Agency Architecture for Teacher Intervention in Robotics Classes

Ilkka Jormanainen [1], Yuejun Zhang [2], Erkki Sutinen [1], Kinshuk [2]
[1] *Department of Computer Science, University of Joensuu, Finland*
{ijorma, sutinen}@cs.joensuu.fi
[2] *Department of Information Systems, Massey University, New Zealand*
yuejun_zhang@hotmail.com, kinshuk@ieee.org

Abstract

Teachers working in robotics classes face a major problem: how to keep track on individual students' or even small groups' progress in a class of 30-40 students. An agency approach to this problem is based on having sensors to monitor students' interaction, robots' movements, and the construction and programming process of robots. The design can be implemented as Java threads in the Lego Mindstorms set using the IPPE programming environment. The designed architecture works well for monitoring small groups, but needs further work to support teacher's intervention to an individual student's learning process.

1. Introduction

Educational robotics is successfully used for teaching in several school contexts. For example, a technology-rich environment containing educational robotics has been used to enhance learning in special education [13]. Artificial organisms have also been introduced to teaching the design and construction of industrial prototypes to engineers, who often have excellent knowledge of fundamental theoretical concepts, but lack experience in construction [11]. Programming, networking, artificial intelligence and many other topics are taught with robots at the university level.

In a typical classroom setting especially at the elementary level, a teacher might have 30-40 children to teach. However, efficient usage of educational robotics requires new kinds of classroom settings. Also teachers have to change their teaching methods according to the needs of the new environment [5]. When using educational robotics in large classroom settings, students are typically divided into groups of 3-4 students. Each group is provided with one set of educational robotics and group members work together within a given topic or project. This setting is not efficient from the teaching point of view. Groups proceed differently when working with the project, and it might be difficult for the teacher to know when to intervene, whom to intervene and how to make it. The problem can be generalized as follows: *How could the robotics environment inform the teacher what students are doing and how they are progressing?* In particular, the teacher might be interested in:

1. Students' communication
2. Physical interaction with the robots
3. Construction of the robots
4. Programming the behavior of the robots

In this article, we describe an educational robotics design which intends to help the teacher to focus his/her attention in potential problems. The design is based on the Lego Mindstorms robotics kit, which is widely used for educational purposes. The concept applies agent technology to help the teacher to understand the progress of each student group. The software itself will be implemented with Java, based on the LeJOS programming platform for Lego Mindstorms.

This article is organized as follows: Chapter 2 describes the concepts lying behind the implementation of the environment: educational robotics and agency. Chapter 3 introduces the concept and architecture of the agent environment for educational robotics. Finally, we draw conclusions in Chapter 4.

2. Background and existing work

2.1. Educational robotics

In the past few decades, researchers and industries have developed a number of different educational robot kits designed to help learning in scientific fields such as mathematics, physics, engineering, and computer science. These kits typically contain all the components which are needed to construct an autonomous robot: a small computer unit, motors, sensors, wheels, gearwheels, and belts. Autonomous robots can communicate and move independently according to the program the user has constructed.

A well-known example of these tools is the Lego Mindstorms robot kit. The history of Lego Mindstorms goes back to 1986 when a research group supervised by Seymour Papert and Mitchel Resnick started to develop the Programmable Brick, a small unit capable of connecting to the external world through a variety of sensors and actuators [10]. The Mindstorms kit includes an RCX unit (Robotic Command Explorer), an independent computer, which is the core of the kit. An advantage of the RCX unit is its flexibility. The unit can be programmed by using a variety of programming languages such as NQC, Java or Visual Basic. The uses of the Lego Mindstorms kit in different contexts have been widely reported (for example [3], [4]).

The educational principles that motivate the usage of educational robotics are rooted in Jean Piaget's theories of cognitive development [11]. Seymour Papert built on these theories in his notion of 'constructionist learning'. According to constructionist principles, the active learner is the center of the learning process. Learners can create concrete new knowledge and learn in the constructionist way by interacting with real world objects [1]. A typical educational robotics project follows an iterative cycle of building, programming, testing, and evaluation. It is typical that groups proceed differently, being in different phases of the cycle at the same time. This causes difficulty for the teacher to notice the needs for intervention. Our approach is to use educational agents to help the teacher to focus his/her attention in potential problems.

2.2. Agent technologies

2.2.1. General concept. There is no agreement on a unified definition for the term "agent". As Van Dyke Parunak [14] pointed out, there are nearly as many definitions as there are researchers. For some, an agent is a software piece as long as it can travel over a network. For others, an agent represents a module that takes action on behalf of a human user. For still others, agenthood means a certain minimal level of intelligence, or the use of a specified inter-agent language, or the ability to manipulate explicit models of beliefs, desires, and intentions. In this paper, we follow the definition made by Giraffa and Viccari [6] for our special context:

From the hardware viewpoint, an agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors. From the software viewpoint, an agent is substantially a program which has a specific plan of action defined in a limited domain and a behavior pattern which allows it to change at the right moment its own interaction with the world depending on stimuli from the environment.

2.2.2. Pedagogical agents. Pedagogical agents are a class of agents that assist a user in an education-related task. The potentials of agency technologies for education have been well addressed by many researchers, such as [7] and [15]. The former concluded that intelligent agents may perform tasks for both students and teachers in a dynamic learning environment, playing the roles of digital teaching assistants, digital tutors or digital secretaries. The latter further claimed that pedagogical agents can monitor learner progress, give instruction when needed, help organize students' work, and provide feedback for tutors.

According to [2], educational agents can roughly be divided into two categories. One category includes pieces of software, which work invisibly within the system by autonomously dealing with delegated tasks and actively interacting with other entities. The other category presents to users a computer character with human characteristics that facilitate social learning. This category is usually referred to as lifelike or animated pedagogical agents. By using gestures, gaze, facial expressions, locomotion as well as verbal exchanges, animated pedagogical agents may greatly enhance the learning environment.

The agents in our application fall into the first category. In particular, they work on behalf of the teacher to monitor the student progress. Pedagogical agents functioning as teaching assistants like ours are frequently found in literature. One example is the teacher agent in the *I-MIND* project [12]. This teacher agent monitors the student activities and helps the teacher manage and better adapt to the class environment. Another example is the mobile agent developed by Kinshuk, Hong and Patel [9], which facilitates an elegant student modeling in order to provide adaptivity in Web-based learning environments.

3. Applying agents in educational robotics

Unlike most current pedagogical agents which are usually applied to computerized learning environments or simulated virtual realities, the agency technologies described in this paper are used in a traditional classroom, a real physical world, in which students are working on a Lego project. In such a context, it is important for the agents to effectively sense the surrounding events and respond to them with both reactivity and proactivity. The surrounding events include what is happening to the robot and the programming activities in the programming environment. The response means a correct suggestion at the right time to the teacher who needs help on when and how to intervene. This suggestion should be made by the agent autonomously and continuously, without the need for any human interventions.

The general question is: *How could the robotic environment inform the teacher what students are doing and how they are progressing?* To answer this question, we have identified a set of questions related to the teacher's concerns. Table 1 presents these questions and the data which can be used by the software agents to find the answers for the questions as well as sources where the data can be obtained.

Table 1. Data sources and sensors needed for teacher intervention

Data \ Question	Are students talking?	Do they touch the robots?	Do they move the robots?	How fast/often do they move the robots?	Are they constructing robots?	What kind of robots are they building?	Do they programming?	What kind of program are they doing?	Source
Sensor configuration						X			Sensor I/O
Motor configuration						X			Motor I/O
Physical construction						X			
Location of the robot(s)		X	X	X					GPS / RFID
State of the robot		X		X					Motors, sensors
Frequency of pressing buttons		X							Buttons
Order of pressing buttons		X			X				Buttons
Direction of the robot			X	X					Compas s sensor
State of the sensors	X								Sensors
Usage of IPPE environment							X	X	User activity

According to the above ideas, two agent modules for data collection will be implemented. One is embodied into the RCX control unit to detect what is happening to the Lego robot, by sensing and analyzing signals from motors, sensors and buttons. The other inhabits the programming environment, monitoring students' programming activities. From another point of view, our agents are not pure software pieces, but they are combinations of both hardware components and software technologies. The hardware focuses on data collecting, while the software takes care of data processing and upon which, decision making. Existing elements already available in the Lego set may be used to construct most of the hardware part. For the software, however, special programming of the robot and extensions to the programming environment are essential. This will be further discussed in the implementation section.

The logical structure of our agent technology is depicted in Figure 1.

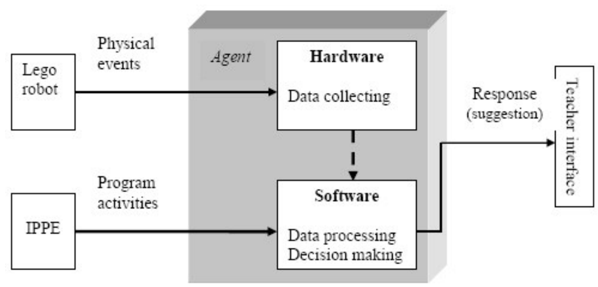

Figure 1. Logical structure of the agent

3.1. Implementation

The design for the agent environment is based on the *IPPE* programming tool. IPPE (Instructive Portable Programming Environment) is a tool for programming Lego Mindstorms robotics with a pseudo-like programming language near to the student's own natural language [8]. The environment has been developed at the University Joensuu since 2001. Programming can be done by composing the student's program with a graphical user interface, or by writing the code straight to the program editor window (Figure 2). Currently, IPPE uses the LeJOS environment [10] to compile and transfer the program code to the robot. The modular structure of the IPPE environment allows developers to write new features to support learning and teaching, such as agency described in this paper.

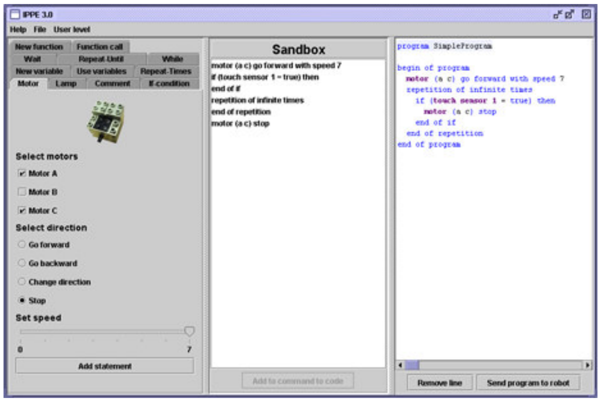

Figure 2. The IPPE environment

To realize the architecture described in this paper, four extensions for the IPPE environment will be developed. The first module implements agency in the robotics environment, that is Lego Mindstorms in this case. The second module implements agency in the IPPE programming environment. The third module works as a network proxy application, and it receives agent data from the robot and the programming environment and delivers the data to the teacher's computer. The fourth module implements visualization in which decisions made by the agents are presented to the teacher. Figure 3 presents the general architecture of the environment. The new modules for IPPE environment have been marked with dotted shapes.

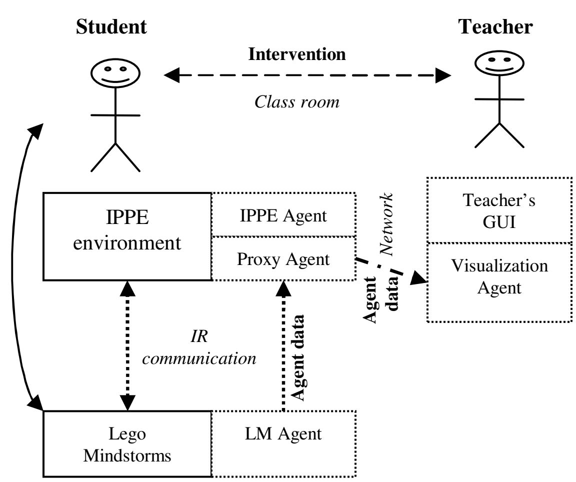

Figure 3. General architecture of the agent environment

The *LM Agent* (Lego Mindstorms Agent) in the robotics environment will be implemented as a Java *thread*. The execution of the thread is started when the user starts running the program. The agent runs in the thread independently, gathering data from the RCX unit and building reactions based on it. The implementation of the agency is transparent. This means that the user does not notice the agent running in the robotics system. The agent is included in the user's program during the compilation process. In this process, the IPPE environment compiles user's pseudo-code to the Java source code. After that, the environment compiles the complete Java source code to Java byte code and uploads it to the robot by using LeJOS environment. After gathering an appropriate set of data, the agent sends its reaction to the *Proxy Agent* running in the user's computer in conjunction with IPPE environment.

The *IPPE Agent* runs in the user's computer as a part of the IPPE programming environment. Again, the agent is implemented as a Java thread, which starts its execution when the user starts the IPPE environment. The agent gathers data related to the user's activity with the programming environment, and after building reactions based on this data, it sends the results to the proxy module.

The *Proxy Agent* is an independent Java program, which collects data coming from the LM and IPPE Agents and sends data collected forward to the teacher's computer through a network socket. The *Visualization Agent* in the teacher's computer presents this data to the teacher. In this way, the system assists the teacher to achieve a better understanding about the progress of the students. It is crucial to note that the system just indicates the progress of the students; it does not assess the results of the learning process. The teacher has to make his/her own decisions concerning the learning outcomes, and the system intends to help the teacher in this task.

3.2 Implementation issues

The implementation of the agent concept presented in this paper has two major tracks.

First, it is necessary to build a framework which allows a reliable communication between agents and other parts of the system. Lego Mindstorms robotics kit uses infrared light (IR) for robots' communication. However, IR communication is very sensitive to the surrounding world, for example changing light conditions may affect the message passing. Also, the obstacles between the IR transmitter and receiver can cause messages to get lost. This means that a strong communication protocol is needed to make sure that the messages containing agent data are passed successfully from the robots to the proxy agent. A partial solution for this is to allow the LM Agent to communicate with all IR receivers in the classroom when sending agent data. In this way, the LM Agent could send data to the teacher through any free Proxy Agent available. Another issue with the framework is the fact that the RCX unit has a limited capacity of computing. Especially the amount of memory is small compared to the modern computers in which agents normally run. This sets limitations for the data which the LM agent collects and analyzes. Also, the thread implementation of the LeJOS platform is less reliable than for example that of the Java Virtual Machine by Sun Microsystem. Thus, a thread implementation of the LM Agent must be planned carefully to avoid any deadlock situation between the agent and the main thread in which the user's program runs.

The second major track in the implementation is the development of the agents. The system contains four different agents (LM Agent, IPPE Agent, Proxy Agent and Visualization Agent). Even though the LM and IPPE Agents have a similar functionality, they can not be implemented in the similar way due to the limitations of the RCX unit. Thread implementation and data storing must be planned carefully especially for the LM Agent. The agents' internal functionality must be carefully planned to endure the fragmented data which sensors in the RCX unit can occasionally produce.

4. Conclusion

We have introduced an agent-based architecture for the educational robotics environment that helps the teacher to intervene into small groups' work, based on the observed student-to-student interaction, robots' movements, and the group's progress in constructing and programming their robots. The design can be implemented for the Lego Mindstorms set with IPPE as the programming environment. Further research is underway to make the environment sensitive to an individual student's progress.

5. References

[1] Ben-Ari, M. (2001). Constructivism in Computer Science Education. *Journal of Computers in Mathematics and Science Teaching* 20(1), 2001, 45–73.

[2] Chou, C. Y., Chan, T. W., Lin, C. J. (2003). Redefining the learning companion: the past, present, and future of educational agents. *Computers & Education*, 40(3), 255-269.

[3] Dagdilelis, V., Sartatzemi, M., Kagani, K. (2005). Teaching (with) robots in secondary schools: some new and not-so-new pedagogical problems. *Proceedings of the 5th IEEE International Conference on Advanced Learning Technology (ICALT 2005)*, IEEE Computer Society Press, Los Alamitos, California, USA, 757-761.

[4] Eronen PJ., Sutinen E., Vesisenaho M., Virnes M. (2002) Kids' Club as an ICT-based learning laboratory. *Informatics in Education*, 1 (1), 61-72.

[5] Fagin, B., Merkle, L. (2003). Measuring the effectiveness of robots in teaching computer science. *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education*. ACM Press, pp. 307–311.

[6] Giraffa, L. M. M. and Viccari, R. M. (1998). The use of agents techniques on intelligent tutoring systems. *XVIII International Conference of the Chilean Society of Computer Science*, 76-83.

[7] Jafari, A. (2002). Conceptualizing Intelligent Agents for Teaching and Learning, *EDCAUSE Quarterly*, 25(3), 28-34.

[8] Jormanainen I., Kannusmäki O., Sutinen E. (2002) IPPE - How to visualize programming with robots. *Proceedings of the Second Program Visualization Workshop*, University of Aarhus, Department of Computer Science, DAIMI PB - 567, 69-73.

[9] Kinshuk, Hong, H., & Patel, A. (2002). Adaptivity through the Use of Mobile Agents in Web-based Student Modelling. *International Journal of E-Learning*, 1(3), 55-64.

[10] Laverdae, D. (Ed.). (2001). *Programming Lego Mindstorms with Java*. Rockland, MA, USA: Syngress Publishing.

[11] Miglino, O., & Lund, H. H. (1999). Robotics as an Educational Tool. *Journal of Interactive Learning Research*, 10(1), 25–47.

[12] Soh, L-K., Jiang, H., Ansorge, C. (2004). Agent-based cooperative learning: a proof-of-concept experiment. *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education*, 368-372.

[13] Sutinen, E., Virmajoki-Tyrväinen, M., Virnes, M. (2005). Physical Learning Objects Can Improve Social Skills in Special Education. In Antikainen A. (ed.), *Transforming a Learning Society: The case of Finland*. Explorationen. Studien zur Erziehungswissenschaft, vol. 49. Peter Lang - European Academic Publishers, ISBN 3-03910-489-6. pp. 117-130.

[14] Van Dyke Parunak, H. (1998). Practical and industrial applications of agent-based systems. Retrieved November 4, 2005 from http://agents.umbc.edu/papers/apps98.pdf

[15] Whatley, J. (2004). Achieving virtual teamwork using software agents. Proceedings of the Fourth International Conference on Networked Learning 2004, 696-702.