The Design of an Object Oriented Agent System for Robotic Soccer

Johan Kummeneje and Håkan Younes
The DECIDE Research Group
Department of Computer and Systems Sciences
The Royal Institute of Technology and Stockholm University
Electrum 230, 16440 KISTA, Sweden
E-mail: johank@dsv.su.se, lorens@acm.org

Abstract
In this paper, we describe the design of an object oriented agent system for robotic soccer. We expect the design to be very reliable, though it is not yet fully implemented. A short survey of related work within the field of RoboCup is included.

Keywords
Multi-agent systems, robotic soccer, RoboCup, systems design, object oriented development

1 Introduction

We will describe the work on a team in the RoboCup domain (see section 3) at the Department of Computer and Systems Sciences, and give our reasons for certain design choices. Students and researchers of Stockholm University and the Royal Institute of Technology are working on the team named *Utility Based Reasoning under Uncertainty* (UBU).

1.1 Background

In the fall of 1997, the idea of a RoboCup-team of agents[1] was realised by Helena Åberg and Åsa Åhman, as a partial fulfilment of their master's degrees (Åberg 1998, Åhman 1998). They developed the foundation in Java upon which a part of the participating team in Paris (Andreasen et al. 1998) was based. At this stage, the small group of Magnus Boman, Åberg, and Åhman handled the development. Unfortunately, Åberg and Åhman finished their theses before the team was finished, and at approximately the same time, there were major updates in the core of the RoboCup environment.

Jens Andreasen, Mats Danielson, Johan Kummeneje, and Harko Verhagen extended the team in the spring of 1998. Kummeneje and Verhagen joined as researchers, while Danielson became the supervisor of the master's thesis of Andreasen. Andreasen wrote the chief part in C while Kummeneje extended the work of Åberg and Åhman into the goalkeeper.

As part of the human world cup, the RoboCup-federation held the second official world cup of RoboCup, at the Cité des Sciences et de l'Industrie in the suburbs of Paris during the first days of July of 1998. The result of our participation was better than expected, but not satisfactory, as UBU won one game, against Brazil and lost the other three. During the summer of 1998, Johan Sikström and Håkan Younes joined the team focusing as parts of their master's theses on information representation and artificial decision-making respectively.

We have since November 1998 focused on developing a new platform upon which we can build the team. Our design choices and main ideas about the platform are described in this paper.

[1] Agents are defined in a variety of ways in (Engström and Kummeneje 1997, Russell and Norvig 1995, and Genesereth and Ketchpel 1994).

1.2 Design goals

The team was badly tested and not easily understandable, which implicates problems with the extensibility of the system. Therefore we decided to start over in order to develop a system with which our own research could benefit. The design of our system is not exclusive to our research interests, as the RoboCup-field is interesting to other research fields, e.g., machine learning and social sciences.
The chief aim of developing UBU is to subject a series of tools and procedures for agent decision support to a dynamic real-time domain.

One of the chief ideas of our future research is that the reactive layer should be directly coupled to the decision support, which we are currently implementing as several support modules and a situated automaton. Those layers are currently being implemented, and will be described in another paper.

This agent system is tailor-made to fit the demands of the RoboCup environment, and might not be applicable to other agent systems, however, we believe that some parts might be transferable. Furthermore, we have put a lot of thought into making it adaptable to other parts of the RoboCup environment, such as the robotic teams.

1.3 Outline

In section 2, we will describe related efforts, followed by an introduction to the concepts of RoboCup in section 3. Thereafter we will present the design of UBU in section 4, after which we conclude and discuss our future work in section 5.

2 Related Work

We are currently focusing on the object-oriented design of the primitive functionality of our team. The future research will be focused on the co-operation between the agents through norm adaptation, as well as on the utility based reasoning under uncertainty. The following examples are the most important and research related to our research at present.

Peter Stone, who recently received his Ph.D. at the Carnegie Mellon University, has in co-operation with Manuela Veloso and Patrick Riley developed a team, CMUnited, which became the champion of the World Cup 1998. Stone's research is focused on machine learning (Stone 1998).

Milind Tambe and Gal Kaminka at the University of Southern California, have used a rule-based framework approach when developing their team ISIS (Tambe et al. 1998) in Soar. Tambe and Kaminka do research in the field of teamwork and social diagnosis, which is not far from the research on social norms performed in our lab (Boman and Verhagen 1998).

Oliver Langer at the Technical University of Chemnitz has developed an object oriented team (Langer 1997). The difference between Langer’s design and ours is that we have built a layered platform which can be used in a range of RoboCup leagues while Langer’s is more low-level than ours, with a greater focus on the RoboCup Simulator League.

3 Introduction of RoboCup

In this section we will try to give an introduction to the relevant aspects of RoboCup, first by describing the research-field in general (section 3.1), and secondly by describing the simulator league (section 3.2).

3.1 RoboCup in General

The idea of robots playing football was introduced in the early 1990’s as a research area by the Canadian professor Alan Mackworth (Mackworth 1993). Unfortunately, the idea did not get the proper response until the idea was further developed and adapted by Hirako Kitano, Minoru Asada, and Yasuo Kuniyoshi

in 1993, by proposing a Japanese research programme, called Robot J-League[2]. During the autumn of 1993, several American researchers took interest in the Robot J-League, and it thereby changed name to the Robot World Cup Initiative or RoboCup for short.

Kitano, Asada, and Kuniyoshi more formally proposed in 1995 the first *Robot World Cup Soccer Games and Conferences* to take place in 1997 (Kitano et al. 1995).

The aim of RoboCup is to present a new standard problem for Artificial Intelligence (AI) and robotics, something jokingly described as the life of AI after Deep Blue[3]. RoboCup differs from previous research in AI by focusing more on a distributed solution instead of a monolithic solution as AI does, and also by challenging researchers from not only traditionally AI-related fields, but also researchers in the areas of robotics, sociology, etc.

RoboCup has set goals and a timetable[4] for its research in order to push the state-of-the-art further by providing a formalised testbed.

The RoboCup teams consist of either robots or programs that co-operate in order to defeat the opponents. Today the RoboCup consists of several leagues: e.g. the Simulator League, Small Robots League, Middle-size Robots League, Sony Legged Robots League[5], Humanoid League, etc.

3.2 Simulator League

The RoboCup Simulator League is based on the RoboCup simulator (Noda et al. 1997) called the soccer server, that is a physical soccer simulation system—documented in (Andre et al. 1999). The soccer server is written to support competition among multiple virtual soccer players in an uncertain multi-agent environment, with real-time demands as well as semi-structured conditions. One of the advantages of the soccer server is the abstraction made, which relieves the researchers from having to handle robot problems, and enables them to focus on concepts such as co-operation and learning.

Since the soccer server provides a challenging environment, there is a need for a referee when playing a match. The referee is partially implemented and can detect trivial situations, e.g., when a team scores. However, there are several hard-to-detect situations in the soccer server, e.g. deadlocks, which brings the need for a human referee. All participating teams have also agreed upon a gentlemen's agreement, which is an agreement, e.g., not to use loopholes.

4 The Design of UBU

In this section we will describe the background of UBU and give a sketch of the design of the teams basic functionality.

4.1 What is UBU?

UBU is an effort within the RoboCup Challenge and the name has a more literary meaning through the reference to the play Ubu Roi, i.e. King Ubu (Jarry 1961) by Alfred Jarry, the father of pataphysics. The development of UBU has now reached the second version, which is written entirely in Java, using several key concepts like multi-threading, encapsulation, polymorphism, listener, and events etc. (Gosling and McGilton 1996).

The greatest disadvantage of Java is the speed of execution, which we had to overcome, as we estimate that each player will have approximately 10-15 threads running throughout the games. Our solution is to

[2] The J-league is the professional soccer league in Japan.
[3] Deep Blue and its games against Kasparov, see http://www.chess.ibm.com.
[4] The goals and the timetable are thoroughly described at http://www.robocup.org.
[5] The league is sponsored by Sony, and there will be a competition with the robots in the World Cup 1999.

implement a clever sleeping schedule, with notifying abilities. For example the higher level while waiting for new information, will sleep for 10 milliseconds and then check if they have been notified about any changes, if not, they will return to sleep. As we have anticipated the need for more speed of execution of the players, we have been careful in the design to enable us to translate the whole system into the faster C++, i.e., we have not used any incompatible concepts.

The team consists of twelve autonomous processes that represent eleven players, and a coach. The team will use different strategies in order to perform as well as possible, e.g., we are using semi-complex situated automata in order to get a mix between reactive agents that is expected to perform reasonable well in 95 percent of the situations (Davidsson 1998), and more deliberate agents which could handle the last 5 percent. We will on top of the situated automaton implement social norms (Boman and Verhagen 1998) to further influence the agents to do the right things.

4.2 Overview of the Layered Approach

The whole architecture is strongly inspired by the OSI-model (Tanenbaum 1996), and therefore by using a layered approach as seen in Figure 1 we have isolated the low-level details from the application, or rather high-level processes. It is possible to exchange the predefined soccer server layer and the communication layer (see section 4.3), by only rewriting at most the primitive action layer (see section 4.4). By utilising the excellent support of threads in Java, we have designed our system to be able to take advantage of multi-processor systems, which are getting increasingly common.

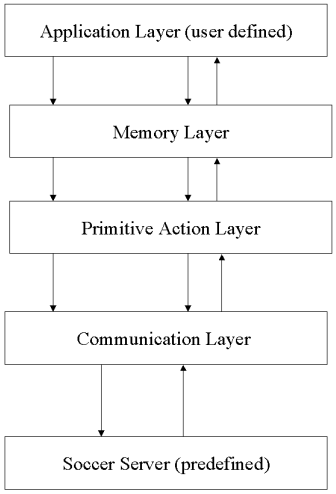

Figure 1. The different layers of the design.

We have also removed the dependency between the interpretation of the input and the representation of the input, e.g., the representation of a ball to the application layer would be the same no matter what the lower levels have to do to interpret the input.

4.3 Communication Layer

The communication layer (see Figure 2) is the interface to the soccer server and is thereby the most low-level part of our design. We will relate the tasks to the numbered boxes in Figure 2. The interface to the higher layers are handled in the boxes numbered 1 and 2, but there are a slight distinction between the boxes, as box 1 handles the reception of data to send from the higher layers. Box 2 allows the higher layers to subscribe to the data received from the soccer server. Box 3 is the most interesting part of the communication layer, as it handles three things:

a) the UDP-connection to the soccer server,
b) the sending of data to the soccer server, and

c) the reception of data from the soccer server.

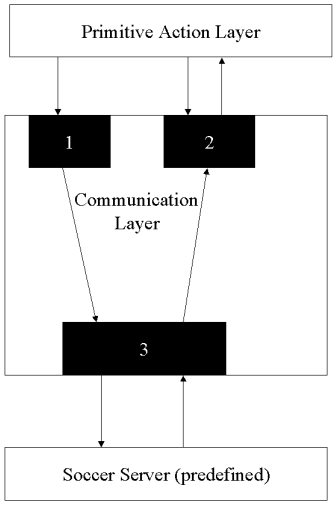

Figure 2. The communicative aspects of the design.

The communication within the communication layer is by nature one-way, since we have divided the sending and the receiving of information in order to be able to change the parsing and memory-classes without having to change the queuing of actions (see section 4.5).

4.4 Primitive Action Layer

The primitive action layer consists primarily of two large classes, the effectorbank and the sensorbank[6]. We will describe the functionality of the four numbered boxes in Figure 3 briefly. The effectorbank consists of boxes 1 and 3, where the reception of actions to perform from higher layers is handled in box 1. In box 3 the mapping of action to a protocol specific message is made and passed on to the lower layers.

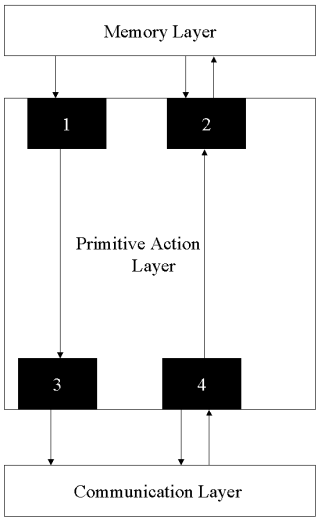

Figure 3. Conceptual view of the primitive action layer.

The sensorbank is the two remaining boxes, i.e. boxes 2 and 4. Box 2 provide a similar service to the subscription functionality of the communication layer, as it allows several higher layer modules to subscribe to any sensorial inputs received, and thereby notified about the exact sensorial information. In

[6] Effectors and sensors are defined in various ways in (Davidsson 1998, Russell and Norvig 1995, and Engström and Kummeneje 1997).

box 4, the sensorbank receives the information from the communication layer and splits it depending on the type of information, e.g., if it is visual information the video sensors and subscribers are notified.

The effectorbank is a collection of effectors, while the sensorbank is a collection of sensors. The idea behind using collections of effectors and sensors is that it becomes trivial to add a sensor or an effector to the system. We have designed the effectorbank to function as a mapping of the desired actions to the corresponding commands in the soccer server. On the other hand the sensorbank is designed to split the different information depending on the type, and send it to the corresponding updating interface in the memory layer (see Figure 4).

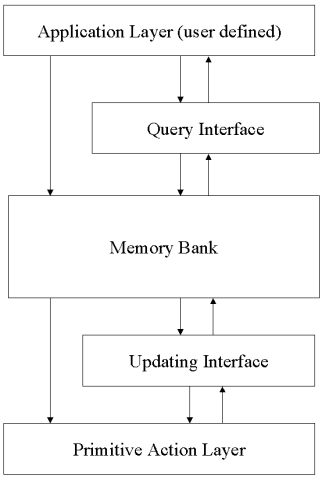

Figure 4. The different layers of the memory layer.

4.5 Memory Layer

We have chosen to divide the memory layer into three sub-layers in order to separate the storage from the updating and querying, as seen in Figure 4. This holds for the information coming from the subsequent layers, i.e. the primitive action layer, the communication layer, and the soccer server, however the information originating in the application layer will be sent directly to a queue in the memory layer.

By using a queue we can adjust the timing to the soccer server[7], and also provide the foundations of planning, by enabling the application layer to send a large set of action to the queue, and then let the application layer to return to plan future actions.

5 Discussion and Future Work

The basic functionality described in this paper implements the low-level details of the agent behaviour, such as parsing the information originating from the sensors and triggering the effectors. Several of the key concepts in this module are directly borrowed from Java, such as events and listeners, as well as multi-threading. By using threads in an efficient way, we have overcome the greatest disadvantage of Java, viz. speed.

We will test the performance of our design at IJCAI'99 in Stockholm during the summer of 1999 in competition with several other teams and other designs. Our expectations on the team are that it will be ranked higher than in 1998.

[7] The soccer server is limited to receiving one message of the types *turn*, *dash*, and *kick*, per 100 ms and player.

When the DECIDE research group joined the RoboCup initiative in 1997, one of the main ambitions was to use the RoboCup environment for testing decentralised control using an entity external to the agents for making decisions (Åhman 1998). The decision analysis functionality would be put into a *pronouncer* (Boman and Verhagen 1998), which is a normative entity advising the agents on what to do in a decision situation. Initially the intention was to base the pronouncer on DELTA (Danielson 1997)—a tool for evaluating decision problems where probabilities and utilities can be represented by vague and imprecise statements. DELTA is, however, currently too slow for use in real-time domains such as RoboCup. Work is being done on optimising the underlying algorithms of DELTA, but until that has been accomplished, alternative tools will be considered. A survey of existing tools for decision analysis suitable for use by artificial decision-makers is given in (Younes 1998). The results presented therein indicate that tools do exist that could meet the real-time requirements of the RoboCup domain.

To start with, the pronouncer will be put in the coach-agent. Players on the field will ask the pronouncer for advice concerning comprehensive planning. In the pronouncer, there will be several template models for decision situations that the querying players might be faced with. One could allow agents to provide the pronouncer with an arbitrary decision problem, but that would increase the complexity of the pronouncer and thus degrade its performance making it unusable for real-time decision support. We feel that the use of templates is not a big constraint, since the agents repeatedly find themselves in the same kind of decision situations.

At a later stage, we have plans to put the decision analysis functionality in decision modules, internal to each agent in the team, in addition to using a pronouncer. If model-based reasoning — in contrast to rule-based reasoning (Russell and Norvig 1995) — can be utilised also for short-term decision-making, it could provide a conceptual gain in the design of the agent system.

Acknowledgement

We would like to thank Lisa Brouwers and Karin Hansson for the proof-reading, Johan Sikström for the invaluable discussions and coding on the platform, and Magnus Boman for the excellent supervision and support during the development.

References

Andre, David, Emiel Corten, Klaus Dorer, Pascal Gugenberger, Marius Joldos, Johan Kummeneje, Paul Arthur Navratil, Itsuki Noda, Patrick Riley, Peter Stone, Tomoichi Takahashi, and Tralvex Yeap. *Soccerserver manual ver. 4 rev. 02*. RoboCup Federation, http://www.dsv.su.se/~johank/RoboCup/manual/main.ps.gz, January 1999.

Andreasen, Jens, Magnus Boman, Mats Danielson, Carl-Gustaf Jansson, Johan Kummeneje, Harko Verhagen, Johan Walter, Helena Åberg, Åsa Åhman. UBU: Utility-based uncertainty handling in synthetic soccer. In Minoru Asada, editor, *RoboCup-98: Robot Soccer World Cup II, Proceedings. Of the second RoboCup Workshop*, pages 379-386. RoboCup Federation, July 1998.

Boman, Magnus and Harko Verhagen. Social Intelligence as norm adaptation. In K. Dautenhahn and B. Edmonds, editors, *SAB'98 Workshop on Socially Situated Intelligence*, pages 17-25, Zurich, 1998. Technical Report, Centre for Policy Modelling.

Danielson, Mats. *Computational Decision Analysis*. PhD thesis, Royal Institute of Technology and Stockholm University, May 1997.

Davidsson, Paul. Dumma robotar fungerar bäst. *Forskning och Framsteg*, 6:46-50, 1998, in Swedish.

Engström, Henrik and Johan Kummeneje. DR ABBility: Agent Technology and Process Control. Master's thesis, Department of Computer and Systems Sciences, Stockholm University and the Royal Institute of Technology, December 1997.

Genesereth, Michael R. and Steven P. Ketchpel. Software agents. *Communications of the ACM*, 37(7):48-53, July 1994.

Gosling, James and Henry McGilton. *The Java™ Language Environment*, Sun Microsystems White Paper, May 1996.

Jarry, Alfred. *Ubu Roi*. W.W. Norton & Company, December 1961.

Kitano, Hiroaki, Minoru Asada, Yasou Kuniyoshi, Itsuki Noda, and Eiichi Osawa. RoboCup: The robot world cup initiative. In *Proceedings of ICJAI-95 Workshop on Entertainment and AI/Alife*, 1995.

Langer, Oliver. Soccer-user-library. http://www.tu-chemnitz.de/~hla/soccer/scu/manualscu.ps.gz, May 1997.

Mackworth, Alan. *On Seeing Robots*, chapter 1, pages 1-13. World Scientific Press, 1993.

Noda, Itsuki, Shoji Suzuki, Hitoshi Matsubara, Minoru Asada, Hiroaki Kitano, Overview of RoboCup-97, in Hiroaki Kitano, editor, *RoboCup-97: Robot Soccer World Cup I*, pages 20-41, Springer Verlag, 1997.

Russell, Stuart and Peter Norvig. *Artificial Intelligence – A Modern Approach.* Prentice Hall, 1995.

Stone, Peter. *Layered Learning in Multi-Agent Systems.* PhD thesis, Computer Science Department, School of Computer Science, Carnegie Mellon University, December 1998.

Tambe, Milind, Jafar Adibi, Y. Alonaizon, A. Erdem, Gal A. Kaminka, S C Marsella, Ion Muslea, and Marcello Tallis. ISIS: Using an explicit model of teamwork in RoboCup'97. In *RoboCup'97: Proceedings of the first robot world cup competition and conferences.* Springer Verlag, 1998. Lecture Notes in Computer Science.

Tanenbaum, Andrew S., *Computer Networks*, Prentice Hall, 1996.

Younes, Håkan L. Current tools for assisting intelligent agents in real-time decision making. Master's thesis, Department of Computer and Systems Sciences, the Royal Institute of Technology and Stockholm University, December 1998.

Åberg, Helena. Agent roles in RoboCup teams. Master's thesis, Department of Computer and Systems Sciences, the Royal Institute of Technology, February 1998.

Åhman, Åsa. Decision control in RoboCup teams. Master's thesis, Department of Computer and Systems Sciences, the Royal Institute of Technology, February 1998.