# UvA Trilearn 2001 Team Description

**R. de Boer, J.R. Kok, F.C.A. Groen**
{remdboer,jellekok,groen}@wins.uva.nl

**Faculty WINS, Department of Computer Science**
**Intelligent Autonomous Systems Group**
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands

**Abstract**

This paper describes the architecture and scientific goals of the *UvA Trilearn 2001* simulation soccer team. This team is built by two masters students from the Intelligent Autonomous Systems Group at the University of Amsterdam for their graduation project. *UvA Trilearn 2001* is a new team that will participate for the first time in RoboCup 2001. The team has been built from scratch and does not contain any code copied from other RoboCup teams. Therefore much of the initial effort has gone into getting the lower level aspects to work. The intention has been to try to improve upon some of the low-level methods used by other top teams from the past. This has among other things led to an advanced synchronisation scheme and very accurate methods for position and velocity estimation. Much attention has also been paid to the software engineering aspects of the project for comprehensibility and to facilitate future use. Long term scientific goals will focus more on higher level aspects such as multi-agent strategic reasoning, adversarial planning and multi-agent modeling using a coach agent. The use of heterogeneous agents will also be investigated.

## 1 Introduction

In the past the University of Amsterdam has been successful in the RoboCup simulation competition with the team *Windmill Wanderers*, which became 3rd in 1998. Sadly the creator of this team, Emiel Corten, deceased in 1999 and as a result the project came to a halt. By the end of the year 2000 however we were asked to build a new soccer simulation team called *UvA Trilearn 2001* for our graduation project. For two reasons we decided not to continue with the semi-finished *Windmill Wanderers* team or copy any code from other past RoboCup teams. Firstly, we found that the released source codes did in no way conform to regular software standards, i.e. they were poorly structured and poorly documented. We therefore felt that it would be more time-consuming to reuse the code and restructure it into the architecture that we desired than it would be to write it all ourselves. Secondly, we also felt that the lower-levels used by the top teams from recent years could be improved in several ways. We thus decided to start from scratch.

Much time has initially been spent on getting the lower levels to work. Although it is considered to be scientifically more prestigious to work on high-level behaviour, it is our opinion that the lower levels in the architecture are the most crucial for the success of the team. Furthermore low-level imperfections cannot be compensated for by high-level behaviour added later on. We thus realized the significance of the low-level implementation and felt that it was worth spending a lot of time on. We had several ideas for improvements in the low-level methods used by top teams from the past, which we have successfully implemented. Among other things these concern the synchronisation method used and techniques for obtaining better estimates of positions and velocities in the world model of each agent. In each of these cases we have implemented several possible solutions and have chosen the best one based on experimental results. During the implementation we have also paid much attention to the software engineering aspects of the project (modular design, documentation, etc.) to facilitate future use by others after our graduation.

Currently the lower levels of our architecture are almost finished. Using a simple high-level decision module on top of these lower levels we have already succeeded in beating the RoboCup 2000 champion *FC Portugal*. From now on we will concentrate more on higher-level aspects including adversarial planning, multi-agent modeling using a coach agent and heterogeneous agents. This paper is organized as follows. In section 2 our current implementation is shortly described and in section 3 we discuss some of our future plans. This is followed by a conclusion in section 4.

# 2 Current implementation

## 2.1 World model

Each agent keeps a world model that contains information about all the objects on the soccer field. This world model is a probabilistic representation of the world state based on past perceptions. For each object an estimation of its position and velocity is stored (among other things) together with a confidence value which indicates the accuracy of the estimation. We have experimentally compared several different methods for estimating positions and velocities of dynamic objects and used the best of these methods based on statistical results. The world model is updated as soon as new information about the world state is received by the agent and thus always contains the last known information. Objects which are not seen are also updated based on their last observed velocity. The confidence in the estimate decreases however for each cycle in which the object is invisible. The world model also contains various methods which use the low-level world state information to derive higher-level conclusions. The agent can use these methods to reason about the best possible action.

## 2.2 Architecture

The agents of *UvA Trilearn 2001* are capable of perception, reasoning and acting. We use a multi-threaded architecture which allows each agent to use a separate thread for each of these three tasks. In this way we minimize the delay caused by I/O to and from the server so that the agent can spend the majority of his time thinking about his next action. The different aspects of the decision making process are thus clearly separated. We use a 3-layer architecture for each agent. The threads used for perception and acting can be seen as the bottom layer which hides the soccer server details from the other layers. On top of this we have the *Skills Layer* which uses the functionality offered by the layer below to implement the different skills of each player (ball interception, player marking, etc.). The *Control Layer* then selects the best possible action from the *Skills Layer* depending on the current world state.

## 2.3 Synchronisation

We use an advanced synchronisation method which for each cycle determines the optimal moment to send an action to the server. This moment depends on the arrival times of the different server messages and thus changes from each cycle to the next. Furthermore our method guarantees that the action sent is always based on the latest information from the server. After sending an action it is checked whether the server has actually performed it. When this is not the case (i.e. the action did not arrive in time) no action is sent in the next cycle to avoid a clash (two actions in one cycle). Experiments have shown that our synchronisation method clearly outperforms several others used in the past. During a full-length match consisting of 6000 simulator cycles the number of holes (no action during a cycle) for each agent is less than 0.3% of the total number of cycles and the number of clashes is less than 0.01%.

## 2.4 Team strategy

The main philosophy of the *UvA Trilearn 2001* soccer simulation team is to keep the ball moving swiftly from each player to the next and preferably in a forward direction. In addition midfielders often try to pass the ball into the depth to the side of the field thereby cutting quickly through the opponents' defense. In this way we often succeed in moving the ball rapidly to the enemy half of the field thereby disorganizing the opponent team. We use a 4-3-3 formation in which each player chooses a strategic position determined by a weighted sum of its home position and the current position of the ball which serves as an attraction factor. When the opponent team is in possession of the ball and close to our penalty area, our defenders use a simple opponent marking scheme to try to ward off the opponent attack.

### 2.5 Software reusability

In order to make it easy for others to use our code after our graduation much attention during implementation has been focused on software engineering issues. We have consequently followed an object oriented design methodology leading to a logical class hierarchy and highly modular code. We have chosen to use the programming language C++ for performance reasons and our current implementation consists of 17000 lines. We have developed a multi-level log system for debugging purposes. This system enables the programmer to specify the abstraction level from which he desires information thereby speeding up the debugging process. Furthermore a strict version management protocol is used to support concurrent development. Much time has also been spent on extensively documenting our code to facilitate future reuse. For this we have used the documentation system Doxygen.

## 3 Future plans

Our lower levels are currently almost finished. The only thing that we still need to add is a communication protocol which enables agents to receive information from other agents about the state of the world. This will make the world model information of each agent more accurate and will thus lead to improved behaviour overall. After this we will focus our attention more on higher-level issues. Some of our future plans are described below. By the time the RoboCup starts we expect to have these implemented.

- Strategic positioning. We still want to enhance our strategic positioning module by also taking opponents into consideration when determining the agent position. Furthermore agents must also be capable of switching roles inside a formation when for instance they become tired. Since strategic positioning has already proven to be a very important factor, we expect that these enhancements will considerably improve the team's playing capabilities.

- Heterogeneous agents and dynamic resource allocation. What is the best way to divide tasks between agents given the fact that they have different skills and that they grow tired? The latest version of the soccer server offers seven different player types each with their own strengths and weaknesses. One can assess the quality of each player type using a formula which returns a utility value based on the player parameters. It is then possible to determine which player types are best suited for which positions on the field. Wing-attackers for instance probably need to be fast players. Possibly one can use a machine learning technique such as a genetic algorithm to determine which player types are best suited for which roles inside a formation.

- Multi-agent modeling using a coach agent. In our current implementation individual agent decisions are often affected by actions from opponents, but strategy decisions for the team as a whole are mostly fixed. Ideally however the team strategy should be adjusted in response to adversary behaviour. We want to use the coach agent to analyze the strengths and weaknesses of the enemy team and to give advice on the best possible strategy. To this end the coach must be able to classify enemy behaviour based on certain features. He can then decide which strategy is most appropriate for the given behaviour class. The coach can for instance determine on which side of the field an enemy attack is most dangerous and recommend to change the formation or switch player types for a certain position inside the current formation. In turn the players must know how to interpret the coach's advice and how to act upon it.

## 4 Conclusion

In this paper we have discussed the current implementation and future plans of the *UvA Trilearn 2001* soccer simulation team. Until now most attention has been focused on the lower levels in the architecture and on software engineering issues. We want to emphasize the fact that software reusability is very important and strongly encourage everyone to spend more time on properly structuring and documenting their code. This will facilitate future use by other researchers inside as well as outside their own community.

Our lower levels are currently almost finished. We have tested them by playing a simplified version of the RoboCup 2000 champion *FC Portugal*. This team uses a very simple high-level decision module that we also used on top of our own lower levels. Over the course of 10 games we managed to beat this team by a total score of 20-17. We can thus conclude that our lower layers are at least as good as those of *FC Portugal*. Since the qualification deadline for Seattle was much sooner than we expected we were then forced to quickly start working on our higher levels. We have extended our simple decision module to encompass the team strategy described earlier and already succeeded in beating *FC Portugal* on a couple of occasions. Up to this point we have not been able to spend much time on the future plans described in the previous section. By the time the RoboCup starts in august however we expect to have these implemented. *UvA Trilearn 2001* will also participate in the German Open in the beginning of june. This will serve as a testcase for Seattle. Since many enhancements are still possible at this moment we are currently very happy with our results and feel that we have achieved a lot in a short time.

# References


[1] Mike Bowling, Peter Stone, Manuela Veloso. *Predictive Memory for an Inaccessible Environment.* In Proceedings of the IROS-96 Workshop on RoboCup, pages 28-34, Osaka, Japan, November 1996.

[2] Frederick P. Brooks Jr., *The Mythical Man-Month: Essays on Software Engineering*, 20th anniversary edition, Addison-Wesley, 1995.

[3] Marc Butler, Mikhail Prokopenko, Thomas Howard. *Flexible Synchronisation within RoboCup Environment: a Comparative Analysis.* In Proceedings of the RoboCup-2000 Workshop.

[4] Emiel Corten, Klaus Dorer, Frederik Heintz, Kostas Kostiadis, Johan Kummeneje, Helmut Myritz, Itsuki Noda, Jukka Reikki, Patrick Riley, Peter Stone, Tralvex Yeap. *Soccerserver Manual Version 5.0beta2.* http://www.dsv.su.se/ johank/RoboCup/manual, 1999.

[5] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, Eiichi Osawa. *RoboCup: The Robot World Cup Initiative.* IJCAI-95 Workshop on Entertainment and AI/Alife, 1995.

[6] Kostas Kostiadis, Huosheng Hu. *A Multi-threaded Approach to Simulated Soccer Agents for the RoboCup Competition.* In Proceedings of the 16th IJCAI RoboCup Workshop, pages 137-142, 1999.

[7] Kostas Kostiadis, Matthew Hunter, Huosheng Hu. *The Use of Design Patterns for the Development of Multi-Agent Systems.* In Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC2000), Tennessee, October 2000.

[8] Roger S. Pressman. *Software Engineering: a Practitioner's Approach.* 4th edition, McGraw-Hill, 1997.

[9] Luís Paulo Reis, Jośe Nuno Lau, Luís Seabra Lopes. *FC Portugal Team Description Paper.* http://www.ieeta.pt/robocup/archive.htm, 2000.

[10] Peter Stone, Manuela Veloso. *Task Decomposition and Dynamic Role Assignment for Real-Time Strategic Teamwork.* 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 293-308. Springer-Verlag, Heidelberg, 1999.

[11] Peter Stone. *Layered Learning in Multi-Agent Systems.* PhD thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, CMU-CS-97-193, Computer Science Department, Carnegie Mellon University, 1998.

[12] Peter Stone, Manuela Veloso. *Multi-agent Systems: A Survey from a Machine Learning Perspective.* In Autonomous Robotics, volume 8, number 3, july 2000.