VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

AGENT MODELING FOR ROBOTIC SOCCER

SEMESTRÁLNÍ PROJEKT
TERM PROJECT

AUTOR PRÁCE
AUTHOR

Bc. JIŘÍ BOBEK

BRNO 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

MODELOVÁNÍ AGENTŮ PRO ROBOTICKÝ FOTBAL

AGENT MODELING FOR ROBOTIC SOCCER

SEMESTRÁLNÍ PROJEKT
TERM PROJECT

AUTOR PRÁCE — Bc. JIŘÍ BOBEK
AUTHOR

VEDOUCÍ PRÁCE — Ing. VLADIMÍR JANOUŠEK, Ph.D.
SUPERVISOR

BRNO 2008

Abstrakt

Tato práce se zabývá modelováním agentů pro simulovaný robotický fotbal. Nejprve je popsán robotický fotbal obecně, spolu s problémy, kterým musi čelit jednotliví agenti při praci v daném prostředí. Dále jsou zmíněny pravidla hry a stručně je popsáno prostředí soccer server, používané jako simulátor pro simulovanou fotbalovou ligu. Dále je navržena metoda pro řízení hráčů, využívající výcevrstvé učení a strojové učení.

Klíčová slova

robotický fotbal, DEVS model, softwarový agent, multi-agentní systém, strojové učení, vícevrstvé učení, umělá inteligence

Abstract

This work is focused on modeling of agents playing simulated soccer league. It portrays the robotic soccer itself, along with problems and challenges, any soccer team has to deal with. Further, there are mentioned the rules of the game, followed by brief description of the soccer server - a software used as a simulator for simulated soccer league. Then, there is proposed a method of controling the players. This method is based on machine learning and multilayer learning.

Keywords

robotic soccer, DEVS model, software agent, multi-agent system, machine learning, multilayer learning, artificial intelligence

Citace

Jiří Bobek: Agent Modeling For Robotic Soccer, semestrální projekt, Brno, FIT VUT v Brně, 2008

Agent Modeling For Robotic Soccer

Prohlášení

Prohlašuji, že jsem tento semestrální projekt vypracoval samostatně pod vedením pana Ing. Vladimíra Janouška, Ph.D.

Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

. . . .
Jiří Bobek
January 3, 2008

Contents

List of Figures

Chapter 1

Introduction

The robotic soccer is just good old-fashioned football game. But instead of human beings, the players are autonomous robots. And, in the case of simulated robotic soccer, even software agents.

The soccer game provides a standard problem where a wide range of technologies can be integrated and examined. The robot teams must operate in real-time, noisy, collaborative and adversarial environment, which brings us a lot of challenging problems. For example:

- sensing and acting is asynchronous
- player's vision is limited, giving him only a partial knowledge about the environment
- each player has a limited stamina
- there are many sources of noise
- everything happens in real time
- ...

And finally - the games are important opportunities for researchers to exchange technical information and they also serve as a great opportunity to educate and entertain the public.

1.1 Simulated Robotic Soccer

One of the advantages of the simulated league is the abstraction made, which relieves the researchers from having to care about robot problems (such as object recognition and communication), and hardware issues (how to actually make the robot move). The abstraction enables to focus on higher level concepts such as cooperation.

1.1.1 The rules of the game

Each team consists of 11 players. One of them is a goalkeeper. At a given time, the game is in one of several play-modes. These play-modes are: before_kick_off (at the onset of a half), play_on (normal play), time_over, kick_off (anounces start of play), kick_in, free_kick, corner_kick, goal_kick, drop_ball, and off_side. During a game, a number of rules are enforced either by the automated referee within the soccer server (see chapter 2), or by a human referee.

Rules judged by the automated referee:

- just before a kick_off (either before the game starts, or after a goal), all players must be in their own half of the field.
- when the ball goes out of the field, the referee moves the ball to a proper position and changes the play mode to kick_in, corner_kick, or goal_kick.
- when the play mode is kick_off, free_kick, kick_in or corner_kick, the referee removes all players from a circle centered on the ball. The radius of the circle is a paramater within the server (normally 9.15 meters). The removed players are placed on the perimeter of that circle.
- when the play mode is offside, all offending players are moved back to non-offside position.
- when the play mode is goal_kick, all offending players are moved outside the penalty area. And they can't reenter this area while the goal_kick mode takes place. The play mode is changed to play_on immediately after the ball leaves the penalty area.
- when the play mode is kick_off, free_kick, kick_in or corner_kick, the referee changes the mode to play_on immediately after the ball starts moving through a kick command.
- the referee suspends the game when the first or the second half finishes. The default length for each half is 3000 simulation cycles (about 5 minutes).
- if the match is drawn after the second half, the match is extended until a goal is scored.

Rules judged by the human referee:

- sorrounding the ball
- blocking the goal with too many players
- not putting the ball into game after given number of cycles
- intentionaly blocking the movement of other players

Chapter 2

Soccer Server

The RoboCup simulated league uses the RoboCup simulator called soccer server. The soccer server allows various simulated robotic teams to compete in game of soccer. Since the match is carried out in a client-server style, there are no restrictions on how the teams are build. The only requirement is that the players are connected to the server over protocol TCP/IP.

During the actual simulation, the time is updated in discrete steps. A simulation step is 100ms. During each simulation step, objects (players and the ball) stay on their positions. If players decide to act within a given simulation step, actions are applied to the players and the ball at the transition from one cycle to the next.

2.1 Sensors

All information an agent has about the environment is obtained from sensors. The soccer server offers aural, vision and body sensor.

2.1.1 Aural Sensor

The aural sensor is used to transmit and receive messages amongst players. Calls from the referee are also delivered as aural messages. All messages are delivered imediately.

A player can only hear a message if he has sufficien hear capacity. When a message is heard, the capacity is decreased, and every cycle, it is increased. With the current settings, a player can hear at most one message every 2nd simulation cycle.

2.1.2 Vision Sensor

The visual sensor reports every 150ms the objects currently seen by the player (other players, ball, goal, flags, lines, and so on - see figure 2.1). A random noise is added into all visual information.

If a player is seen, his team and number may be seen as well, depending on his distance.

The visual sensor does not see the "entire world", but has limited range of view, shown on figure 2.2.

2.1.3 Body Sensor

The body sensor reports every 100ms the physical status of the player. The information includes:

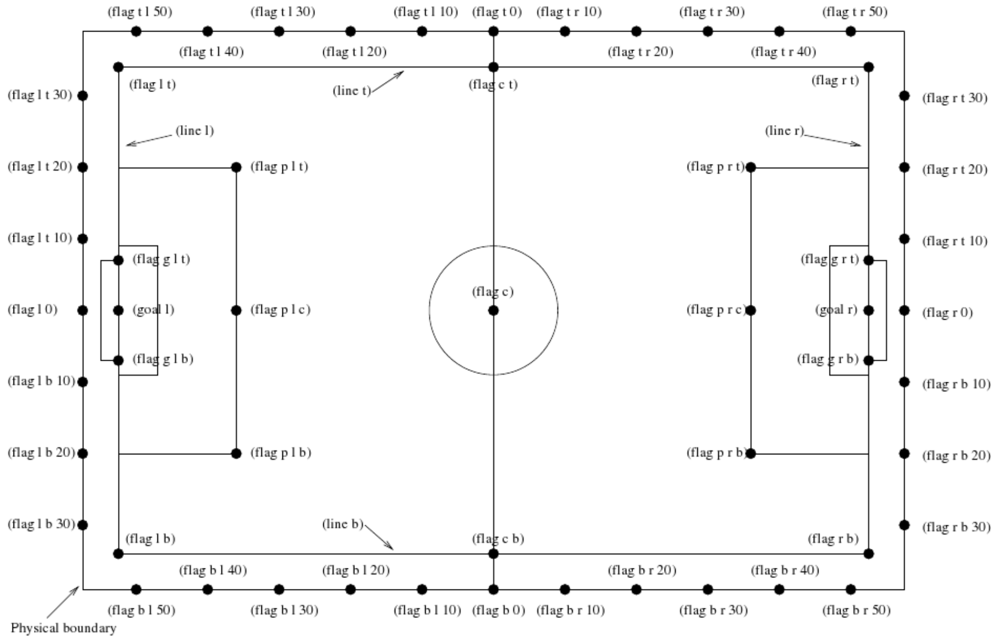

Figure 2.1: The flags and lines in the simulation. Image taken from [3].

- player speed and direction (with added noise)
- relative direction of the player's head
- stamina

2.2 Actions

Further, the soccer server allows a player to perform some actions. Depending on the play mode, not all action are allowed, so only allowed actions will be applied and will take effect.

The overview of those actions follows:

- *catch* - the goalkeeper (and only the goalkeeper) can catch the ball if the play mode is play_on, the ball is within catchable area and the goalkeeper is within the penalty area.
- *dash* - accelerates the player in the direction of his body.
- *kick* - accelerates the ball if the ball is within kickable area of the player and the player is not offside.
- *move* - places the player directly onto a desired position on the field. move is used to set up the team and does not work during normal game.
- *turn* - while dash is used to accelerate the player in the direction of his body, turn is used to change that direction.
- *turn_neck* - turns the player's neck relative to his body. The angle of the head of a player is the viewing angle of the player.

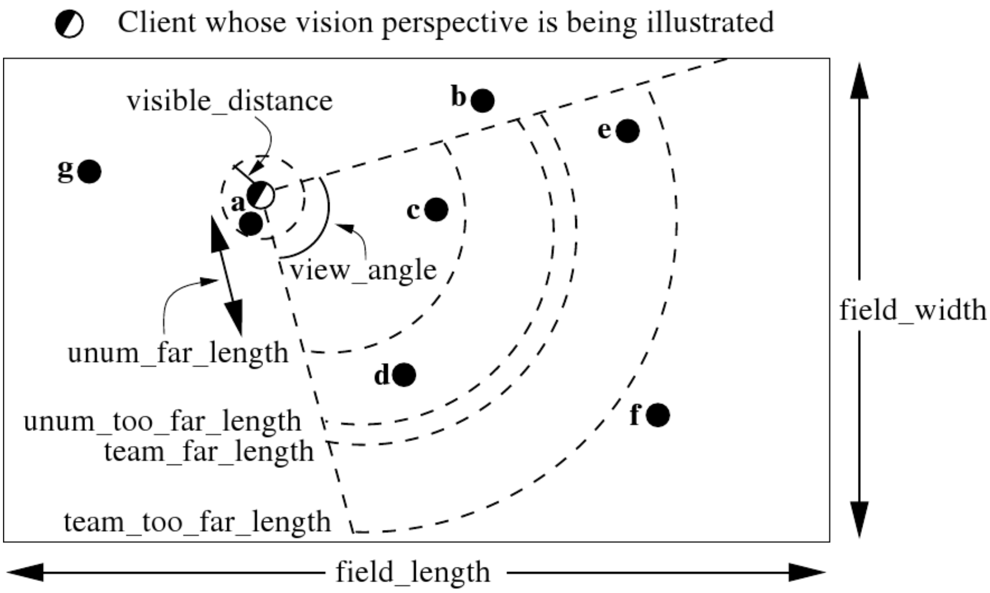

Figure 2.2: The visible range of an individual agent in the soccer server. The light semi-circle is its front. The black circles represent objects in the world. Only objects within *view_angle*/2 and those within *visible_distance* can be seen. *unum_far_length*, *unum_too_far_length*, *team_far_length* and *team_too_far_length* affects the precision with which a player's identity is given. Image and description taken from [3].

Chapter 3

A Method Of Controling The Players

We can assume that, due to the complexity of the environment, agents in the soccer game would be unable to learn effective direct mappings from their sensors to their effectors, even if saving past states of the world. No matter what learning method we would use, the state space is just too big. Hence, the approach picked here is to break the problem into few layers and to use machine learning (ML) when appropriate.

All processes are split into 3 layers. The *Action Layer* where the low level skills take place, the *Cognitive Layer* where the teammate cooperation and strategy take place, and the *Team Formation Layer* where are made strategical decisions affecting the entire team. While building the layers, the bottom-up approach is followed - just like a young soccer player has to learn how to control the ball before he can think of cooperation with his teammates or even of stategy.

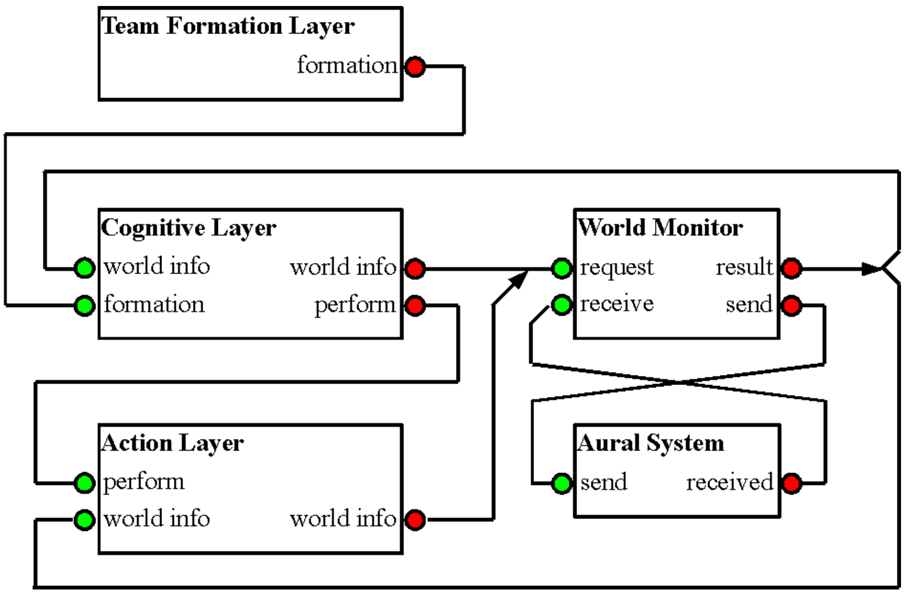

Figure 3.1: Overall architecture DEVS model

Figure 3.1 shows the overall architecture. There can be seen all already mentioned layers and a handful of auxiliar components.

The *Aural System* is an interface over the aural system implemented by soccer server. The information about the environment is interchanged amongst teammates, so they can create more precise world model.

Other components are described in separated sections.

3.1 World Monitor

The *World Monitor* uses sensory information to create the world model and on demand sends the collected knowledge to action and cognitive layers. The architecture of the world monitor is shown on figure 3.2.

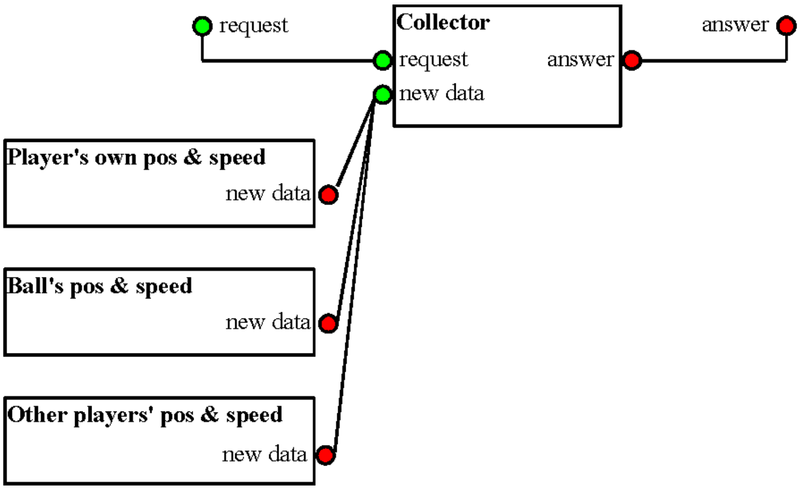

Figure 3.2: DEVS Model of World Monitor

The information to be collected contains:

- The position and speed of the player's own self.
- The position and speed of the ball.
- The position and speed of other players on the field. And whether a player is a teammate or an opponent.

All position and speed vectors are stored as absolute vectors, not as relative ones.

At some accasions, there are not enough sensory data, to collect all required information, therefore the World Monitor uses the Aural System to ask teammates questions (for example - request/answer player position, request/answer ball position)

3.2 Action Layer

Figure 3.3 shows the scheme of the action layer. There are a number of separated blocks, each hosting one basic skill. Each of this basic skills can be implemented either by hand or using a machine learning technique.

The skills are as follows:

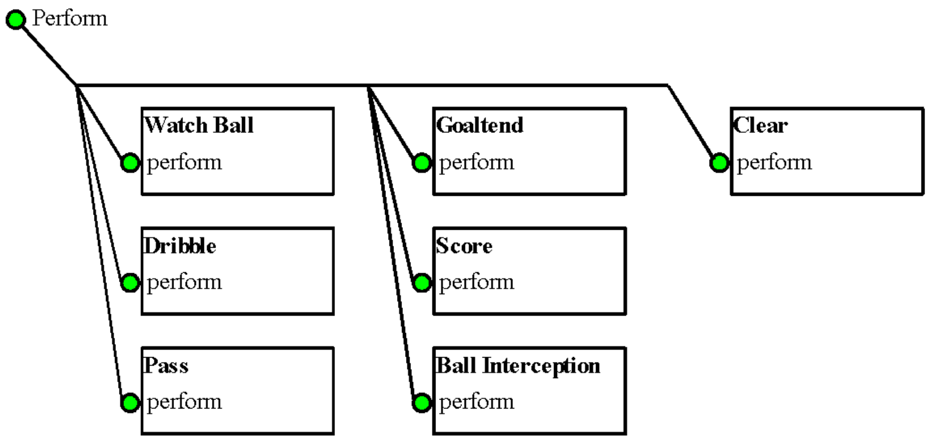

Figure 3.3: DEVS Model of Action Layer

- *Watch Ball* - the ball is too far for the player to take action. Then he just watch it and awaits his opportunity.
- *Dribble* - the skill that allows player to move on field while keeping the ball close to himself all the time. That can be done using alternate kicks and dashes (see chapter 2.2).
- *Pass* - the player shoots the ball to a teammate.
- *Goalkeep* - this skill is used only by a goalkeeper. When the ball is getting closer the to the goal, the goalkeeper tries to catch it.
- *Score* - an attacker is near the goal and tries to score.
- *Ball Interception* - the player moves as close to the ball's trajectory as possible.
- *Clear* - the player takes over the ball from an opponent and kicks it into the field.

3.3 Cognitive Layer

Figure 3.4 shows architecture of this layer. There are a number of components used for evaluating alternatives for tactics, call them evaluators. The task of these evaluators is to estimate whether or not the player will succeed if he takes given action.

- *Dribble Evaluation* - estimates how likely it is that the player will succed dribbling to a given point, without losing the ball in behalf of an opponent.
- *Pass Evaluation* - estimates whether or not would be successful an attemt to pass the ball to a given teammate.
- *Score Evaluation* - estimates the probability that the ball will make its way into opponent's goal if shooting from current position.

Further, there is a box called *Decision Making*. This block sends commands to the action layer. The decision of which command to perform is based on current formation, player's role in the formation and information from the evaluators.

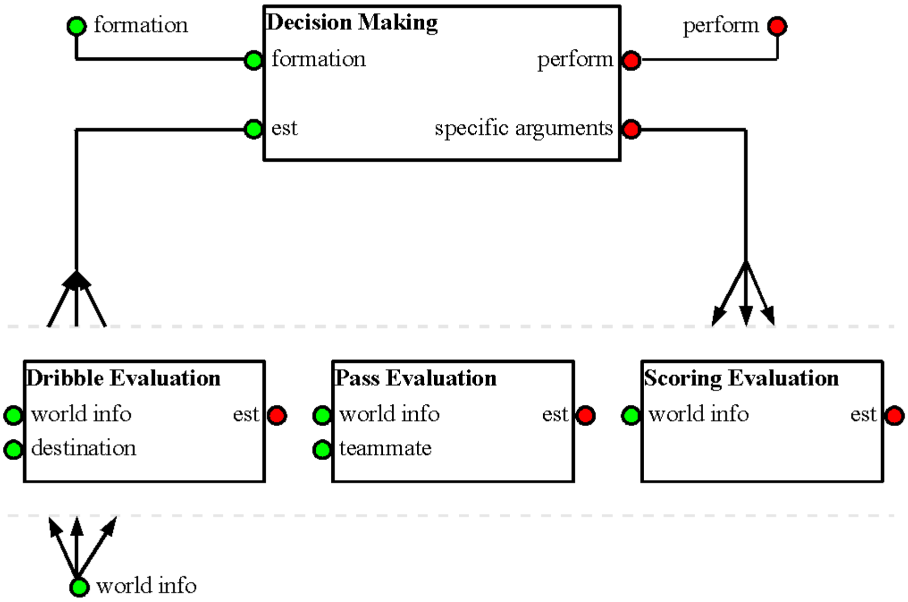

Figure 3.4: DEVS Model of Cognitive Layer

The evaluators are in fact classifiers - they split state space into 2 classes - successful and unsuccessful. Thus it will be beneficial to use machine learning methods, instead of implementing them by hand. The decision making can be implemented by hand or there can be used a reinforcement learning. I'll examine both approaches.

3.4 Team Formation Layer

In soccer, the formation describes how the players in the team are positioned on the field (see figure 3.5). Different formations can be used depending on whether the team wishes to play more aggressively or more deffensive. Formations are described as the number of players in each area from the deffensive line.

The *Team Formation Layer* observes the current state of the game - such as the play mode, the remaining time, the score and so on - and then switches the team formation appropriately. For example, if the team is loosing the game, and there is not much time left, then this layer could switch the team to a more aggressive formation. On the other hand, if the team is winning, it could switch it to a more defensive one.

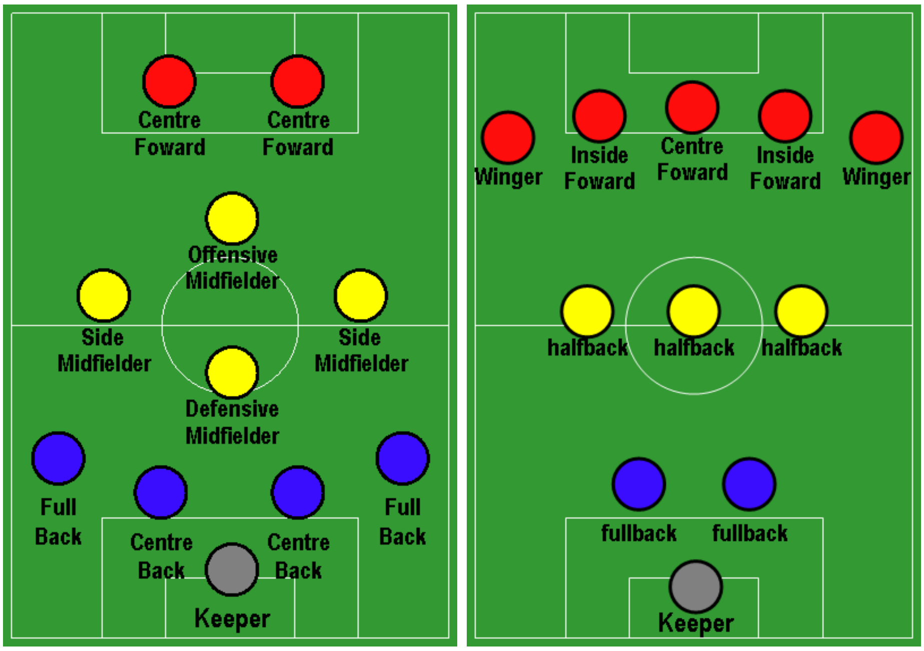

Figure 3.5: Examples of formations. Image taken from [1].

Chapter 4

Conclusion

The robotic soccer requires the players - agents to act in real-time, noisy, collaborative and adversarial environment. Further, due to the complexity of the environment, there is no feasible way of mapping sensory information directly onto actions, even if saving past states of the world. A hierarchy is needed and this work proposes one. In consequential work, I'll fully implement the proposed architecture and evaluate whether or not it works properly.

Bibliography

[1] http://en.wikipedia.org/wiki/Formation_(football)

[2] http://en.wikipedia.org/wiki/RoboCup

[3] RoboCup soccer server Users manual
http://sserver.sf.net/docs/manual.pdf

[4] Layered Learning in Multi-Agent Systems, Peter Stone, 1998