team contribute significantly to the team's success. Table 4 shows the results when a team using flexible positions but no set-plays plays against the default team and when a team using set-plays but rigid positions plays against the default team, again over the course of 38 games. Both characteristics provide a significant advantage over the default team, but they perform even better in combination.

Only Flexible Positions

(Game = 10 min.)	Flexible	Default
Games won	26	6
Total goals	157	87
Avg. goals	4.13	2.29
Ball in own half	44.1%	55.9%

Only Set-Plays

(Game = 10 min.)	Set-Plays	Default
Games won	28	5
Total goals	187	108
Avg. goals	4.92	2.84
Ball in own half	47.6%	52.4%

Table 4: Results when only using flexible positions and only using set-plays. Each individually works better than using neither.

7.1.2 Formations

In addition to the above tests, we tested the various formations against each other, as reported in Table 5. Each entry shows the goals scored for and against when a team using one formation played against a team using another formation over the course of 24 10-minute games. The right-most column collects the total goals scored for and against the team using that formation when playing against all the other teams. In all cases, the teams used flexible positions, but no set-plays.

formations	4-3-3	4-4-2	3-5-2	8-2-0	3-3-4	2-4-4	totals
4-3-3		68–60	68–54	24–28	59–64	70–65	289–271 (51.6%)
4-4-2	60–68		68–46	22–24	51–57	81–50	282–245 (53.5%)
3-5-2	54–68	46–68		13–32	61–72	75–73	249–313 (44.3%)
8-2-0	28–24	24–22	32–13		27–28	45–36	156–96 (61.9%)
3-3-4	64–59	57–51	72–61	28–27		87–69	308–267 (53.6%)
2-4-4	65–70	50–81	73–75	36–45	69–87		293–385 (43.2%)

Table 5: Comparison of the different formations. Entries in the table show the number of goals scored. Total (and percentage) cumulative goals scored against all formations appear in the right-most column.

The results show that the defensive formation (8-2-0) does the best. However the total goals scored when using the defensive formation is quite low. On the other hand, the 3-3-4 formation performs well with a high goal total.

This study allowed us to devise an effective formation-switching strategy for RoboCup-97. Our team [34] used a 4-4-2 formation in general, switching to a 8-2-0 formation if winning near the end of the game, or a 3-3-4 formation if losing. This strategy, along with the flexible teamwork structure as a whole, and the novel communication paradigm, helped us to perform well in the tournament, making it to the semi-finals in a field of 29 teams and out-scoring opponents by a total score of 67–9 [16].

We used this flexible teamwork structure on our real robot team [42] which won the RoboCup small-size robot competition. Although developed in simulation, all of the teamwork concepts apply directly to real robot teams as well. We were able to reuse the developed approach directly from our simulator clients on the robots and immediately achieve variable formations, flexible positions, and position switching.

7.2 Communication Paradigm Results

While contributing to the overall success of the CMUnited-97 simulator team, our novel communication paradigm is also demonstrably effective in controlled experimentation. In this section, we report results reflecting the agents' robustness to active interference, their ability to handle messages that require responses from multiple team members, and their ability to maintain a coordinated team strategy.

7.2.1 Robustness to Interference

Relying on communication protocols involves the danger that an opponent could actively interfere by mimicking an agent's obsolete messages: since there is a single communication channel, opponents can hear and mimic messages intended for teammates. However, the <Encoded-stamp> field guards against such an attempt. As a test, we played a communicating team (team C) against a team that periodically repeats past opponent messages (team D). Team C set the <Encoded-stamp> field to <Uniform-number> $*$(`send-time` + 37). Thus teammates could determine `send-time` by inverting the same calculation (known to all through the locker-room agreement). Messages received more than a second after the `send-time` were disregarded. The one-second leeway accounts for the fact that teammates may have slightly different notions of the current global time.

In our experiment, agents from team D sent a total of 73 false messages over the course of a 5-minute game. Not knowing team C's locker-room agreement, they were unable to adjust the <Encoded-stamp> field appropriately. The number of team C agents hearing a false message ranged from 0 to 11, averaging 3.6. In all cases, each of the team C agents hearing the false message correctly ignored it. Only one message truly from a team C player was incorrectly ignored by team C players, due to the sending agent's internal clock temporarily diverging from the correct value by more than a second. Although it didn't happen in the experiment, it is also theoretically possible that an agent from team D could mimic a message within a second of the time that it was originally sent, thus causing it to be indistinguishable from valid messages. However, in this case, the content of the message is presumably still appropriate and consequently unlikely to confuse team C.

7.2.2 Handling Multiple Responses

Next we tested our method of handling multiple simultaneous responses to a single message. Placing all 11 agents within hearing range, a single agent periodically sent a "where are you" message to the entire team and recorded the responses it received. In all cases, all 10 teammates heard the original message and responded. However, as shown in Table 6, the use of our method dramatically increased the number of responses that got through to the sending agent. When the team used `communicate-delay` as specified in Section 6, message responses were staggered over the course of about 2.5 seconds, allowing most of the 10 responses to get through. When all agents responded at once (no delay), only one response (from a random teammate) was heard.

	Number of Responses			Response Time (sec)		
	Min	Max	Avg	Min	Max	Avg
No Delay	1	1	1.0	0.0	0.0	0.0
Delay	6	9	8.1	0.0	2.6	0.9

Table 6: The number of responses that get through to agents when responses are delayed and when they are not. When the team uses `communicate-delay` as specified in Section 6, an average of 7.1 more responses get through than when not using it. Average response time remains under one second. Both experiments were performed 50 times.

7.2.3 Team Coordination

Finally, we tested the team's ability to maintain coordinated team strategies when changing formations via communication. One player was given the power to toggle the team's formation between a defensive and an offensive formation. Announcing the change only once, the rest of team had to either react to the original message, or get the news from another teammate via other communications. As described in Section 6, the <Formation-number> and <Formation-set-time> fields are used for this purpose. We ran two different experiments, each consisting of 50 formation changes. In the first, a midfielder made the changes, thus making it possible for most teammates to hear the original message. In the second experiment, fewer players heard the original message since it was sent by the goaltender from the far end of the field. Even so, the team was able to change formations in an average time of 3.4 seconds. Results are summarized in Table 7.

	Entire Team Change Time (sec)				Heard From
Decision-Maker	Min	Max	Avg	Var	Decision-Maker
Goaltender	0.0	23.8	3.4	17.8	46.6%
Midfielder	0.0	7.9	1.3	2.8	80.6%

Table 7: The time it takes for the entire team to change team strategies when a single agent makes the decision. Even when the decision-making agent is at the edge of the field (goaltender) so that fewer than half of teammates can hear the single message indicating the switch, the team is completely coordinated after an average of 3.4 seconds.

7.3 RoboCup-97

The RoboCup-97 simulator competition was the first formal simulated robotic soccer competition. With 29 teams entering from all around the world, it was a very successful tournament.

It was in preparation for this competition that the team member agent architecture, including both the flexible teamwork structure and the inter-agent communication paradigm, was developed. Since competitions are not controlled experiments, their results are not presented as scientific validation of our individual techniques. Such validation is presented in Section 7.1 and 7.2. However, qualitative evaluations and anecdotes from these competitions can provide insights into the strengths and weaknesses of various approaches.

Table 8 shows the results of CMUnited-97's games in this tournament. CMUnited-97 won 3 of its first 4 matches by wide margins, with the other match being a close victory. Its 5th opponent, FCMellon, was also our own team and was identical to CMUnited except that it did not use a flexible teamwork structure: players did not switch positions, did not use flexible positioning of any sort, and did not use set-plays. Our goal in entering FCMellon in the competition was to highlight the impact of our research contributions in CMUnited. Due to the results reported in Section 7.1, we expected CMUnited to win this game. Before the game between CMUnited and FCMellon, FCMellon won its 4 games by a combined score of 49–4.

Opponent	Affiliation	Score (CMU–Opp.)
LAI	Universidad Carlos III De Madrid, Spain	9 – 1
RM Knights	Royal Melbourne Inst. of Tech., Australia	16 – 0
Kinki	Kinki University, Japan	6 – 5
Team Garbage Collectors	Justsystem, Japan	24 – 0
FCMellon	Carnegie Mellon University, USA	6 – 0
AT-Humboldt	Humboldt University of Berlin, Germany	0 – 6
ISIS	Information Sciences Institute (USC), USA	1 – 2*
TOTAL		62 – 14

Table 8: The scores of CMUnited-97's games in the simulator league of RoboCup-97. CMUnited-97 won 5 of its 7 games, finishing in 4th place out of 29 teams. *The last game was lost by one goal in overtime.

The subsequent game was against the eventual tournament champion AT-Humboldt [5]. As described in Section 7.1, CMUnited-97 used a 4-4-2 formation in general, switching to an 8-2-0 formation if winning near the end of the game, or a 3-3-4 formation if losing. The triggers for these formation switches were defined as part of the locker-room agreement. However, by the time CMUnited-97 played against them, it was clear from watching other games that AT-Humboldt was the team to beat. Therefore, we altered the team's locker-room agreement so that it would adopt a more conservative, defensive strategy at the beginning of the game. As a result, AT-Humboldt scored fewer goals against CMUnited-97 than it did against any of its other competitors. The 6–0 result might have been even closer had CMUnited-97 not switched to the more offensive 3-3-4 formation near the end of the game when it was losing in an attempt, though unsuccessful, to score some goals.

One of the main advantages of the AT-Humboldt team was its ability to kick the ball harder than any other team. Its players did so by kicking the ball around themselves, continually increasing its velocity so that it ended up moving towards the goal faster than was imagined possible. Since the soccer server did not enforce a maximum ball speed, a property that was changed immediately after the competition, the ball could move arbitrarily fast, making it impossible to stop. With this advantage at the low-level behavior level, no team, regardless of how strategically sophisticated, was able to defeat AT-Humboldt.

Having lost in the semi-finals, CMUnited-97 then played in the 3rd-place game against ISIS [41]. CMUnited-97 scored first in this game off of a corner-kick set-play. However, ISIS equalized near the end of the game and the game went to overtime. ISIS scored to win in what proved to be one of the more exciting matches of the tournament.

8 Discussion and Related Work

This article has presented a team member agent architecture appropriate for PTS domains. While the implementation described here is in robotic soccer, it is easy to see how the architecture would apply to other sports, such as American football. In that case, the synchronization opportunities are more frequent, but formations can change during the course of a game, roles are defined with some flexibility so that agents can adjust to opponent behaviors on the fly, and agents must communicate efficiently both between plays on a drive and during plays.

There are several other examples of non-sports-related PTS domains. Having successfully developed and deployed an autonomous spacecraft [27], NASA is now interested in multi-spacecraft missions, or constellations [30]. Since spacecraft pointing constraints limit the communication both between the spacecraft and ground control and among the spacecraft, the spacecraft must be able to act autonomously while still working towards the constellations overall goal. Using interferometry missions—in which several spacecraft coordinate parts of a powerful imaging instrument to view distant objects—as an example, the locker-room agreement could be used to define several formations to be used for viewing objects that are at various distances or in different parts of the sky. Depending on the relative locations of these objects, the various spacecraft might interchange roles as they image different objects.

Search and rescue scenarios could also be formulated as PTS domains. If several robotic agents are trying to locate victims in a remote disaster sight, they may have to act quickly and autonomously. Nonetheless, before beginning the search, they could define several formations corresponding to different geographical areas of focusing their search. Within these formations, agents would need to be assigned flexible roles given that the precise situation may not be known or may change unexpectedly. The agents might also agree, as part of their locker-room agreement to switch formations either after a certain time or as a result of some limited communication, perhaps from a unit captain.

Other PTS domains that could be applications for the team member agent architecture are hospital/factory maintenance [9] and battlefield combat [40]. Network packet routing [4] could also be formulated as a PTS domain if the network nodes are permitted to freely use network bandwidth during periods of otherwise low usage. They could then exchange policies and feedback with regards to network performance.

The remainder of this section summarizes the previous work most closely related to the teamwork structure and communication paradigm as presented in this article and gives an overview of previous research in the robotic soccer domain.

8.1 Teamwork Structure

Two popular multi-agent teamwork structures, joint intentions [7] and shared plans [13], consider a team to be a group of agents that negotiate and/or contract with each other in order to initiate a team plan. The team forms dynamically and stays in close communication until the execution of the plan is completed. In contrast, the teamwork structure presented in this article supports a persistent team effort towards a common high-level goal in the face of limited communication.

The concept of the locker-room agreement facilitates coordination with little or no communication. Although it has been claimed that pre-determined team actions are not flexible or robust to failure [40], the locker-room agreement provides a mechanism for pre-defining team actions with enough flexibility to succeed. In particular, set-plays are pre-determined team actions that can be executed without the need to negotiate or use extensive inter-agent communication: the locker-room agreement provides enough flexibility that the agents are able to seamlessly assume the appropriate roles.

While we use the term "formation" to refer to the largest unit of the teamwork structure, soccer formations are not to be confused with military-type formations in which agents must stay in precise relative positions. Despite this dual usage of the term, we use it because formation is a standard term within the soccer domain [18]. For an example of a multi-agent system designed for military formations, see [3].

Castelfranchi classifies different types of commitments in multi-agent environments [6]. In this context, locker-room agreements can be viewed as C-commitments, or commitments by team members to do the appropriate thing at the right time, as opposed to S-commitments with which agents adopt each other's goals. In the context of [8], the creation of a locker-room agreement is norm acceptance while its use is norm compliance. Within the framework presented in [24], the architecture is for interactive software and hardware multi-agents.

As mentioned in Section 3, the concept of behavior in the context of our team member agent architecture is consistent with that laid out by Mataric [22]. There, "behavior" is defined as "a control law with a particular goal, such as wall-following or collision avoidance." Behaviors can be nested at different levels with selection among lower-level behaviors consisting of a higher-level behavior. Similarly, internal and external behaviors in our system maintain team coordination goals, physical positioning goals, communication goals, and environmental information goals (such as knowledge of where the ball is). These behaviors are combined into top-level internal and external behaviors.

8.2 Communication Paradigm

Most inter-agent communication models assume reliable point-to-point messages passing with negligible communication costs. In particular, KQML assumes point-to-point message passing, possibly with the aid of facilitator agents [10]. Nonetheless, KQML performatives could be used for the content portions of our communication scheme. KQML does not address the problems raised by having a single, low-bandwidth communication channel.

When communication is reliable and the cost of communication relative to other actions is small, agents have the luxury of using reliable, multi-step negotiation protocols. For example, in Cohen's convoy example [7], the communication time required to form and maintain a convoy of vehicles is insignificant compared to the time it takes the convoy to drive to its destination. Similarly, message passing among distributed information agents is typically very quick compared to the searches and services that they are performing. Thus, it makes sense for agents to initiate and confirm their coalition while guaranteeing that they will inform each other if they have trouble fulfilling their part of the joint action.

With only a single team present, a situation similar to the one considered here is examined in [21]. In that case, like in the soccer server, messages sent are only heard by agents within a circular region of the sender. Communication is used for cooperation and for knowledge sharing. Like in the examples presented in the soccer domain, agents attempt to update each other on their own internal states when communicating. However, the exploration task considered there is much simpler than soccer, particularly in that there are no opponents using the same communication channel and in that the nature of the task allows for simpler, less urgent communication.

Although communication in the presence of hostile agents is well studied in military contexts from the standpoint of encryption, the problem considered here is not the same. While any encryption scheme could be used for the message content, the work presented here assumes that the adversaries cannot decode the message. Even so, they can disrupt communication by mimicking past messages textually: presumably past message have some meaning to the team that uttered them. Our method of message coding based on a globally accessible clock circumvents this latter problem.

Even when communication time is insignificant compared to action execution, such as in a helicopter fighting domain, it can be risky for agents to rely on communication. As pointed out in [39], if the teammate with whom an agent is communicating gets shot down, the agent could be incapacitated if it requires a response from the teammate. This work also considers the cost of communication in terms of risking opponent eavesdropping and the benefits of communication in terms of shifting roles among team members. However, the problems raised by a single communication channel and the possibility of active interference are not considered, nor are the challenges raised when communication conflicts with real-time action.

Another approach that recognizes the danger of basing behaviors upon multi-step communication protocols is ALLIANCE [26]. Since a primary goal of this work is fault-tolerance, only broadcast communications are used. Agents inform each other of what they are currently doing, but never ask for responses. In ALLIANCE, the team uses time-slice communication so that each agent periodically gets exclusive use of the single communication channel.

A possible application of our communication method is to robots using audio communication. This type of communication is inherently single-channel and low-bandwidth. An example of such a system is the Robot Entertainment Systems which uses a tonal language [11]. Agents can communicate by emitting and recognizing a range of audible pitches. In such a system, the number of bits per message would have to be lowered, but the general techniques presented in this article still apply.

Another example of such a communication environment is arbitrarily expandable systems. If agents aren't aware of what other agents exist in the environment, then all agents must use a single universally-known communication channel, at least in order to initiate communication.

8.3 Robotic Soccer

Many other robotic soccer systems have been developed both in simulation and with real robots. Using a simulator based closely upon the Dynasim system [28], we previously used Memory-based Learning to allow a player to learn when to shoot and when to pass the ball [32]. We then used Neural Networks to teach a player to shoot a moving ball into the goal [36]. In the soccer server, we then layered two learned behaviors to produce a higher-level multi-agent behavior: passing [35]. Also in the soccer server Matsubara et al. used a Neural Network to allow a player to learn when to shoot and when to pass [23] (as opposed to the Memory-based technique used by us for a similar task). The RoboCup-97 simulator competition included 29 teams, many of which demonstrated novel scientific contributions, particularly in the field of multi-agent learning [16].

Robotic soccer with real robots was pioneered by the Dynamo group [29] and Asada's laboratory [2]. Recent international competitions have motivated the creation of a wide variety of robot soccer teams [14, 16].

Most previous research, both in simulation and on real robots has concentrated on individual skills with little attention paid to team coordination. A rare exception is [19] in which team coordination is evolved using genetic programming. Unfortunately, no general teamwork structure can be extracted from this work as it is evolved in a domain specific setting.

9 Conclusion

In this article, we introduced a flexible teamwork structure for periodic team synchronization (PTS) domains and a communication paradigm effective in domains with low-bandwidth, single-channel, unreliable communication.

The teamwork structure allows for multi-agent tasks using homogeneous agents to be decomposed into flexible roles. Roles are organized into formations, and agents can fill any role in any formation. Agents dynamically change roles and formations in response to changing environments. The teamwork structure includes pre-planning for frequent situations, and agents act individually, but keep the team's goals in mind. This flexible teamwork structure builds upon our team member agent architecture, which maintains both an internal and world state, and a set of internal and external behaviors. Coordination is achieved through limited communication and pre-determined procedures as part of a locker-room agreement.

In domains with low-bandwidth, single-channel, unreliable communication, several issues arise that need not be considered in most multi-agent domains. We have identified these issues and presented a communication paradigm which successfully addresses these challenges. Like the teamwork structure, the communication paradigm is defined within the locker-room agreement.

Both the teamwork structure and the communication paradigm are situated within a team member agent architecture. The locker-room agreement is a central component of this architecture.

We presented a full implementation of our innovations in the simulated robotic soccer domain. Our flexible teamwork structure approach was developed in simulation and subsequently also successfully used on real robots. Using this teamwork structure as well as the presented communication paradigm, the CMUnited-97 simulator team made it to the semi-finals of RoboCup-97. The real robot team, using the flexible teamwork structure, won the small-size, real-robot, RoboCup-97 world championship.

References

[1] David Andre, Emiel Corten, Klaus Dorer, Pascal Gugenberger, Marius Joldos, Johan Kummeneje, Paul Arthur Navratil, Itsuki Noda, Patrick Riley, Peter Stone, Romoichi Takahashi, and Travlex Yeap. Soccer server manual, version 4.0. Technical Report RoboCup-1998-001, RoboCup, 1998. At URL http://ci.etl.go.jp/~noda/soccer/server/Documents.html.

[2] Minoru Asada, Eiji Uchibe, Shoichi Noda, Sukoya Tawaratsumida, and Koh Hosoda. Coordination of multiple behaviors acquired by vision-based reinforcement learning. In *Proc. of IEEE/RSJ/GI International Conference on Intelligent Robots and Systems 1994*, pages 917–924, 1994.

[3] Tucker Balch and Ronald C. Arkin. Motor schema-based formation control for multiagent robot teams. In *Proceedings of the First International Conference on Multi-Agent Systems*, pages 10–16, Menlo Park, California, June 1995. AAAI Press.

[4] J. A. Boyan and M. L. Littman. Packet routing in dynamically changing networks: A reinforcement learning approach. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances In Neural Information Processing Systems 6*. Morgan Kaufmann Publishers, 1994.

[5] Hans-Dieter Burkhard, Markus Hannebauer, and Jan Wendler. AT Humboldt — development, practice and theory. In Hiroaki Kitano, editor, *RoboCup-97: Robot Soccer World Cup I*, pages 357–372. Springer Verlag, Berlin, 1998.

[6] Cristiano Castelfranchi. Commitments: From individual intentions to groups and organizations. In *Proceedings of the First International Conference on Multi-Agent Systems*, pages 41–48, Menlo Park, California, June 1995. AAAI Press.

[7] Philip R. Cohen, Hector J. Levesque, and Ira Smith. On team formation. In J. Hintikka and R. Tuomela, editors, *Contemporary Action Theory*. Synthese, 1999. to appear.

[8] R. Conte, C. Castelfranchi, and F. Dignum. Autonomous norm-acceptance. In J. P. Müller, M. P. Singh, and A. S. Rao, editors, *Intelligent Agents V — Proceedings of the Fifth International Workshop on Agent Theories, Architectures, and Languages (ATAL-98)*, volume 1555 of *Lecture Notes in Artificial Intelligence*, pages 99–112. Springer-Verlag, Heidelberg, 1999.

[9] Keith S. Decker. Task environment centered simulation. In M. Prietula, K. Carley, and L. Gasser, editors, *Simulating Organizations: Computational Models of Institutions and Groups*. AAAI Press/MIT Press, 1996.

[10] Tim Finin, Don McKay, Rich Fritzson, and Robin McEntire. KQML: An information and knowledge exchange protocol. In Kazuhiro Fuchi and Toshio Yokoi, editors, *Knowledge Building and Knowledge Sharing*. Ohmsha and IOS Press, 1994.

[11] Masahiro Fujita and Koji Kageyama. An open architecture for robot entertainment. In *Proceedings of the First International Conference on Autonomous Agents*, pages 435–442, Marina del Rey, CA, February 1997.

[12] M. R. Genesereth and R. E. Fikes. Knowledge interchange format, version 3.0 reference manual. Technical Report Logic-92-1, Computer Science Department, Stanford University, 1992.

[13] Barbara J. Grosz. Collaborative systems. *AI Magazine*, 17(2):67–85, Summer 1996.

[14] Jong-Hwan Kim, editor. *Proceedings of the Micro-Robot World Cup Soccer Tournament*, Taejon, Korea, November 1996.

[15] Hiroaki Kitano, editor. *Proceedings of the IROS-96 Workshop on RoboCup*, Osaka, Japan, November 1996.

[16] Hiroaki Kitano, editor. *RoboCup-97: Robot Soccer World Cup I*. lecture notes in AI. Springer Verlag, Berlin, 1998.

[17] Hiroaki Kitano, Yasuo Kuniyoshi, Itsuki Noda, Minoru Asada, Hitoshi Matsubara, and Eiichi Osawa. RoboCup: A challenge problem for AI. *AI Magazine*, 18(1):73–85, Spring 1997.

[18] Michael L. LaBlanc and Richard Henshaw. *The World Encyclopedia of Soccer*. Visible Ink Press, 1994.

[19] Sean Luke, Charles Hohn, Jonathan Farris, Gary Jackson, and James Hendler. Co-evolving soccer softbot team coordination with genetic programming. In Hiroaki Kitano, editor, *RoboCup-97: Robot Soccer World Cup I*, pages 398–411, Berlin, 1998. Springer Verlag.

[20] A. K. Mackworth. On seeing robots. In A. Basu and X. Li, editors, *Computer Vision: Systems, Theory, and Applications*, pages 1–13. World Scientific Press, Singapore, 1993.

[21] Dario Maio and Stefano Rizzi. Unsupervised multi-agent exploration of structured environments. In *Proceedings of the First International Conference on Multi-Agent Systems*, pages 269–275, Menlo Park, California, June 1995. AAAI Press.

[22] Maja J. Mataric. Interaction and intelligent behavior. MIT EECS PhD Thesis AITR–1495, MIT AI Lab, August 1994.

[23] Hitoshi Matsubara, Itsuki Noda, and Kazuo Hiraki. Learning of cooperative actions in multi-agent systems: a case study of pass play in soccer. In *Adaptation, Coevolution and Learning in Multiagent Systems: Papers from the 1996 AAAI Spring Symposium*, pages 63–67, Menlo Park,CA, March 1996. AAAI Press. AAAI Technical Report SS-96-01.

[24] J. P. Müller. The right agent (architecture) to do the right thing. In J. P. Müller, M. P. Singh, and A. S. Rao, editors, *Intelligent Agents V — Proceedings of the Fifth International Workshop on Agent Theories, Architectures, and Languages (ATAL-98)*, volume 1555 of *Lecture Notes in Artificial Intelligence*, pages 211–225. Springer-Verlag, Heidelberg, 1999.

[25] Itsuki Noda and Hitoshi Matsubara. Soccer server and researches on multi-agent systems. In *Proceedings of the IROS-96 Workshop on RoboCup*, November 1996. Soccer server is available at URL http://ci.etl.go.jp/˜noda/soccer/server/index.html.

[26] Lynne E. Parker. *Heterogeneous Multi-Robot Cooperation*. PhD thesis, Massachusetts Institute of Technology, 1994.

[27] Barney Pell, Douglas E. Bernard, Steven A. Chien, Erann Gat, Nicola Muscettola, P. Pandurang Nayak, Michael D. Wagner, and Brian C. Williams. An autonomous spacecraft agent prototype. *Autonomous Robots*, 5(1), March 1998.

[28] Michael K. Sahota. Dynasim user guide. At URL http://www.cs.ubc.ca/nest/lci/soccer, January 1996.

[29] Michael K. Sahota, Alan K. Mackworth, Rod A. Barman, and Stewart J. Kingdon. Real-time control of soccer-playing robots using off-board vision: the Dynamite testbed. In *IEEE International Conference on Systems, Man, and Cybernetics*, pages 3690–3663, 1995.

[30] Peter Stone. Multiagent learning for autonomous spacecraft constellations. In *Proceedings of the NASA Workshop on Planning and Scheduling for Space*, 1997.

[31] Peter Stone. *Layered Learning in Multi-Agent Systems*. PhD thesis, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, December 1998. Available as technical report CMU-CS-98-187.

[32] Peter Stone and Manuela Veloso. Beating a defender in robotic soccer: Memory-based learning of a continuous function. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 896–902, Cambridge, MA, 1996. MIT Press.

[33] Peter Stone and Manuela Veloso. Multiagent systems: A survey from a machine learning perspective. Technical Report CMU-CS-97-193, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, December 1997.

[34] Peter Stone and Manuela Veloso. The CMUnited-97 simulator team. In Hiroaki Kitano, editor, *RoboCup-97: Robot Soccer World Cup I*. Springer Verlag, Berlin, 1998.

[35] Peter Stone and Manuela Veloso. A layered approach to learning client behaviors in the RoboCup soccer server. *Applied Artificial Intelligence*, 12:165–188, 1998.

[36] Peter Stone and Manuela Veloso. Towards collaborative and adversarial learning: A case study in robotic soccer. *International Journal of Human-Computer Studies*, 48(1):83–104, January 1998.

[37] Peter Stone and Manuela Veloso. Using decision tree confidence factors for multiagent control. In Hiroaki Kitano, editor, *RoboCup-97: Robot Soccer World Cup I*, pages 99–111. Springer Verlag, Berlin, 1998. Also in *Proceedings of the Second International Conference on Autonomous Agents*, 1998.

[38] Peter Stone, Manuela Veloso, and Patrick Riley. The CMUnited-98 champion simulator team. In Minoru Asada and Hiroaki Kitano, editors, *RoboCup-98: Robot Soccer World Cup II*. Springer Verlag, Berlin, 1999.

[39] Milind Tambe. Teamwork in real-world, dynamic environments. In *Proceedings of the Second International Conference on Multi-Agent Systems*, Menlo Park, California, December 1996. AAAI Press.

[40] Milind Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7:81–124, 1997.

[41] Milind Tambe, Jafar Adibi, Yaser Al-Onaizan, Ali Erdem andGal A. Kaminka, Stacy C. Marsela, Ion Muslea, and Marcelo Tallis. Using an explicit model of teamwork in RoboCup-97. In Hiroaki Kitano, editor, *RoboCup-97: Robot Soccer World Cup I*, pages 123–131. Springer Verlag, Berlin, 1998.

[42] Manuela Veloso, Peter Stone, Kwun Han, and Sorin Achim. The CMUnited-97 small-robot team. In Hiroaki Kitano, editor, *RoboCup-97: Robot Soccer World Cup I*, pages 242–256. Springer Verlag, Berlin, 1998.