Communication in Domains with Unreliable, Single-Channel, Low-Bandwidth Communication[*]

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}

Submitted to ICMAS'98 in November 1997

Abstract

In most multiagent systems with communicating agents, the agents have the luxury of using reliable, multi-step negotiation protocols. They can do so primarily when communication is reliable and the cost of communication relative to other actions is small. Conversely, this paper considers multiagent environments with unreliable, high-cost communication. This paper presents techniques for dealing with the obstacles to inter-agent communication in such environments. A successful prototype system is fully implemented and tested in the simulated robotic soccer domain.

Topic Areas:
- Communication languages and protocols;
- Organization and social structure

[*]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.

1 Introduction

In most multiagent systems with communicating agents, the agents have the luxury of using reliable, multi-step negotiation protocols (see [1] for instance). They can do so primarily when communication is reliable and the cost of communication relative to other actions is small. For example, in Cohen's convoy example [2], 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.

Conversely, this paper considers multiagent environments with unreliable, high-cost communication. For example, if there is only a single, low-bandwidth, unreliable communication channel for all the agents, and if the agents must sacrifice valuable resources in order to communicate, then although inter-agent communication may be beneficial, the agents' behaviors must not *depend* upon it.

One clear example of such an environment is the Soccer Server—a widely used robotic soccer simulator—with a single, low-bandwidth, unreliable communication channel for all 22 agents and with high communication costs [8]. We use this domain for the research reported here. Another example domain is one that uses aural communication in crowded settings. Both people and robots using aural sensors ([4]) 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. A third example of such an 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.

This paper presents techniques for dealing with the obstacles to inter-agent communication in such environments, particularly those with several *teams* of agents.

2 Team Member Architecture

Our new communication paradigm is situated within a team member architecture suitable for multiagent domains in which team members must act autonomously while working towards a common team goal. The team can synchronize ahead of time but while executing the task, communication is limited. Based on a standard agent architecture, our team member 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 low communication.

An 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 processed 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 team structure as presented below as well as inter-agent communication protocols. The locker-room agreement may change periodically when the team is able to re-synchronize; however, it generally remains unchanged. 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.

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, sending commands 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. 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.

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)`.

Locker-room agreements can be used to eliminate or reduce the need for future communication, and they can also be used to increase communication reliability. For example,

Communication Environment	**Challenges**
• many agents, teams	• message targeting/distinguishing
• single-channel	• robustness to active interference
• low-bandwidth	• multiple simultaneous responses
• unreliable	• robustness to lost messages
• high cost	• team coordination

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

team members could agree upon a code number with which all messages should start in order to distinguish their messages from those of other teams in case other teams send similar messages on the single communication channel. They could also synchronize internal clocks if there is no globally accessible clock.

3 Communication Paradigm

The challenge for an agent to distinguish messages that are meant for it from those that are not is the first of five challenges that arise in the type of environment considered here. Second, 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. Third, since the communication channel has low bandwidth, the team must prevent itself from all "talking 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. Fourth, since communication is unreliable, agents must be robust to lost messages: their behaviors cannot depend upon receiving communications from a teammate. Fifth, teams must determine how to maximize the chances that they are using the same team strategy despite the facts that each is acting autonomously and that communication is unreliable.

In order to meet these challenges, we propose that a team should use messages of the following form:

> (<team-identifier> <unique-team-member-ID> <encoded-time-stamp> <time-stamped-team-strategy> <selected-internal-state> <target> <message-type> <message-data>)

Such a formulation assumes that the bandwidth allows for messages of several bytes in length to be transmitted in a reasonable amount of time. Some aural communication scenarios may need fewer, or condensed fields.

The contents of these fields are the product of the locker-room agreement. When forming the team, the agents must agree upon their team name (<team-identifier>) and a unique ID number for each member. For simplicity, the member IDs can be sequential numbers. These first two fields ensure that any teammate hearing the message knows precisely who uttered it. Teammates also agree ahead of time upon the security code used to create the field $<$encoded-time-stamp$>$. To coordinate, they agree upon a method for encoding and changing team strategies, and possibly upon positions of their internal states that should be communicated to help keep teammate information up to date. In addition, they must choose a set of acceptable message-types. The messages can use any syntactic and semantic codes (KQML [3] and KIF [5] for example). The only requirement is that the agents also agree on a mapping from message type to response requirements. Finally, the $<$target$>$ field can be used to indicate the intended recipient(s) of the message. It could be intended for a single team member, for some subset of them, or for all team members.

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

3.1 Message Targeting/Distinguishing

Agents can distinguish messages that are intended for them by checking the $<$team-identifier$>$ and $<$target$>$ fields. An agent A listens to a message from a member of the same team that is targeted to A, to the entire team, or to some subset of the team that includes A[1]. All other messages may be ignored, or since all team members know the locker-room agreement, agents may use these messages to monitor their teammates' internal states.

3.2 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 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 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 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 somehow discovers which field is the time stamp, 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

[1]The subsets could also be indicated by tokens if predetermined "units," or sub-formations, are formed.

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. Of course, 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. [2]

3.3 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:

	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 should update its internal beliefs of the other agent's status as indicated by the <time-stamped-team-strategy> field. However, only when the message is intended for it should it consider the content of the message. Then it should use the following algorithm in response to the message:

1. If the message requires no response (cases 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 (case b1), respond immediately: `communicate-delay` = 0;
   - if the message is sent to more than one target (cases 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.

Then, if an internal behavior keeps decrementing `communicate-delay` as time passes, an external behavior can use the communication condition-action pair presented in

[2] The degree of complexity necessary depends upon the number of messages that will be sent after the locker-room agreement. With few enough messages, a simple linear combination of the numerical message fields may suffice.

Section 2: `if (response-flag set and communicate-delay==0) then SAY(response)`. Players can 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.

3.4 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 should be structured so that it helps agents update their world and internal states. But agents should not stop acting while waiting for communications from teammates. As brought up in [10], such a case could cause infinite looping if a critical teammate fails to respond for any reason. In the same way that agents continue acting while waiting for `communicate-delay` to expire, agents must do their best to maintain accurate world and internal states without help from teammates and continue acting while waiting for responses from teammates.

3.5 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 should be no disastrous results should team members temporarily adopt different strategies; however they should always do their best to stay coordinated. One method of coordination is via the locker-room agreement. Agents could 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.

The <selected-internal-state> can also be used to facilitate team coordination by helping to keep the team members up-to-date regarding each other's status. Due to bandwidth constraints, it should not in general be an agent's entire internal state. However it might indicate the role that the agent is currently filling within the team strategy and any other particularly useful information as determined during the locker-room agreement.

3.6 Related Work

Most inter-agent communication models assume reliable point-to-point messages passing with negligeable communication costs. In particular, KQML assumes point-to-point message passing, possibly with the aid of facilitator agents [3]. Nonetheless, KQML performatives could be used for the content portions of our proposed communication scheme. KQML does not address the problems raised by having a single, low-bandwidth communication channel.

With only a single team present, a situation similar to the one considered here is examined in [7]. In that case, like in ours, 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.

Even when communication time is insignificant compared to action execution, such as in a helicopter fighting domain, it can be risky for agents to absolutely rely on communication. As pointed out in [10], 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.

A possible application of the method described here 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 [4]. 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 above still apply.

4 Implementation in the Robotic Soccer Domain

The soccer server [8] system used successfully at RoboCup'97 [6] during IJCAI'97 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 200ms (actions are possible every 100ms, 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. Furthermore, on every 100ms action cycle, agents can either communicate or move in the world. Since the real-time nature of the domain requires quick and timely reactions, and since opponents hear all messages, communicating involves a significant cost.

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

Table 2: Characteristics of the Soccer Server communication model.

4.1 Our Communication Approach in the Soccer Server

In our team 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. For more details on the issues relating to this team structure, see [9].

As proposed in Section 3, all of our agent messages are of the form:

> (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 <Endcoded-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.

Upon hearing such a message, any teammate would update its internal state 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 (200ms in this case). 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 $(4.1, -24.5)$.

Notice that 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 incurs high action costs 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. 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. Thus our team was able to remain coordinated even when changing formations.

Other examples of communication 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

We found that the resulting updates of player world states and internal states greatly improved the performance of our team.

4.2 Results

Detailed empirical testing indicates that the implementation detailed above is successful in the challenging communication environment of the Soccer Server. In this section, we report results reflecting the cost of communication, 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.

To test the cost of communication, we played a team using no communication (team A) against a team identical to the first in all regards except that its members say random strings periodically (team B). Thus team B gained no benefit from communication, but its action rate was reduced by the interleaving of random statements. With an average of 18% of its actions taken by these random communications, team B suffered a significant degradation in performance, losing to team A by an average score of 3.54 to 1.08 over 50 games. Clearly, communication in this domain involves a significant cost.

Relying on communication protocols also involves the danger that an opponent could actively interfere by mimicking an agent's obsolete messages. However, our <Endcoded-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 <Endcoded-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 <Endcoded-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.

Next we tested our proposed 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 3, the use of our proposed method dramatically increased the number of responses that got through to the sending agent. When the team used `communicate-delay` as specified in Section 4, 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.

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 4, 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

	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 3: When the team uses `communicate-delay` as specified in Section 4, 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.

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 4.

	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 4: The time it takes for the entire team to change team strategies even 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.

In addition to the above controlled experiments, we used our communication method in the CMUnited simulator team that competed in RoboCup'97. In a field of 29 teams, CMUnited made it to the semi-finals, indicating that the overall team construction, of which this communication model was a significant part, was successful.

5 Conclusion

In domains with low-bandwidth, single-channel, unreliable communication, several issues arise that need not be considered in most multiagent domains. We have presented a communication paradigm which successfully addresses these challenges. Having fully implemented it in the robotic soccer domain, we have tested the paradigm empirically both in a controlled setting and in competition against several previously unseen opponents. Using this paradigm, the CMUnited'97 simulator team made it to the semi-finals of RoboCup'97.

References

[1] Mihai Barbuceanu and Mark S. Fox. Cool: A language for describing coordination in multi

agent systems. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 17–24, Menlo Park, California, June 1995. AAAI Press.

[2] Philip R. Cohen, Hector J. Levesque, and Ira Smith. On team formation.

[3] 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.

[4] 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.

[5] 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.

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

[7] Dario Maio and Stefano Rizzi. Unsupervised multi-agent exploration of structured environments. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 269–275, Menlo Park, California, June 1995. AAAI Press.

[8] Itsuki Noda and Hitoshi Matsubara. Soccer server and researches on multi-agent systems. In *Proceedings of the IROS-96 Workshop on RoboCup*, November 1996.

[9] Peter Stone and Manuela Veloso. The cmunited'97 simulator team. In H. Kitano, editor, *RoboCup-97: The First Robot World Cup Soccer Games and Conferences*. Springer Verlag, Berlin, 1998, forthcoming.

[10] Milind Tambe. Teamwork in real-world, dynamic environments. In *Proceedings of the Second International Conference on Multi-Agent Systems (ICMAS-96)*, Menlo Park, California, December 1996. AAAI Press.