Using Simulated RoboCup in Undergraduate Education

Fredrik Heintz[†], Johan Kummeneje[‡], and Paul Scerri[†]

[†] *Department of Computer and Information Science*
Linköping University
SE-581 83 Linköping, Sweden
E-mail: {frehe, pausc}@ida.liu.se
[‡] *Department of Computer and System Sciences*
Stockholm University and the Royal Institute of Technology
Electrum 230, SE-164 40 Kista, Sweden
E-mail: johank@dsv.su.se

Abstract. We argue that RoboCup can be used to improve the teaching of AI in undergraduate education. We give some examples of how AI courses using RoboCup can be implemented using a problem based approach at two different Universities. To reduce the negative aspects found we present a solution, with the aim of easing the burden of grasping the domain of RoboCup for the students, RoboSoc which is a general framework for developing simulated RoboCup agents.

1 Introduction

Several World Cup 1999 teams made extensive use of undergraduate skills and time in their development, often resulting in undergraduates theses [Hei00, Lyb99, Ril99]. Another, larger, group of interested undergraduate students come into contact with RoboCup during normal undergraduate courses. RoboCup has been successfully used in an AI course at Linköping university since 1997 [CM99] and in two courses on agents at Stockholm University with good results during the years 1999-2000. The courses focus on the simulated RoboCup competition because it allows students to study AI and agent technologies in an exciting framework without the expense and expertise required by the real robot leagues. By using a problem based learning approach to courses using RoboCup we argue that RoboCup's educational value can be further increased. However to do this effectively requires a generic library, like RoboSoc, that handles the basic interactions with the simulator, so that the students can focus on the main topic of the course.

2 Using Simulated RoboCup to Teach AI

Using RoboCup in education is generally a positive experience for all involved, albeit with some caveats. One way of improving RoboCup courses is to use problem based learning (PBL) teaching techniques. Although not universally agreed on, for the purposes of this paper we define PBL as *posing questions to be answered or problems to be solved, with the role of the teacher changed to be a coach rather than an expert teacher* [Amb92].

A problem with teaching abstract science is finding illustrative examples for the concepts presented, i.e. finding real problems to solve with the abstract methods provided by the sciences. The prevailing teaching paradigm is based on pedagogy (meaning to teach children). Within this paradigm, the teacher instructs the student on how to solve a problem step by step as well as telling what problems are solvable with this problem-solving method. Thus, the teacher performs the greatest part of the cognitive processing. This is the way most children are taught in first grade.

Andragogy (the teaching of adults) differs from pedagogy in that it is more of a process. In the process the teacher becomes a coach and the students are the driving force. Closely related to andragogy is the concept of PBL. In PBL the student acquires knowledge in a search for solutions to a problem. The problem is formulated with the task being to find suitable solutions to the problem. This contrasts to pedagogy where the task is to first acquire knowledge then, hopefully, find a problem requiring the acquired knowledge.

Since it is not always obvious what knowledge is required, RoboCup is well suited for PBL. We have used PBL in all the courses and found that it increases the interest among the students as they are allowed to formulate their own specific problem and solutions. As RoboCup poses a problem that has no "right" solutions, students are challenged to try to improve on both the ideas they read about and hear from their peers. In RoboCup, AI is mixed with more traditional computer science concepts such as sockets and multi-threading, exposing students to a variety of topics at the same time – which is both good and bad.

PBL decreases the course-specific preparational load but increases the general preparational load for a teacher. For the RoboCup courses, as is common with PBL courses [Amb92], the teachers are or were active researchers in the field hence had already done basic preparational work. Furthermore, as [Amb92] notes in the domain of medical education, PBL is also "fun" for all involved. A teacher is made to look at the area from a new perspective, as the subject is discussed and innovative solutions proposed [Amb92]. The students-teacher discussion constitutes yet another academic conversation, yet often one with a novel angle. This has actually led to the researcher/teacher being aided in his current RoboCup research by teaching a PBL course.

A problem with PBL is that the students usually needs more support during the course. To minimize the extra work for the teachers it is very important to make goals, assignments and expectations on the students as clear and thorough as possible. If the requirements are unclear the students might lose interest in the course or do something totally different than what was expected by the teachers.

3 Description of the Courses

Three AI courses have been held all with RoboCup as a vital element, though with slightly different focus. The courses *Agent Programming I and II* were taught at Stockholm University and the course *AI Programming* was taught at Linköping university.

In *Agent Programming I* [Int00], held Spring 2000, 10 of the 40 participants chose to take the RoboCup based assignment. The aim of the assignment was to let the students learn more about agents and AI by hands on experience. To this end students used an existing team, UBU [KLY99], to design and perform their own experiments. The resulting student reports will be made available to future students as a source of inspiration, for reference and as a launching pad for future experimentation.

Agent Programming II [Int99], held Spring 1999, had 14 participants. The students chose from three tracks, namely Social Aspects of Agents, RoboCup, and Artificial Decision Makers. Six students took the RoboCup-track, three of whom ended up working on UBU, a 1999 World Cup participant [BKLY99]. The other three students researched RoboCup relevant areas, including genetic programming and neural nets.

Since 1997 an AI programming course based on RoboCup has been taught at Linköping university [CM99, AIP]. The course is project based, with students in groups of two or three designing and implementing a RoboCup team. Initially, a series of lectures on relevant topics are given. At the end of the course there is a competition between the student teams and some publicly available teams. Together with a written report, the team's design and performance in the competition constitutes the student's examination.

The Students Perspective From a student's perspective, the major problem with using RoboCup in a course is that building RoboCup agents requires a range of non-AI knowledge, like process programming and networking. Hence a large effort and code is required just to get a simple agent doing things like communicating with the server, parsing the server messages, calculating it's position and sending simple commands at the right time. It is not surprising that the most requested RoboCup course improvement from students has been for more help with the practical problems of developing RoboCup agents [CM99].

To solve these problems, RoboSoc, a framework for developing RoboCup teams was developed [Hei00]. In 1999, the AI Programming course in Linköping used an incomplete version of RoboSoc. In an informal evaluation after the course students reported that RoboSoc improved their ability to focus on the issues they were interested in. The students had less problems creating working teams, of surprisingly high quality, and implementing their own skills and decision making within the RoboSoc framework. There were, however, some concerns about the implementation and documentation of RoboSoc.

At Stockholm University the existing team UBU was made available to the students. The students were very positive about the assignment, generally finding it to be very relevant to the course material, for example, getting insights in the application of neural nets in real-time environments. The students appreciated working on a real software project as opposed to working on an artificial assignment, though some commented that it was a too large a system to grasp in a couple of weeks – especially given the poor documentation. Other students were, however, a little disappointed that they could not implement their own teams from scratch.

4 RoboSoc

For RoboCup beginners there is a need for a well documented, well structured and generic library which helps and encourages developers to create and share their code. Previous attempts to build RoboCup libraries have usually begun with a team releasing parts of their code. The problem with such efforts is the often limited documentation and the released code not being as generic as one would hope, simply because is was not intended to be generic. People attempting to use the released code are too heavily tied to the assumptions and design decisions of original developers. RoboSoc was designed to fill this gap and was subsequently used to develop a World Cup team, rather than the usual opposite ordering. The use of RoboSoc also lowers the amount of extra support needed from the teachers and reduces the risk of the students loosing interest in the course due to too many possibilities and unclear tasks, since it creates a framework within which the students should develop their RoboCup agents.

Students struggle with the practical problems of working with the RoboCup simulator – RoboSoc aims to solve those problems. By using RoboSoc it becomes possible to share solutions of the students with others and make it possible to extend them since the a growing body of students will become familiar with the framework. Using RoboSoc in RoboCup relayed courses lets the students focus on the interesting AI-aspects of the problem. Since RoboSoc has a modular structure, choosing an area of interest and working on that area is possible without affecting the overall system. Trying out different solutions to the problem of interest is thus straightforward. The resulting collection of modules can be made available to other users both in the current year and in the future, allowing the teams to improve over the years. The design means students can go deeply into a single area rather than being forced to learn a little about a lot.

The features of RoboSoc are:

- The interface between the decision maker and the rest of RoboSoc is based on events, which makes it very general.
- RoboSoc handles the interactions with the soccer server, including timing of the agents reactions.
- Modularized world model, with support for sharing, extending, and expressing dependencies between modules.
- Modularized procedural actions, with support for sharing, extending, and expressing dependencies between modules.

The resulting system consists of the three parts described below, the library, the basic system, and the framework shown in figure 1.

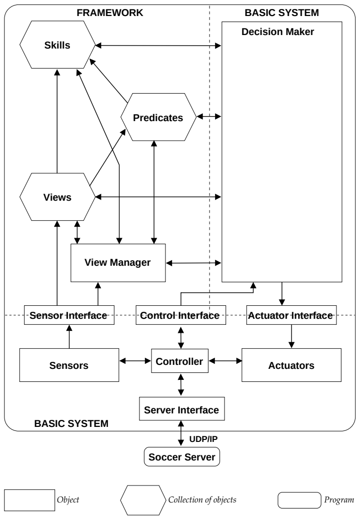

Fig. 1. The RoboSoc architecture.

The RoboSoc library is the foundation for the rest of the system. It provides the necessary types and tools needed by the other parts of the system. There are classes for representing geometric objects, the objects found in the soccer environment and the commands that can be sent to the soccer server. It also defines basic types (integers and floating-point numbers) that can handle unknown values, and uncertainty via the use of probabilities.

The basic system takes care of the interactions with the server, i.e., sending and receiving data. It is also responsible for the timing and provides the basic support for decision making by generating events when the environment changes.

On top of the basic system a framework is designed to support and modularize world modeling and adding support for decision making. By providing building blocks, the user can use existing blocks, extend them with new functionality and create completely new blocks. The framework defines three concepts: *view*, *skill*, and *predicate*. They each encapsulate an important concept used in the information processing and the decision making. A view is a way of looking at information, focusing on some special property or object, from a larger collection of data (which is dependent either on the current time or on the history of the agent). A skill is a complex action, combined of several primitive actions, that will take an agent towards a certain goal state. A predicate can be used either by the decision maker or the skills as a test on the state of the world. The framework depends on both the library and the basic system.

5 Summary

RoboCup is a good domain for demonstrating AI techniques and to let the students get a feeling for what AI is and how it can be used in real applications.

We believe that RoboCup can be used with a traditional style of teaching and we know that it can be used with a problem based learning approach.

As [Kum99, Hei00, Bom99] note, the main problem of using RoboCup in education is the time it takes to come to terms with the peculiarities of the domain, and the focus that is put on improving low level skills, like passing the ball in an optimal way. However RoboCup enables a PBL based learning approach to be used for teaching undergraduate students. We, as have others, have found this way of teaching, especially when RoboCup is used, to be fun and rewarding for both teachers and students. With the development of generic libraries, like RoboSoc, and the use of existing teams some of the negative aspects of using RoboCup and PBL are reduced while the positive aspects are magnified.

References

[AIP] AI-Programming course web-page. `http://www.ida.liu.se/~TDDA14/`. verified 18th July 2000.

[Amb92] George Ambury. Beginning to Tutor Problem-Based Learning: A Qualitative Investigation of Andragogy in Medical Curriculum Innovation. presented at the annual meeting of the CASAE, Ottawa, Canada., June 1992.

[BKLY99] Magnus Boman, Johan Kummeneje, David Lybäck, and Håkan L. Younes. UBU Team. In *RoboCup-99 Team Descriptions*, pages 133–138, 1999.

[Bom99] Magnus Boman. Agent Programming in RoboCup'99. *AgentLink NewsLetter*, (4), November 1999.

[CM99] Silvia Coradeschi and Jacek Malec. How to make a challenging AI course enjoyable using the RoboCup soccer simulation system. In *RoboCup-98: The Second Robot World Cup Soccer Games and Conference*, pages 120–124. Springer verlag, 1999.

[Hei00] Fredrik Heintz. RoboSoc a System for Developing RoboCup Agents for Educational Use. Master's thesis, Department of Computer and Information Science, Linköping university, March 2000.

[Int99] Agent Programming II Course Home Page. `http://www.dsv.su.se/~mab/7v99/Index.html`, 1999. verified 18th July 2000.

[Int00] Agent Oriented Programming Course Home Page. `http://www.dsv.su.se/~mab/4v00/`, 2000. verified 18th July 2000.

[KLY99] Johan Kummeneje, David Lybäck, and Håkan L. Younes. UBU - an object-oriented RoboCup Team. In Johan Kummeneje and Magnus Boman, editors, *Int7 1999 Papers*. 1999.

[Kum99] Johan Kummeneje. Simulated Robotic Soccer and the Use of Sociology in Real Time Mission Critical Systems. In L. R. Welch and M. W. Masters, editors, *Proceedings of RTMCS Workshop*. IEEE, December 1999.

[Lyb99] David Lybäck. Transient Diversity in Multi-Agent Systems. Master's thesis, Department of Computer and Systems Sciences, Stockholm University and the Royal Institute of Technology, September 1999.

[Ril99] Patrick Riley. Classifying adversarial behaviors in a dynamic, inaccessible, multi-agent environment. Technical report, Computer Science, Carnegie Mellon University, 1999. CMU-CS-99-175.