Teaching Artificial Intelligence Techniques Using Multi-Agent Soccer

I.J.J. Borm
Delft University of Technology
Mekelweg 4, 2628 CD Delft,
The Netherlands
iweinb@gmail.com

L.J.M. Rothkrantz
Delft University of Technology
Mekelweg 4, 2628 CD Delft,
The Netherlands
L.J.M.Rothkrantz@ewi.tudelft.nl

ABSTRACT
This paper describes a method of teaching rule-based reasoning, ad-hoc networks, multi-agent systems and agent technology through multi-agent soccer. The coursework is a set of assignments that require students to implement intelligent agents that can control the soccer players. The players form an ad-hoc network that is utilized for communication and cooperation. Students have to beat the reference team to pass the assignment, where a competition between student teams provides an extra incentive for the students. We implemented a running version of the system. The system was tested in a classroom environment. The assignment, system and test results will be discussed in the paper. The coursework presented in this paper bridges the gap between theory and reality in a fun, motivating way.

Categories and Subject Descriptors
K.3.1 [**Computers and Education**]: Computer and Information Science Education – *computer science education.*.

I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems – *games*

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence – *coherence and coordination, intelligent agents, multi-agent systems.*

General Terms
Algorithms, Documentation, Design, Experimentation

Keywords
Agent Framework, Ad-hoc Networks, Distance Learning, Multi-Agent Soccer, Multi-Agent Systems, Programming Assignments.

1. INTRODUCTION
Multi Agent Systems (MAS) are an increasingly popular domain in AI. A special example of MAS are robot or embodied agents. One of the challenges of this domain is "By the year 2050, develop a team of fully autonomous humanoid robots that can win against the human world soccer champion team" [4].

Researchers are looking at the high-level and low-level aspects of building a team that can reason autonomously and cooperate with its team mates to achieve this ambitious goal.

The threshold for starting in this domain however, is rather high. A lot of low-level issues have to be resolved before reasoning on a strategic level can be done. Robots are primarily concerned with things they can determine about their environment – is this robot friend or foe? Is that object a robot or a ball? Where am I? How fast is the other player moving? Although (probabilistic) answers to these questions can be given, the processing time, camera quality and position tracking are some of the problems that harm a proper strategic approach to a soccer team.

The coursework presented in this paper provides students with a multi-agent soccer simulator that has all low-level functionality preprogrammed. A simple interface for controlling players is provided, allowing students to start immediately on a strategic level, without being bothered by the low-level image processing and reasoning bottleneck. There is no central authority – to better resemble reality, and to keep the focus of the assignment on controlling the players.

Students have to program a team of intelligent agents that control the players. The team has to communicate and cooperate in order to beat the reference implementation. To further stimulate students, a competition is held. Through designing and implementing their teams, students learn about Multi-Agent Systems (MAS), Ad-hoc Networks, Rule-based Reasoning and Agent Technology.

The coursework uses a Java-implemented Agent Framework specifically designed for teaching (introductory) artificial intelligence.

A classical approach of teaching students AI is through (conventional) programming assignments. We designed a new method, and expect students to be highly motivated and to learn more.

In this paper we shall first provide the background information about the introductory AI course to which the coursework belongs. Then a brief overview of the simple agent framework on which the coursework is built is given. Then the soccer simulator is explained. The assignment forming the coursework will be explained next. Finally, survey results and classroom experience will be discussed

2. INTRODUCTORY COURSE ON AI
The coursework presented in this paper was used in an introductory first-year undergraduate course on AI. Participants are all from Media and Knowledge Technology, a variant of Computer Science studies at Delft University of Technology. The course started in 2001-2002 and aims to achieve the following [1]:

- Introduce basic concepts of knowledge engineering and relevant AI techniques including search algorithms, knowledge representation methods, rule-based reasoning algorithms, and agent technology.

- Explain and instruct in issues related to AI programming in general and intelligent MAS in particular.

Rather than pursuing these goals through conventional programming assignments, the project consists of a number of challenging (group) assignments.

The course consists of 20 hours of lectures – followed by an oral exam, and 80 hours of practical work.

The teachers are actively involved in supporting the students throughout the project. Through a kick-off lecture, the assignments are introduced. Groups are formed based on performance in the individual assignment, creating homogeneity in groups. Good groups are given more freedom and are stimulated to be more creative, whereas extra guidance is given to the groups that did not perform as well.

The teachers are always present during the lab hours, for answering questions and monitoring the groups. Every week, a brief group conversation with the teacher is held to monitor progress and recognize problems within groups.

The project consists of 4 assignments; A,B,C and D.

- The first is an individual assignment where a Roshambo agent has to be programmed that defeats our reference agents. This assignment filters out students that lack the programming skills required to succeed in further assignments. Throughout the rest of the project, it is assumed all students are capable of programming java at a reasonable level. All further assignments are in groups of 5-6 students.
- The second assignment focuses on rule-based reasoning. Based on questionnaire data, the ideal group member is found.
- The third assignment introduces students to semantic networks. A network of the International Movie Data Base (IMDB) is created and the 'hero' of Hollywood determined.
- The last assignment, D, has always been problematic. The goal of the assignment is originally to teach students about MAS and provide insight in the difficulties that are encountered when dealing with MAS. Various approaches have been attempted, but both software and conceptual flaws in the assignments have so far prevented this assignment from being a success. A new assignment was created in 2004-2005 [1]. Although perceived as enjoyable and motivating, the artificial point distribution of the assignment caused simple strategies to win from intelligent ones.

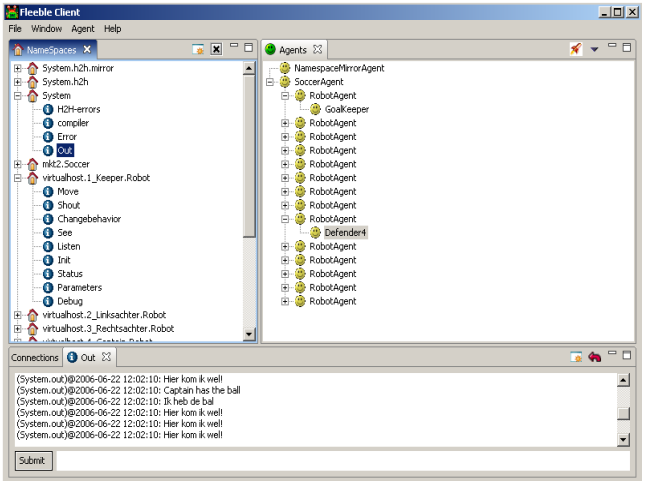
Throughout the project, students become more acquainted with the agent framework (Fleeble), programming agents in java and working together in groups. Students also learn to deal with increased amounts of freedom. For the first and second assignments, the path from goal to implementation is relatively straightforward. Students know what steps need to be taken and implement them accordingly. In the third and fourth assignments, a lot more freedom is given. The goal is clearly specified, but several paths can be taken, none necessarily much better than the other.

3. FLEEBLE AGENT FRAMEWORK

The first-year undergraduate students participating in the project have only little experience in programming Java. To keep the focus of the project on the assignments rather than on learning a complex agent framework, a simple agent framework called Fleeble was developed in previous years of the project.

Fleeble is Java-based and provides all functionality required for the project. It allows concurrency (multi-threading), multiple agents (and easy communication between these agents) and namespaces. A thorough description of these (and all other) features can be found in [2].

Namespaces simulate different computers. When loading a child Agent, a certain name space can be given and Fleeble will lock the Agent's communication to this namespace. This is particularly useful for multi-agent soccer, as it enables us to force namespaces on player agents, such that they can only communicate with the framework and not directly with each other. Fig. 1 shows the Fleeble GUI, with the soccer simulator running.



A comprehensive tutorial can be found in [2]. Templates for agents are included in the assignments, to give students a head

Figure 1. The Fleeble GUI

start. Fleeble will automatically compile agent code, so the only requirements for working with Fleeble are an editor, Fleeble, and Java 1.5 installed.

Because of its excellent documentation, tutorials and example code, only basic Java knowledge is required to start working in Fleeble.

4. MULTI-AGENT SOCCER SIMULATOR

The goal of the assignment is for students to learn about MAS, ad-hoc networks and rule-based reasoning. To help students in achieving these goals, while not distracting them from low-level problems and concerns, the soccer simulator was designed (See Fig. 2).

Figure 2. The Multi Agent Soccer Simulator

Teams consists of 7 players. A player on the field is a *'Robot Agent'* (the hardware), which is controlled by a *'Player Agent'* (the brains). The *Player Agent* has a number of inputs (Listen, See), and a number of outputs (Shout, Move).

4.1 Input / Output

Shouting can be done at any time by any agent. Agents *listening* within a predefined distance (See Fig. 3) will receive the message.

Figure 3. Part of the soccer simulator. The yellow circle defines the shouting distance, red arc is what the player sees

A shout can include any Java Object, so it can be very valuable to exchange information about positions and strategies of other players. Rather than restricting shouting by allowing only a certain number of shouts per time unit, or by setting a maximum message size, shouting is penalized by reducing the speed of the shouter by a certain percentage for a few seconds. This causes students to have very diverse strategies. Some will be hesitant to use shouting often, whereas others will take the speed penalty for granted.

Players receive a visual update about 3 times per second, through the *See* channel. This consists of a list of all *Visible Objects*. A *Visible Object* contains position information, the name and team of the player. The player self, and all objects within a certain distance / angle are added to this list (See Fig 3.). This information, and what is gathered through *listening* are the only sources of input that a player has.

The position information contained in the visual updates comes in three flavors. Absolute (screen) coordinates, relative to team coordinates, and relative to self coordinates. The relative to team coordinates are very useful for providing orientation – a team has to work on both left and right sides of the field. The relative to self coordinates make life a lot easier for students for determining how far away objects are, whether they are on your left or right side, and several other useful facts.

Players can *move* their *Robot Agent* by telling it to move (Forward, Backward, Left or Right). A turn can be specified as well (Left, Right, Straight) and a player can block or kick. Although *Player Agents* are not restricted in the amount of move requests they send, only the last one that is received before a frame update is processed. Bumping into other players, or being bumped into is penalized by a fixed speed reduction lasting several seconds.

Since it is not possible to know beforehand the strategy of the team you are playing against, it would be nice to have some autonomous adaptive behavior. Therefore, an option to change the 'brains', the *Player Agent* on a *Robot Agent* is included. When a team is losing with 5-0, it is probable that proceeding with the current strategy is not going to yield better results. Changing an offensive player to a defensive one, or vice versa, is an interesting dimension that motivated students could explore.

4.2 Randomness

The simulator is mostly deterministic. All parameters are known to all players, and can be used to calculate for instance where a ball will come to a stop, how much time it will take to move to a certain position, etc. There are only two random elements in the game, causing every run to be different.

First, there is a random factor when kicking the ball. To prevent lucky shots from a very large distance, and to stimulate strategic behavior, a certain random distortion is added to every shot.

Secondly, there is *"Java-induced randomness"*. There are 2x7 *Robot Agents*, 2x7 *Player Agents* and a *Framework Agent*. All Agents have their own thread, and thread handling in Java is not completely deterministic. Because of this, sometimes a certain player will get his *See* updated just before a frame update, and another player after. These random variations are equally distributed amongst both teams, and make every game unique.

4.3 Team creation

A soccer team consists of 7 players, each with a *Player Agent*, name, base position and an icon. The information is stored in an XML file. The XML file can be hand written, or generated using a visual editor (See Fig. 4).

Figure 4. Interface for creating team setup XML

5. MULTI AGENT SOCCER ASSIGNMENT

The assignment is formulated as follows:

Design and implement a number of Player Agents that cooperate in order to win the game. It is only possible to pass the assignment if the team convincingly beats the reference team. A team consists of 7 players. A player on the field is a 'Robot Agent' (the hardware), which is controlled by a 'Player Agent' (the brains). The Player Agent has a number of inputs (Listen, See), and a number of outputs (Shout, Move). There is no central authority. The user has no influence whatsoever once a game has started. The focus of the assignment should be on cooperation and communication. A lot of thought has to be spent on what approach and strategies are best, and what the weak spots of these would be.

To provide a guideline to the students, the assignment is split up into three parts. In the first part, students have to think about a number of predefined scenarios, and form a strategy for each. The second part requests students to design their *Player Agents*, to provide a clear picture of what their team will do. The third part is the actual implementation of the team.

5.1 Scenarios

In the first part of the assignment, students have to describe how their players will act in a number of scenario's. A differentiation may be made between types of players (i.e. a defense player will respond different from an offense player).

Scenario's are divided in micro (player) and macro (team) scenarios. An example micro scenario is: *'The ball is free'*. Students have to reason about how their agent reacts. Will it work towards the ball, get in between the goal and the ball, communicate to see if other players are closer to the ball? An example macro scenario is: *"Someone from your team has the ball"*. Do you try to stand in a free spot? Do you help him by communicating position information of enemies?

A lot of decisions have to be made in this part of the assignment. Letting students think as their player agents would, is a good exercise for getting them to understand difficulties and tradeoffs that have to be made.

Before being allowed to proceed to the next part of the assignment, the teacher has to approve of the scenarios.

5.2 Design

During the design phase, students will have a good idea of what is possible and what is not. They use the scenarios from the first part to come up with a full description of their system. This includes additional scenarios (the scenarios from part one only included basic, trivial events), different players (i.e. goalkeeper, defense, captain, offense,…) and a plan for how communication and cooperation between these shall occur.

5.3 Implementation

The final phase deals with the actual implementation of the agents. Considering that Fleeble and the soccer simulator are still (relatively) new to the students, it is impossible for them to estimate beforehand exactly how much time a certain task will take, and whether everything will have the expected outcome. Even for experience programmers, this is a hard task, but for novice programmers, it is a serious problem. Even when their design is perfect, and time constraints seem reasonable, the result will still depend on the individual skills of the group members. It is important to use their design as a guideline, but students will need to learn to iteratively refine their design as the implementation continues.

In this phase, students start working with rule-based reasoning. Although during the year 2005-2006, it was neither obligatory, nor stated in the manual, all groups used a rule-based approach to programming their agents. The rules primarily defined the behavior of the agents. How a rule-based approach is incorporated in soccer agents is shown in 5.4

5.4 Reference Team

To get a passing grade, a student team has to convincingly beat the reference team. Students do not get to see the code for the reference team, but they can observe how it works by playing at it.

The reference team consist of one goalkeeper agent and six *Simple Players, spread over the field.* The reference team uses a rule-based approach to reason about its environment. Every time input arrives through *See* or through *Listen*, the list of positions of all objects is updated. Every time a *See* comes in (3 times per second), the known info is updated and a number of Boolean values are derived. These Booleans deal with: Is the ball position known? visible? kickable? free? in enemy goal area? near base position? with a team player? Is there a team mate standing free? An enemy up ahead? Is it total chaos?

The values of these Booleans are then put into a Rule base, which

```
IF ball_kickable AND NOT in_enemy_goal_area AND team_mate_free
THEN pass_ball
```

reasons about the situation. For example:

The pass_ball fact will then cause the player to kick the ball.

The success of any given strategy depends on their analysis (what Boolean values were derived and how well they were

recognized), the Rule base, and the implementation of the actual actions.

The reference team does not communicate at all. All players reason autonomously about what to do.

The keeper will try to stay between the goal and the ball. It will try to take the ball when it is free, and when it has the ball, shoot it into the field. When it doesn't see the ball, it will stay on its base position and scout for it.

The Simple Player will also scout for the ball when it doesn't see it. When the ball is visible and it is no total chaos, it will hunt for the ball. When a team mate has the ball, it will try to stand in a free spot. When it has the ball, it will try to avoid enemies and attack over the flank at which it was positioned (a left back will attack over left flank), moving directly towards the goal when it is getting close. When a ball was recently visible, but isn't anymore, it will try to find out where it is by turning and moving towards it.

The resulting team was neither very smart nor very stupid. It was a challenge for students to convincingly beat the reference team.

6. RESULTS

Evaluation of the assignment occurred through classroom observations and through a survey. After a discussion of the results coming from these, a comparison is made with previous assignments.

6.1 Classroom Results

The expected result was that students would find the assignment enjoyable, motivating and very educative. Based on classroom observations and several conversations with participating students, this is also the actual result. A few things were remarkable, however.

First of all, there were hardly any questions about the assignment all. With previous assignments, and especially with the introduction of new assignments, there have always been loads of questions. The only occasion the teacher was really necessary was for approving scenarios and designs. This leads to believe that the assignment is very well suitable for distance learning.

Second of all, it was remarkable that a wide variety of approaches were attempted. The creativity and skill of individual group members have been put to good use in designing and implementing the agents, and gave diverse results.

A third remarkable observation was that several groups gave the players the group members names. Rather than referring to a certain player as 'Right back', or 'Defense Player' – as expected, they felt affinity for their players. It is also interesting to note that many groups were so enthusiastic about the assignment that they worked on the implementation far beyond the lab hours. Even after knowing their team was good enough to beat the reference implementation, many groups made a big effort to try and win the competition between the groups.

Many students were pleased with their implementations, but regretted to see that a lot of their hard work had turned out useless as it was not used. Getting a complicated strategy to work may look trivial on paper, students gained first-hand experience in the hard reality that it isn't.

Another remarkable result is the results of the tournament. Since groups were formed based on homogeneity in performance, the expected result would be to have group 1 as champions, group 2 as number 2, etc. Rather than this happening, groups 4,3 and 5 (out of 5 groups) finished 1st, 2nd and 3rd. Although group 1 could have won, their approach was too complicated, causing it not to work. Most groups encountered similar problems, but the urge to win the competition caused some groups to invest a lot more time.

Finally, at the competition between the groups, it was remarkable to see how enthusiastic students were about winning, and how much they enjoyed seeing their strategies at work.

Regarding the educational goals for the assignment, these were achieved through all parts of the assignment. Students gained a lot of insight in the problem of dealing with decentralized control in ad-hoc networks through the scenarios and design. Through implementing the players, most groups found that complexity kills. A clever combination of rather simple methods yields a better result than a poorly implemented ingenious approach.

Students used a Rule-based approach in their implementations, but did so without using separate rule base software. Separating the rules from the code would force students to program in a better organized manner. Changing the assignment to explicitly include this element would probably help students a lot during the implementation.

6.2 Survey Results

After the competition, a survey was distributed among the students to get their opinion of the assignment, the manual and the software. It was remarkable to see several questionnaires with a (very) positive rating, and a lot of useful feedback in the *remarks* section on the one hand, and several very negative results without any remarks.

Because the survey is anonymous, it is not possible to know which students gave what kind of feedback, but a plausible explanation for the result is that within groups, there are two subgroups: Those that did a large share of the programming work, worked with the simulator and read the manual, and those that prepared the group's presentation, wrote reports, helped the programmers and took part in the design process. The first group generally has a positive opinion (this is in agreement with classroom observations), whereas the second group does not.

One of the questions was *"Describe in 5 keywords the things that you learning from the assignment:"*

Some common answers were: *"Teamwork, planning, ad-hoc networks, problem solving, cooperation (software), how to implement a (high level) strategy "*

45% of the students indicated that they would like to do more assignments with Multi-Agent Soccer.

80% (strongly) agreed that they learned more from this assignment than a *'conventional'* practical that asks to implement a certain algorithm.

75% felt that the assignment gave them insight in the problems you encounter in environments without any centralized control, specifically ad-hoc networks.

85% (strongly) agreed that the assignment was challenging.

80% liked the assignment (very much).

85% liked the competitive element (very much)

65% felt that they would have been able to finish the assignment without the teacher's help

In the remarks, the most commonly heard complaint was the lack of time. Students only had 3 weeks to finish the entire assignment. Some even proposed to remove assignment B and enlarge this assignment. Another negative remark was the coordinates (relative to team, relative to self) were not explained properly in the manual.

6.3 Comparison

In the past two years, other assignments were given to the students with similar educational goals [1]. It is not possible to compare survey results, as the questions and the way they were posed differ too much. It is possible however to compare the classroom observations.

Two years ago, the assignment was flawed both conceptually and by software. To combat these problems, the *Smurf* assignment came into existence [1]. Although the software was working reasonably well, the artificial point distribution system caused trivial solutions to beat highly intelligent ones. Because of this, no 'good' implementation could be made.

With the lessons learned from previous assignments, the soccer assignment tries to stick close to a realistic environment. Speed penalties are given for communication, rather than an artificial point distribution. Because the assignment was closer to reality, students could identify better with their agents and this in term further increased their motivation.

"It actually works" is a common utterance by students that participated in the course in previous years, after having seen the soccer assignment.

7. CONCLUSIONS

In this paper we have described a method of teaching Ad-hoc Networks, Rule-based Reasoning, Multi Agent Systems and Agent Technology using a multi agent soccer system. We implemented a running version of the soccer system and tested the assignment in a classroom environment.

We gathered results through anonymous questionnaires and classroom observations. Both methods show that students enjoyed the assignment and felt motivated by it. The educational goals were achieved, and many group work related lessons were learned.

Since the assignment only focuses on high-level strategic decisions, rather than the low-level issues currently being looked at in the robotic soccer field, students had more freedom for creative, original solutions. Students learned while implementing that complexity kills. A clever combination between simple methods yields far better results.

Adding the competitive element to the assignment gave students extra incentive for hard work, and the competitive element was greatly appreciated by the students.

8. FUTURE WORK

The assignment is a good challenge for first-year undergraduate computer science students. Due to the level of these students and the short period of time allocated for the assignment, the resulting teams were not spectacular.

It would be interesting to extend the system with various other (more difficult) reference implementations, alter the assignment to defeat these teams and give it to graduate students.

Adding a self-learning reference implementation would add an interesting dimension to the analysis phase of the assignment, as well as provide a real challenge for graduate students.

Most students felt that they could have done the assignment without the help of the teacher. This leads to believe that a modified version of the assignment would be particularly useful for distance learning.

9. ACKNOWLEDGMENTS

The authors would like to thank all MKT undergraduates who evaluated the course in 2005-2006. Special thanks go to Cristiano Betta, Ilyaz Nasrullah and Paul van den Haak for their help in managing the course in 2005-2006.

10. REFERENCES

[1] Pantic, M.; Zwitserloot, R.; Grootjans, R.-J., "Teaching ad-hoc networks using a simple agent framework," *Information Technology Based Higher Education and Training, 2005. ITHET 2005. 6th International Conference on* , vol., no.pp. S2A/6- S2A11, 7-9 July 2005

[2] http://www.fleeble.net/ (last visited: June 23, 2006)

[3] Pantic, M., Grootjans, R.J., Zwitserloot, R., "Fleeble Agent Framework for teaching an introductory course in AI", *Proc Int'l Conf. Cognition and Exploratory Learning in Digital Age*, pp. 525-530, Lisbon, Portugal, 2004.

[4] http://www.robocup.org/ (last visited: June 23, 2006)