J.T. Yao, V.V. Raghvan, G.Y. Wang (Eds.): WSS'04, pp. 37-44, 2004.

Facilitating Web-based Education using Intelligent Agent Technologies

Yang Cao and Jim Greer
ARIES Laboratory
Department of Computer Science, University of Saskatchewan
57 Campus Drive Saskatoon, SK. Canada S7N 5A9
{ yangc@athabascau.ca, jim.greer@usask.ca}

Abstract

Software agents will soon proliferate human organizations, education and society, helping users with information gathering, activity scheduling, email management, and individual and collaborative learning. This paper presents an intelligent Web-based educational system using multi-agent system technology and Web services technology. In this multi-agent architecture, each user is assigned with a personal assistant– software agent. In order to achieve teaching/ learning tasks, humans and agents need an effective way to interact. Two alternative approaches were developed for programmable agents in which a human user can define a set of rules to direct an agent's activities at execution time. This research also investigates concerns over user privacy and system security caused by agent programmability in an web-based interactive learning environment.

1. Introduction

As the demand for access to education grows and an increasing numbers of adults return to universities/colleges for continuing education and training [1], so grows the need for new technologies to facilitate learning. Online teaching and learning provide great opportunities to increase flexibility in time and location of study, in terms of availability of information and resources, synchronous and asynchronous communication and various types of interaction via the World Wide Web.

Agents have become popular additions to an interactive learning environments. In general, an interactive learning environment consists of the teachers and the fellow learners with whom the learner interacts during the learning process; the teachers and learners can be human or artificial companions. Besides teacher/learners, the learning environment also consists of a set of computer-based tools that can be used by the learner (i.e. educational software, communication tools), and the learning material that contains the topics the learner has to learn.

The agent-based approach is suitable for supporting Web-based education since relationships among learners, courses, and instructors last for a considerable period of time [2]. Due to the inherent distributed nature of Web-based learning, a Web-based educational environment can be enhanced by a set of software agents [3][4]. A lot of experimental research has shown that intelligent software agents have great potentials for reducing information workload and for automatically performing many knowledge/labor-intensive tasks for learners and educators [5].

An agent is known as a computer system that is "situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives" [6]. The agent's ability to play the role of a personal assistant arises from its autonomy, reactivity, and pro-activity properties. An agent with such properties could enter into negotiations, acting independently to help achieve the user's goals in an unpredictable environment, and communicate with the user. However, it is also these properties, particularly autonomy that raises significant challenges in human-agent interaction. The agent research community has focused on technologies for constructing autonomous agents and techniques for collaboration among agents. Little attention has been paid to supporting interactions between human and agent.

The issues in human-agent interaction may be more generally described by the following four categories [7]: delegating tasks and authority, instructing agents to act and react, sharing context, and dialogue issues. For

example, questions arise as to how a user can successfully delegate a task to an agent, how agents acquire knowledge needed to understand a particular task and find a way to accomplish it, and how a system can deal with a disagreement between the user and his or her agent. Trust, user privacy and security issues have also become concerns in the design of agent-based systems.

This research has been focused on delegating tasks to an agent in a web-based supported learning and specifically within the bounds of a multi-agent online learning environment named I-Help.

The objectives of this research were to investigate how users behave when given the ability to program their agents, what are the users' concerns about their privacy, how agent-based systems can be built to protect users' privacy, whether the overall performance of the system will be affected with agent programmability, and whether agent programmability be better achieved by adding a full-fledged programming environment (like a rule based expert system shell) to the agent versus by adding a simpler customised and restricted rule system.

2. Background

Animated pedagogical agents [8] have been used in learning environments as artificial trainers. The pedagogical agents are animated characters that guide and encourage learners' study in computer-based learning environments. They interact with learners in a manner simulating the behavior of human tutors that includes a combination of verbal communication and nonverbal gestures. They can express both thoughts and emotions which are significant for human teachers. These pedagogical agents are not only knowledgeable about the topics being taught, but also have knowledge about pedagogical strategies and how to obtain relevant information from available resources such as the World Wide Web. One of the example pedagogical agents is STEVE, a virtual trainer for 3D environments [9]. STEVE can answer questions, monitor students' action, and advise learners when playing the role of a tutor as well as a learner's teammate. It provides more humanlike assistance than previous automated tutors could because of his animated body and interaction in the virtual world with students.

AUTOTUTOR and ATLAS are two other successful tutoring systems. AUTOTUTOR [10] is a fully automated computer tutor that has provided guidance for college students in a computer introductory course. AUTOTUTOR tries to comprehend student contributions and stimulate dialogues to guide students answering deep-reasoning questions. ATLAS [11] is a computer tutor for college physics that focuses on improving students' conceptual knowledge.

As the telecommunication infrastructures and the Internet grow, they provide great facilities for online delivering education and collaborative learning. Online learning is defined as Internet-enabled learning or e-learning, including any use of computers and the Internet to facilitate education [12]. Unlike the traditional distance learning, the success of the new online learning environment is not only just delivering the instructional materials but also providing a collaborative learning environment in the virtual learning community. One of the key elements for successful collaborative learning is peer-to-peer sharing of experiences [13] [14]. This provides a sense of belonging, a sense of feeling part of the community.

In the next section an agent based online collaborative learning environment, I-Help, is introduced, followed by an activities awareness issue in this learning environment.

3. I-Help system

The I-Help [15] system was developed in the Advanced Research in Intelligent Educational System Lab of the Department of computer Science, University of Saskatchewan, Canada. I-Help is designed to provide just in time help for students over the Internet. It is a "peer help" system where the students share their knowledge and exchange information with each other. That means people who receive help also give help [16]. There are two main components in the current I-Help system: the public discussion component and the one-to-one private discussion component.

3.1. Public discussion

Public discussion forums are also known as bulletin boards or newsgroups. In the public discussion forums, learners can post questions, discussion problems of common interest, reply to questions posted by others, read posting and search for posting according to author, concepts, keywords, etc. The public discussion component clusters user discussions around the courses in which they are currently enrolled. All the students who are taking a particular course share the same information including questions and answers within the various course forums. Each question or response to that question is called a posting which consist of a unique posting id and author name, etc.

The information about postings and the users' activities in the Public Discussions such as when a user reads a particular posting, when a user posts to a forum, etc. are recorded in the I-Help database.

3.2. One-to-One private discussion

In the one-to-one private discussion component, conversations are private and restricted to pairs of people. When a learner asks a question, an appropriate helper is recommended by the system. The system will match the student model with the models of other students, to find peers who are more suitable to provide help in a timely fashion. The helper is rated according to several factors, such as the knowledge level, availability, and eagerness to help, etc. Once the helper is selected, the helper and the helpee can start to communicate. The dialogues may be synchronous or asynchronous and many private discussions with different partners can proceed simultaneously.

The I-Help system is built on a multi-agent architecture where each person is augmented with a personal agent who acts on the user's behalf to manage the offering and getting of help. In particular, the personal agents are designed to monitor user activity, and to assist learners in locating help resources (both human helper and electronic help resources). Each personal agent keeps a model of its "owner" and this is used to find the best helpee-helper matches when negotiating help with other agents [17]. The user model information is obtained from the learners' self-assessment of knowledge level of the various topics, from short peer evaluations that occur at the end of a help session, and from monitoring student activities in both parts of the I-Help system. Users' activities which are used to measure student participation in I-Help include whether or not the student is currently or frequently online, how often a student reads/posts a message on the public discussion forum, and how often a student answers or replies questions/messages in the private discussion, etc. An agent negotiates with other agents on behalf of its user using a negotiation mechanism [18].

The Matchmaker agent is a coordinator agent that facilitates finding a best helpee-helper match. Matchmaker maintains profiles of the knowledge and some other characteristics of all the users in the system. Each user is able to change their help preference at any time. The user can specify the knowledge level for the various concepts that are relevant to the courses, the number of discussions he/she would like to process at once, about which topics or whom he/she will not help at all. As well the user can tell his/her agent how much he/she wishes to be paid for offering help and how much she is willing to pay for getting help.

A peer evaluation form is available for a learner to evaluate his/her partner after the help session completes. The evaluation includes whether the helper is helpful and knowledgeable on the topic they are working on. This information is stored in personal agent and maintained by matchmaker who uses it in subsequent matches.

3.3. Awareness issues in the current I-Help

Both I-Help prototypes have been used in computer science courses in the University of Saskatchewan. The students found the I-Help system useful and helpful. Most students responded that "reading postings helped their learning"; most found "answers received useful"; many found that "answering other people's questions helped in their own learning"[3].

I-Help users could send out help requests, read postings, or get replying from helpers. However, the WWW techniques and current version of I-Help do not address the problem of feeling “deaf”, “blind” and “alone” due to the lack of mechanisms to support awareness. In the current system there is no way or efficient manner for a user to know about other persons' presence, availability, willingness to interact, and other events happening in the I-Help environment. However, users want to know what is going on in their virtual community just as they would in any real society. For example, a user may want to know when another user logs in to the system, which posting attracts most of the people, and whether users have read a particular message, etc. These events could be used by agents to determine which person would be a good helper and most likely answer a question in a timely manner. Users could use this information to learn or infer about each other in order to cooperate in the learning community. A preliminary user study on activity awareness in I-Help demonstrates that awareness of other learners’ activities facilitates both individual learning and collaborative learning [19]. Other research [20][21] also show that in an online environment the participants' awareness of each other's activities is a critical feature when trying to build successful communities.

A human-agent interface should potentially be contrived to allow users to program their agents to obtain the activities information. Next section describes the design and implementation of programmable agents in I-Help.

4. System Design and implementation

4.1. Example scenarios

Figure 1 presents some examples of the usage of our Agent programming system. It illustrates how the end user programming environment enables different users to monitor others' activities in the I-Help world.

There are three types of users and they have different intentions using the system. Figure 1 includes one instructor, one tutor and two students. The instructor wants to know about common problems encountered by the students, the tutor wants to know whether there is a new question posted, and the two students need help from the instructor and the tutor. Each user can program his/her agent through a specific user interface about what he/she likes to watch and what action should be taken when a particular event happens. When these events happen, the agents will take appropriate actions according to the preference of their owners.

The student "A" may configure a rule to program his agent to send a special notification to the tutor within a half hour after the tutor reads his particular message in the public discussion forum. The rule looks like:

If *the tutor has read message 19765 within the past 30 minutes*

then *notify tutor with the subject "Can we talk?"*.

The student "B" might configure a rule to program her agent to notify her when the instructor signs in to the system. The rule looks like:

If *the instructor has logged in within the past 2 minutes*

then *notify me with the subject "The Instructor just signed in"*.

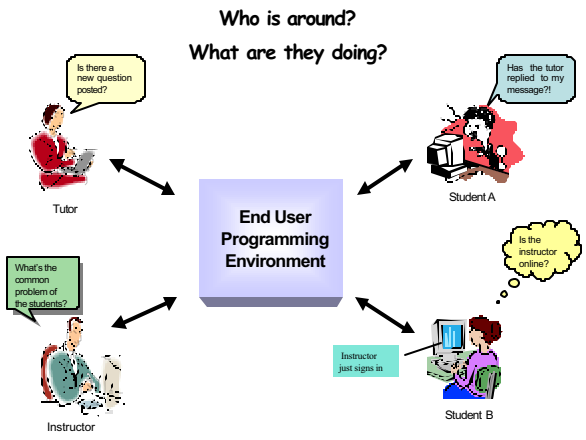

Figure 1. Examples of usage of the end user environment

When the instructor signs in to the system, the system will inform student B. The tutor will get a notification message with the subject "Can we talk? after she reads message 19765.

The users are able to generate complex rules by combining several conditions and actions. See examples in the next section. The users can also write a rule to trigger another rule.

4.2. User Interfaces

In this research, two alternative approaches are employed to build user agent programming environments on top of the I-Help system. One approach is to add to each agent in I-Help a simpler customized rule system, which is called Agent Rule Management System (ARMS). Another approach is implementing CLIPS-based agents that involve connecting a rule based expert system shell to each personal agent in I-Help.

The primary user interface for a user to program his/her agent is the rule management interface which includes one notification signal bar named as Notify, an index frame with the names of the existing rules, and a rule editor frame (Figure 2).

Figure 2. Rule Management Interface

The notification signal (left upper) is used to notify a user when a new notification message is received. When a new notification message arrives, the notification signal bar will turn blue. Figure 2 shows that there is a new notification message for the user. The index frame (left part in Figure 2) enables a user to view the names of all the existing rules, to delete selected rules, to look at a particular rule, and to create

a new rule. The actual generation and modification of the rules are performed in the rule editor (the right part in Figure 2), condition specification, and action specification interfaces. Once the rules are generated, they are displayed as an understandable pseudo-English sentence in the rule editor.

The rule management interface of the CLIPS-based rule environment is similar to Figure 2, which includes one notification signal bar named as Notify, an index frame with the names of the existing rules, and a rule editor frame.

There is a list of rule templates in the index frame of the rule management interface. Similar as ARMS approach, each rule contains three parts: rule name, a condition part, and an action part. A user is able to configure a rule by selecting and filling the value in a template. Figure 3 is a sample rule called loginNotification. The meaning for this template is

When a particular user logged in to the system within the past " 2 " minutes, then create a login notice with the information about his / her login status and send it to me or other users.

A user can specify who will receive the notification message when someone logs in at a particular time by filling the blanks in the templates (see Figure 3). In addition to selecting and filling the value in a template, the users are able to make complex rules by combining several events&actions as well as to define their own rules without using any functions provided by the system.

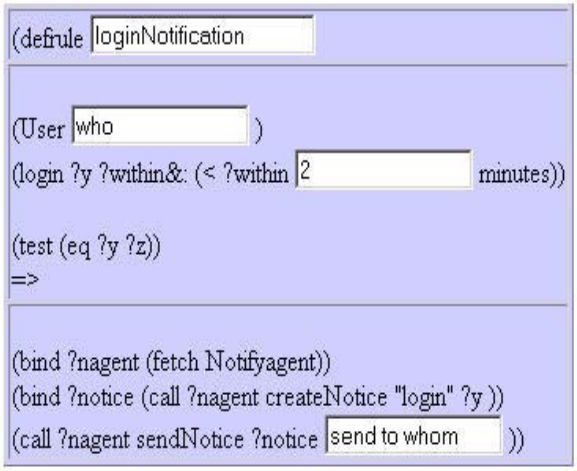

Figure 3. A CLIPS Interface for Login Notification Template

CLIPS permits users to code arbitrary rules to make their agents act in various ways. The full power of CLIPS would allow users to behave in ways that might compromises the system. For this reason, we decided to limit users to making CLIPS rules through template filling. There would be fewer syntax/run time errors when users are filling templates than when they are coding rules for themselves.

4.3. System architecture and implementation

Figure 4 represents a high level architecture for the I-Help end user programmable environment. The Rule Management module, Rule Cycle Detector modules are inside the box of Other Applications.

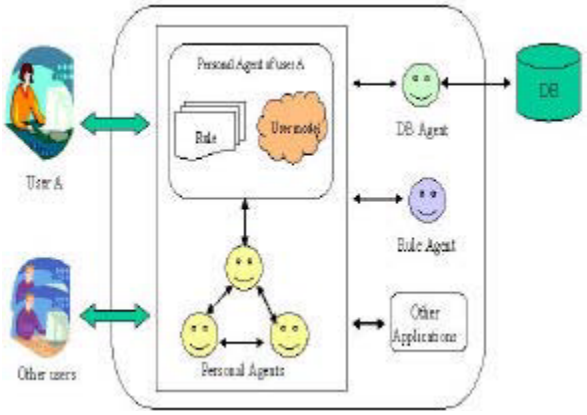

Figure 4. Architecture of I-Help End User Programming Environment

The function of each module is briefly introduced below:

- The main function of a PersonalAgent here is to communicate on behalf of its owner and this is achieved by collaborating with DBAgent, RuleAgent, and other applications. Each personal agent consists of a user model and a set of tasks to be performed. The functionality of the personal agents includes notifying the owner when specified conditions occur, delivering messages to other users or their agents, and responding to messages from other users or agents.
- DB Agent is an application agent that handles all writes and most reads from the database. The information about users' activities are stored and retrieved via DB Agent.
- RuleAgent is an agent that deals with managing the rule repository and detecting interactions among a set of rules.
- The interactions between a user and his/her agent are through a set of user interfaces which are composed of static and dynamically created html pages. Information is sent between the user and agent via Java Servlets.

- Other applications include Rule Management module, Rule Cycle Detector, and other Java components.

More information on structure of rules and system implementation can be found in [22].

5. Evaluations and results

5.1. Usability study on ARMS approach

An experiment on usability of the ARMS approach [22] was conducted with human subjects, to observe the behaviors of subjects during the experiments and analyze the questionnaire and rules generated by the subjects in terms of: Whether the users feel agent programmability is helpful; How easy/hard for a user to configure a rule and how they feel the difficulty of configuration; What kind of rules they would write, what they would watch, what kind of dangers to the system/ users would risk; What were the users' concerns on their privacy including personal information and activity information; and Whether the students had better performance in a learning task with agent programmability than no programmability support.

5.2. Comparative usability study

One of the objectives of this research was to evaluate and compare the strengths and weaknesses of the two approaches technically to see whether the agent programmability would be better achieved by adding a full programming environment CLIPS than a simpler customized rule system ARMS.

A one-hour comparative study on the comparisons of the CLIPS versus ARMS approach was conducted with ten human subjects who were selected from staff from various departments of Athabasca University and students from the University of Alberta. Some staff work in educational media development and some work in the computing centre. These people have experiences with online teaching, educational technology, and online course delivery techniques.

During the experiment, the subjects' activities included attending an introduction session, comparing two approaches, and completing an exit questionnaire on feelings about the system and security and privacy concerns.

In the introduction session, brief information was given on I-Help public discussion and private discussion forums and an explanation was given on how to use the systems. Each subject was given a demonstration of the agent programming environment in I-Help, which described what kind of activities agents can watch, how they can respond, and how to use the system, with both the ARMS and CLIPS user interfaces.

After the introduction, the users compared these two approaches by looking in detail at the interfaces of the CLIPS and ARMS approaches, filling in a form about their opinions on these two approaches, such as which approach is easy or hard to use, which is more or less powerful, and which is more or less secure, etc. Finally the users were required to take part in a structured interview session. During the interview, the author asked the subjects for their responses to a set of general questions, which included how they felt about the usefulness of the I-Help agent programming environment and whether surveillance issue and privacy concern might prevent them from using the system in future.

The questionnaire on system usability and the comparison as well as a record of interview were collected for analysis.

5.3. Result and discussion

We asked similar questions on usability and privacy concerns in the comparative experiment as the ones we did in ARMS usability study. In general the result is encouraging [22]. People would like to watch the login, read message, and send message activities of instructor/tutor and knowledgeable students and group members when they need help, want to discuss a question with others, or during a group discussion. None of the people indicated that they like to watch others just for curiosity. People indicated that security or privacy was not a big concern and it would not prevent them from using the system. However, similarly as the survey in ARMS usability study, concerns were raised by some users when people other than a tutor or a friend was watching them on certain events, such as send message, read message.

The majority of users felt the I-Help programmable agents would be very useful or useful to some extent and they would tend to use the system more than before or as same as now if programmable agents were available. No one said that it was useless or they would stop using it.

Table 1. (Appendix) shows the comparison result of the two approaches.

An interesting observation is that compared to the answers in the first ARMS usability study, more people would feel tempted to try to write some rules that could threaten the system or surveil other users if they had

the power. This may be caused by their belief in the power of the CLIPS approach. In the interview, one user said she was very curious to know/see the power of CLIPS, and she would like to see how she could affect the system by writing her own rules.

6. Conclusions and future work

This paper presents two alternative systems were developed for programmable agents in which a human user can define a set of rules to direct an agent's activities at execution time, such as to communicate with other agents and to monitor the activities of other users and their agents. We reached the following conclusion based on the experiment result: (1) Agent programmability is able to support different users' needs and preferences (i.e. awareness of users activity) in the I-Help world. (2) The provision of agent programmability facilitates the participants accessing necessary resources (human and electronic) in their collaborative learning environment. (3) Agent programmability supports individual and collaborative learning by facilitating information exchange and enhancing communication among students within the virtual learning environment.

This research also provides a platform for investigating concerns over user privacy caused by agent programmability and how an online learning environment can be built to protect users' privacy. The result of the survey on users' privacy shows that people would like to expose more activity information to the public. However different degrees of privacy concern occur in the participants on different kinds of events in the learning environment. There is a desire that the users should have control over their agents to protect their privacy.

The future work for I-Help agent programming environment include making more system events available to users and developing programmable anti-spy agents that will enable a user to program his/her agent to detect surveillance activities of other agents, to notify the user, to take other actions, such as filter / block the information.

It is desirable to integrate the programmable agents with other interactive help facilities or e-learning applications, such as an instant messenger and the course delivery system.

7. References

[1] CIHE: The Council for Industry and Higher Education, Response to the joint consultation document from HEFCE and the Learning and Skills Council (2002), http://www.cihe-uk.com/partnershipsfor.htm

[2] Chan, T-W. (1995), Artificial Agents in Distance Learning, International Journal of Educational Telecommunications, 1(2/3), 263-282.

[3] Greer J., McCalla G., Vassileva J., Deters R., Bull S., and Kettel L. (2001) Lessons Learned in Deploying a Multi-Agent Learning Support System: The I-Help Experience, Proceedings of AIED'2001, San Antonio, 410-421.

[4] Baylor, A. (1999). Intelligent agents as cognitive tools for education. Educational Technology, Volume XXXIX(2), 36-41.

[5]Thaiupathump, C., Bourne, J., and Campbell, J. O. (1999) Intelligent Agents for Online Learning, JALN Vol.3, Issue 2.

[6]Wooldridge, M., and Jennings, N. R. (1995) Intelligent agents: Theory and practice. The Knowledge Engineering Review, 10(2), 115-152.

[7]Dickinson, L.(1998) Human-Agent Communication. http://www.hpl.hp.com/techreports/98/HPL-98-130.pdf

[8]Johnson, W. L., Rickel, W., and Lester, J.C. (2000). Animated Pedagogical Agents: Face-to-Face Interaction in Interactive Learning Environments. International Journal of Artificial Intelligence in Education 11(1), 47-78.

[9]Rickel, J., and Johnson, W. L.(1999) Animated Agents for Procedural Training in Virtual Reality: Perception, Cognition, and Motor Control. Applied Artificial Intelligence 13(4-5), 343-382.

[10]Graesser, A.C., Wiemer-Hastings, K., Wiemer-Hasting, P., Kreuz, R., and the Tutoring Resarch Group.(1999) AUTOTUTOR: A Simulation of a Human Tutor. Journal of Cognitive Systems Research 1(1), 35-51.

[11]Vanlehn, K., Freedman, R., Jordan, P., Murray, C., Osan, R., Ringenberg, M., Rose, C. P., Schulze, K., Shelby, R., Treacy, D., Weinstein, A., and Wintersgill, M. (2000). Fading and Deepening: The Next Steps for ANDES and Other Model-Tracing Tutors. In Intelligent Tutoring systems: Fifth International Conference, ITS 2000, eds. G.Gauthier, C. Frasson, and K. Vanlehn,. Berlin: Springer-Verlag. 474-483.

[12]Downs, S. (1998) The future of online learning. http://www.atl.ualberta.ca/downes/future/home.html

[13]Greer, J., McCalla, G., Cooke, J., Collins, J., Kumar, V., Bishop, A., and Vassileva, J. (1998). The Intelligent HelpDesk: Supporting Peer Help in a University Course, in B.Goettl, H.Halff, C.Redfield, V.Shute (eds.) Intelligent Tutoring Systems, Proceedings ITS'98, San Antonio, Texas, LNCS No1452, Springer Verlag: Berlin. 494-503.

[14]Pressley, M., Wood E., Woloshyn, V, Martin, V., King, A., and Menke, D. (1992) Encouraging mindful use of prior knowledge: Attempting to construct explanatory answers facilitate learning, Educational Psychologist, 27(1), 91-109.

[15]Greer, J., McCalla, G., Cooke, J., Collins, J., Kumar, V., Bishop, A., and Vassileva, J. (1998). The Intelligent

HelpDesk: Supporting Peer Help in a University Course, in B.Goettl, H.Halff, C.Redfield, V.Shute (eds.) Intelligent Tutoring Systems, Proceedings ITS'98, San Antonio, Texas, LNCS No1452, Springer Verlag: Berlin. 494-503.

[16]Greer J., McCalla, G., Cooke, J., Collins, J., Kumar, V., Bishop, A., & Vassileva, J. (2000). Integrating Cognitive Tools for Peer Help in Computers as Cognitive Tools: The Next Generation, Susanne P.Lajoie (Ed.) Mahwah, NJ: Lawrence Erlbaum Publishers, 69-96.

[17]Vasseleva, J. Greer, G. McCalla, R. Deters, D. Zapata, C. Mudgal, S. Grant (1999) A Multi-Agent Approach to the Design of Peer-Help Environments, in Proceedings of AIED'99, Le Mans, France, 38-45.

[18]Mudgal, C., and Vassileva, J. (2000) Multi-agent negotiation to support an economy for online help and tutoring, in Proceedings of ITS'2000, Springer LNCS 1839, 83-92.

[19]Cao, Y., and Greer, J (2003). Supporting Awareness to Facilitate Collaborative Learning in an Online Learning Environment. Proceedings of the Computer -Supported Collaborative Learning (CSCL 2003), Bergen, Norway. 183-187.

[20] Jermann, P., Soller, A., and Muehlenbrock, M. (2001) From mirroring to guiding: A review of the state of art of technology for supporting collaborative learning. In Proceedings of the First European Conference of Computer-supported Collaborative Learning (Euro-CSCL). McLuhan Institute: University of Maastricht.
http://www.mmi.unimaas.nl/euro -csclPapers/197.pdf

[21]Schichter, J.H, Koch, M., and Xu, C. (1998) Awareness-The common link between Groupware and community support system. Community computing and support systems: Social interaction in networked communities. Berling:Springer-Verlag. 77-93.

[22]Cao, Y., and Greer, J.(2003) Agent Programmability in a Multi-Agent Learning Environment. Proceedings of the 11th International Conference on Artificial Intelligence in Education, Sydney, Australia. 8 pp.

Appendix

Table 1. Comparison on ARMS and CLIPS approach

1. How easy to program the agent (configure a rule):

	ARMS (%)	**CLIPS (%)**
Easy to understand/operate without help	70%	20%
It's easy with a little help	30%	20%
It's confusing to understand/operate without help	0	60%
It's very hard to understand/operate even with help	0	0

2. For the tasks that are available in both approaches, the approach which the users were preferred to use:

	ARMS(%)	**CLIPS (%)**
Prefer to use	90%	10%

3. Which approach the users thought have more power:

	ARMS(%)	**CLIPS (%)**
Which has more power	20%	80%

4. How do you feel the risk of the system security or your own privacy?

	ARMS(%)	**CLIPS (%)**
This approach is less secure	20%	60%
This approach is dangerous	0	30%