UBU Team

Magnus Boman; Johan Kummeneje; David Lybäck; Håkan L. Younes

In *RoboCup-99 Team Descriptions. Simulation League*, edited by Silvia Coradeschi, Tucker Balch, Gerhard Kraetzschmar, and Peter Stone, 133–138, Stockholm, Sweden. Linköping University Electronic Press.

Author Annotations

An alternate version was later published in the RoboCup-99 proceedings (Kummeneje et al. 2000).

References

Kummeneje, Johan, David Lybäck, Håkan L. S. Younes, and Magnus Boman. 2000. UBU team. In *RoboCup-99: Robot Soccer World Cup III*, 642–645. Springer.

RoboCup-99 Team Descriptions
Simulation League, Team *UBU*, pages 133–138
http://www.ep.liu.se/ea/cis/1999/007/30/

UBU Team

UBU

Magnus Boman, Johan Kummeneje, David Lybäck, Håkan L. Younes

JK, DL, HY: The DECIDE Research Group, Department of Computer and Systems Sciences, Stockholm University
MB: DSV, Stockholm

Abstract. *The underlying research topics and the architecture of the UBU team are briefly described. The aim of developing UBU is to subject a series of tools and procedures for agent decision support to a dynamic real-time domain. The current version of the UBU team is written in Java.*

1 Background

The UBU team first competed in RoboCup98, where it won one game, and lost three [1]. The team being badly tested and not optimised for speed could explain its limited success.

Team UBU originally started out in Java, with parts of a team being implemented in 1997. These parts were then partly used in a new implementation, written in C. As an example of design heterogeneity, we decided to keep the Java goalie, however. This was the version competing in RoboCup98, an event from which we learned very much. Investigations made after RoboCup98 showed that multi-threading in C can be most problematic, and a hard decision had to be made. Should we continue the refinements of the team, or put all our experience in a new team? We decided to start afresh, and use Java throughout. This document reports on this latest generation of UBU.

2 Research Goals

The aim of developing UBU is to subject a series of tools and procedures for agent decision support to a dynamic real-time domain. These tools and procedures have previously been tested in various other domains, e.g., intelligent buildings [3] and social simulations [14]. The harsh time constraints of RoboCup requires true bounded rationality, however, as well as the development of anytime algorithms not called for in less constrained domains (cf. [4]). Artificial decision makers are in the AI and agent communities usually

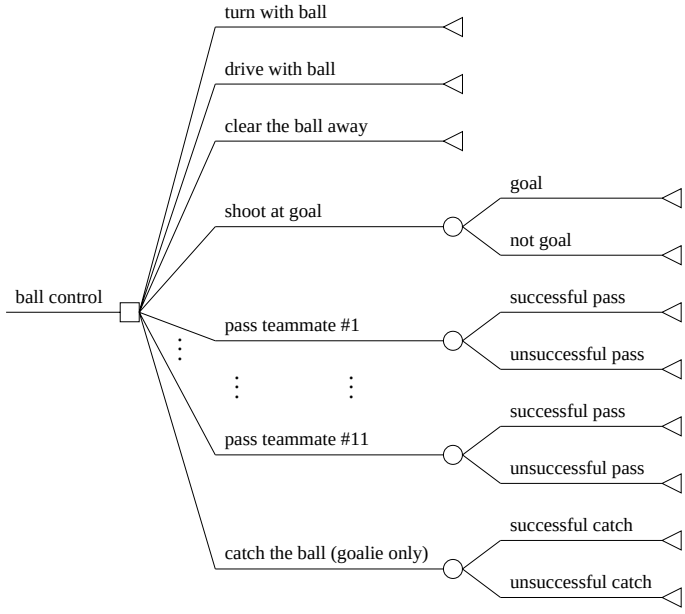

Figure 1: A simple template decision tree.

associated with planning and rational (as in utility maximising) behaviour. We have instead argued for the coupling of the reactive layer directly to decision support. A main hypothesis is that in dynamic domains (such as RoboCup), time for updating plans is insufficient. Basically depending on the size requirements of agents, and on the communication facilities available to the agents, we have placed decision support either in the agents, or externally. In the former case, deliberation is made in a decision module. In the latter case, a kind of external calculator which we have named pronouncer provides rational action alternatives. The input to the pronouncer is decision trees or influence diagrams. The structure and size of these models are kept small, to guarantee fast evaluation (cf. [15]). A typical decision tree is shown in Figure 1. A RoboCup player store empty diagrams as templates and fills them with utilities and probabilities when making queries to the pronouncer. The pronouncer can be made into an agent too, e.g., by using a wrapper. The coach function is particularly interesting in this context, since it is "free" and since it could hold the pronouncer code. An important problem here is the uncertainty and space constraints on the communication with the coach.

The concept of norms as constraints on agent actions have also been investigated [2]. A team in which each boundedly rational player maximises its individual expected utility does not yield the best possible team: Group constraints on actions must be taken into account (see, e.g., [7]). Norms is our way of letting the coalitions that an agent is part of play a part in the deliberation of the agent.

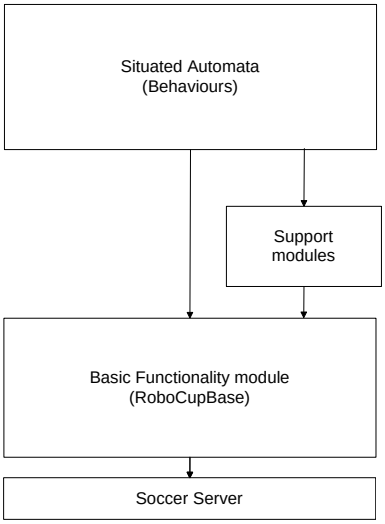

Figure 2: A layered architecture.

3 Architecture

3.1 Overview

Our players are currently rather individualistic, as their explicit co-operative skills are almost non-existing. By providing the players with certain areas which limit their movements, we keep the team from all running for the ball at the same time. The players also try to some extent to predict the behaviour of the other players. The actual design of the team has not been focused on the communication between the players. Instead, a lot of work has been put into the communication within the players themselves. Because of the inherent characteristics of Java, we have designed the system in an object-oriented way to provide it with the highest flexibility and extensibility possible. This has also brought the advantage of portability between platforms, and inspired us into writing the players layered, as illustrated in Figure 2. The main layers of the design are the soccer server, the basic functionality module (RoboCupBase), the support modules, and the situated automata. The latter are designed manually and linked to the various player behaviours (cf. [10]). As the soccer server is provided by the RoboCup Federation, we will not discuss its design in this paper.

3.2 Basic Functionality

The basic functionality implements the low-level details of 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. In each player, we estimate the number of threads to be around 10-15 during a game, and through efficient use of sleep, we have overcome the greatest disadvantage of Java, viz. speed. The architecture of the basic functionality

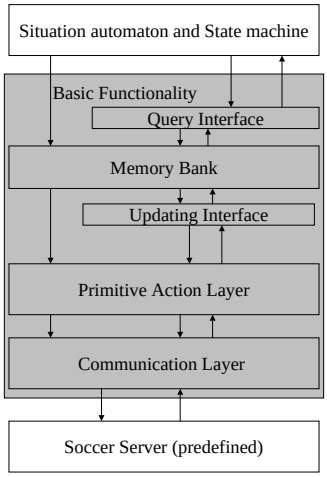

Figure 3: Basic functionality.

module can be seen in Figure 3, where it is shown that there is only one layer (or in this case one class), called the communicator-layer, that interfaces the soccer server (cf. [8]). The communicator-layer defines a set of methods, interfacing two banks in the primitiveaction-layer, viz. the effector bank and the sensor bank. The effector bank translates the action-objects send through the system into the actual messages understood by the soccer server. This is done in parallel with the sensor bank splitting the incoming information from the soccer server into the different senses, e.g., the hear-messages are sent to the hearing-sense.

The next layer is memory-layer, which is basically divided into two parts, the queue of actions, and the short-time memory, i.e. the sphere of attention [6]. The queueing of the actions is simplistic as it only limits the transmission of the time-consuming actions by not allowing them to be executed more than once each 100 milliseconds. In the memory-layer, the interesting part is the short-time memory, which in turn consists of three layers, where we have separated the representation (layer 2) of the information from both the updating (layer 1) and the querying (layer 3). On top of the basic functionality module, we have put a general interface in order for the higher-level programmers to easily access all the functionality needed in the basic functionality module. The main reason for layering the architecture is to enable the future extension of applying this architecture in another league of RoboCup, e.g. the middle-size robot league. Our design is inspired by the OSI-architecture [13], and it has enabled us to reimplement only the two lowest layers to provide the same interfaces upwards, to a different underlying platform.

3.3 Support Modules

We quickly realised that there is a lot of information not provided by the soccer server, e.g., which team has the ball at a given moment. Our solution

is to implement several support modules. At the moment, we have two, which we will describe here. The two are the self model and the team model. It may seem a bit strange to have models in support modules, but they are dynamically providing information independently of the situated automata. Self model is the part where the player has an idea of its role in the team, and of its so-called home position. The team model is a model of the rest of the team, as well as of the opposing team. At the moment, the model calculates the probability for each of the teams having the ball.

3.4 Situated Automata

The situated automata are the tip of the iceberg in our architecture. Basically, an automaton consists of several parts, but it can be looked upon as consisting of two layers, i.e., the reactive behaviour and the deliberative behaviour [5].

We use a combination of reactive and deliberate behaviours, rather than a hybrid approach. We have found that situated automata work well for real-time systems, and we are using them in connection with parts of the subsumption architecture. Some of the behaviours required in the RoboCup environment are mandatory, e.g., correctly interpreting the half-time message from the referee. Therefore, we have implemented reactive responses to such messages, superseding deliberation. As it is a situated automaton, the player is always in a distinct state. Reactive parts are always evaluated first. Thereafter, the deliberative parts are evaluated, and the activity with the highest utility out of all the available activities is performed. Note the difference between activities and actions: we use the term action for the primitive actions, while activity is reserved for a more complex set of conditions and actions. The situated automaton of our architecture strongly resembles the one of the highly successful CMUnited98 [12], something we realised after completing its design.

3.5 Future Research

We are currently investigating social norms as a factor in agent team design. A RoboCup team is an artificial social system in the terminology of [9]. Decision representations, now done through decision trees, will be extended also to include influence diagrams [11]. These will then be evaluated by pronouncers [4].

References

[1] Jens Andreasen, Magnus Boman, Mats Danielson, Carl-Gustaf Jansson, Johan Kummeneje, Harko Verhagen, Johan Walter, Helena Åberg, and 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.

[2] Magnus Boman. Norms in artificial decision making. *Artificial Intelligence and Law*, 7:17–35, 1999.

[3] Magnus Boman, Paul Davidsson, Nikolaos Skarmeas, Keith Clark, and Rune Gustavsson. Energy saving and added customer value in intelligent buildings. In Hyacinth S. Nwana and Divine T. Ndumu, editors, *Proceedings of PAAM98*, pages 505–516. The Practical Application Company, 1998.

[4] Magnus Boman, Paul Davidsson, and Håkan Younes. Artificial decision making under uncertainty in intelligent buildings. In *Proceedings of UAI'99*, 1999. In press.

[5] Rodney A. Brooks. A robust layered control structure for a mobile robot. *IEEE Journal of Robotics and Automation*, 1986.

[6] Henrik Engström and Johan Kummeneje. Dr abbility: Agent technology and process control. Master's thesis, Department of Computer and Systems Sciences, Stockholm University, 1997. DSV Report 97-49-DSV-SU.

[7] Susan Kalenka and Nicholas R. Jennings. Socially responsible decision making by autonomous agents. In *Proceedings of the 5th International Colloquium on Cognitive Science*, 1997.

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

[9] Yoram Moses and Moshe Tennenholtz. Artificial social systems. *Computers and AI*, 1998(?).

[10] Stuart Russell and Peter Norvig. *Artificial Intelligence - A Modern Approach*. Prentice Hall, 1995.

[11] Ross D. Shachter and Mark A. Peot. Decision making using probabilistic inference methods. In *the Eight Conference on Uncertainty in Artificial Intelligence*, San Mateo, CA ,USA, 1992. Morgan Kaufmann.

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

[13] Andrew S. Tanenbaum. *Computer Networks*. Prentice Hall, 1996.

[14] Harko Verhagen. On the learning of norms. Submitted.

[15] Håkan L. Younes. 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, 1998. DSV Report 98-x-073.