Lessons from Deploying I-Help

Julita Vassileva, Ralph Deters, Jim Greer, Gord McCalla, Susan Bull, Lori Kettel
University of Saskatchewan
57 Campus Drive
Saskatoon, S7N 5A9 Canada
{jiv, ralph,mccalla, greer}@cs.usask.ca

Abstract

In this paper, we describe the multi-agent infrastructure underlying I-Help, an internet-based learning and peer-help system that was deployed over the last two years with over 1000 students at the University of Saskatchewan. The system contains a variety of learning resources, most prominently, public discussion forums, on-line materials, and a chat-tool for private discussion between peer-learners. In this paper we focus only on the private discussion tool (called also "I-Help one-to-one"). We present some of the lessons learned in developing and deploying this part of the system.

1. Introduction

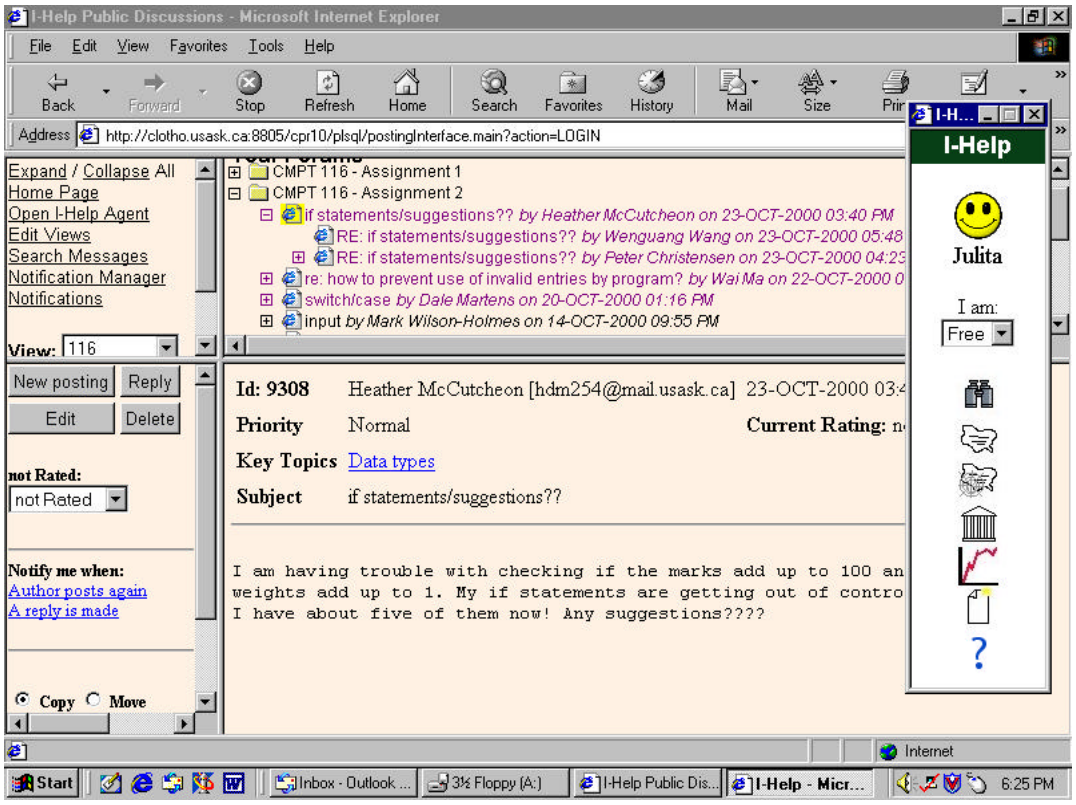
It seems that the future learning environments will be wired and wireless, accessible from anywhere at anytime. Learning in these environments will be distributed in space and time. Multi-agent architectures offer a promising software design approach for such environments, since they are inherently distributed. Through the modularity and uniformity of agents and through a standardized interaction protocol, a level of scalability and interoperation can be achieved that cannot be achieved with other techniques. Multi-agent architectures allow for additiveness and heterogeneity in the software environment. However, currently, there aren't many distributed learning environments based on large - scale multiagent system architectures. Why is this the case? What are the difficulties underlying the development of such environments? We try to provide some answers to these questions using our experience in developing and deploying I-Help, an internet-based peer-help system designed to assist learners as they engage in authentic problem-solving activities.

2. I – Help

I-Help [5] works by locating resources (both electronic materials, like web-pages, discussion forum groups/postings and humans able to help) that are suited to a learner's help request. The I-Help project has explored a number of interesting research issues, especially in the areas of learner modelling and agent technology. In the last two years we have deployed various versions of I-Help in large-scale experiments involving hundreds and sometimes thousands of learners. This has led to many challenges and lessons learned.

2.1 An Example Scenario

To illustrate the functionality of I-Help we will use an example scenario. Imagine that a student working on a programming assignment has a question. She delegates the task of finding help to her personal agent (see Figure 1). The personal agent tries to find another agent (either application agent or another personal agent) which offers information resources related to the help request. These resources can be electronic resources, for example web-pages created by the instructor or other students (and represented by their application agents in the system), or threads / postings in the I-Help discussion forum (represented by the discussion forums application agents). They can be also human resources, i.e. students who are currently on line and competent on the topic of the question (represented in the system by their personal agents). The agents share a common taxonomy for indexing the information resources, which is based on the topics/concepts taught in the class, assignments, etc.. Usually the class instructor creates the taxonomy (using the course outline) when the system is initiated for a given class and it can be expanded later, if necessary. Locating agents that possess information resources is facilitated by matchmaker agents that maintain profiles of the knowledge and some other characteristics of users and applications.



http://clotho.usask.ca:8805/cpr10/plsql/postingInterface.main?action=LOGIN

CMPT 116 - Assignment 1
CMPT 116 - Assignment 2
if statements/suggestions?? *by Heather McCutcheon on 23-OCT-2000 03:40 PM*
RE: if statements/suggestions?? *by Wenguang Wang on 23-OCT-2000 05:48*
RE: if statements/suggestions?? *by Peter Christensen on 23-OCT-2000 04:23*
re: how to prevent use of invalid entries by program? *by Wai Ma on 22-OCT-2000 0*
switch/case *by Dale Martens on 20-OCT-2000 01:16 PM*
input *by Mark Wilson-Holmes on 14-OCT-2000 09:55 PM*

Id: 9308 Heather McCutcheon [hdm254@mail.usask.ca] 23-OCT-2000 03:4
Priority Normal **Current Rating:** n
Key Topics Data types
Subject if statements/suggestions??

```
I am having trouble with checking if the marks add up to 100 an
weights add up to 1. My if statements are getting out of contro
I have about five of them now! Any suggestions????
```

Figure 1: The personal agent (in the right) has found a relevant posting in the public discussion forum related to the help request ("Assignment 2"). The four parts of the window are the interface of the public discussion forum, which is an application participating in I-Help. They allow viewing the postings in a hierarchical way (organized along the topic taxonomy) sending a new posting, replying to a posting, deleting postings, notification for particular postings.

Back to the scenario: if there is no appropriate electronic resource for the question, the matchmaker creates an ordered list of the users who are on line and know the topic and sends it to the personal agent of student who asked for help. The personal agent starts negotiation with the first of the personal agents of the users on the list, trying to find one that would agree to help at a satisfactory price in I-Help credit units (ICUs), the virtual currency of the underlying I-Help economy. Once the negotiation process has succeeded, the agent of the potential helper notifies its user and asks her if she would be willing to help. If not, the personal agent has to negotiate with other agents from the list of suggested helpers. If the student is willing to help, a communication channel is opened between the two users (a simple chat tool at the moment), and a help session starts. After one of the parties closes the chat window, an evaluation form pops up in which the student has to evaluate the other one. This information is used to update the knowledge profiles as well as some of the other characteristics of users maintained by the matchmaker agent.

2.2 Multi-Agent Architecture

I-Help is based on a multi-agent architecture [6], consisting of personal agents (of human users) and application agents (of software applications) (see figure 1). These agents use a common ontology and communication language. Each agent manages specific resources of the user (or application) it represents, including for example, the knowledge resources of the user about certain concepts, or the instructional materials belonging to an application. The agents use their resources to achieve the goals of their users, their own goals, and goals of other agents. Thus all the agents are *autonomous* and *goal-driven*. In their goal pursuit the agents can also use resources borrowed from other agents, i.e. they are *collaborative*. For this they have to negotiate. Each agent possesses a model of its user [2] and of other agents it has encountered and negotiated with [3]. The agents communicate with each other and with matchmaker agents to search for appropriate help resources for their users, depending on the topic of the help-request. If an electronic resource is found (represented by application agents), the personal agent "borrows" the resource and presents it to the user in a browser. However, if a human helper is located, the agents negotiate the price for help, since human help involves inherent costs (time and effort) for the helper. Help is arranged (negotiated) entirely by the personal agents, thus freeing the learner from the need to bargain and think about the currency spent / earned. In this way the personal agents trade the help of their users on a virtual help market. Thus the multi-agent architecture involves various levels of organization, including the negotiation between agents [3], an economical model [1] and control / policing institutions [7]. In this way, we achieve a distributed (multi-user, multi-application) adaptive (self-organized) system that supports users in locating and using help resources (other users, applications, and information) to achieve their goals. In this paper we will focus evolution of the system design, on the deployment of the system and the lessons learned.

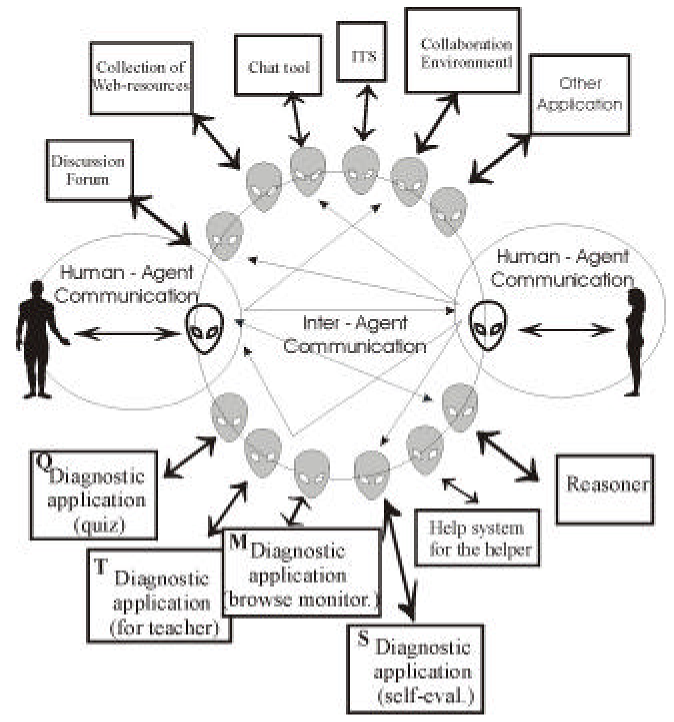

Collection of Web-resources
Chat tool
ITS
Collaboration Environment
Other Application
Discussion Forum
Human - Agent Communication
Human - Agent Communication
Inter - Agent Communication
Q Diagnostic application (quiz)
Reasoner
Help system for the helper
M Diagnostic application (browse monitor.)
T Diagnostic application (for teacher)
S Diagnostic application (self-eval.)

Figure 2: The multi-agent Architecture of I-Help

3. Software Development

Through its various versions the I-Help system has had three basic requirements: to be accessible, dependable and scalable. I-Help has to be widely *accessible* to users. Since it operates in a highly heterogeneous environment, the best solution to the accessibility problem has been to make I-Help available from a web-browser. The http-clients targeted have been Netscape and Internet Explorer.

Dependability is the second requirement. It has been crucial to ensure that the services offered to the students are available, reliable, secure and safe, and that the system doesn't crash. The third requirement is that I-Help is able to *scale up* to allow more and more students to use the system in a wider variety of contexts.

3.1 Proof of Concept Prototype

The first I-Help 1-on-1 "proof of concept" prototype took a single-process server approach. It was written in Java (jdk1.1) and designed to run on a single PC. The application consisted of three modules, a simple communication module (ComServer), an agent host and a module to handle the database connection issues. The agents used in this implementation were simple Java threads that reacted to incoming messages. Small applets embedded in the page ensured a connection of the clients with the application. While all tests indicated a stable system the first real usage ended in a total disaster. The sudden load caused by the simultaneous login of over 60 users within a minute led to a temporary high demand of processor power by the DB-Connection module. This meant that the agents had too little power, which led to slow creation of the web pages. The reaction of the students to the decreased performance was a series of logoff-login commands, which lead to an extremely high load, which, in turn, resulted in total collapse of the application. With this first disappointing experience in mind the students refused to work with improved versions that year. We clearly had to do better if we were to go beyond a proof of concept prototype. In the remainder of this section we will trace the evolution of the I-Help design.

3.2 The First Version

Thus, the version of the I-Help architecture (used in deployment 1) attempted to overcome the problems of resource conflicts by making use of RMI to distribute the server-side application. Each module became an independent process. In addition more complex agents that were able to communicate via KQML messages were introduced. These agents contained simple goal-queues and rudimentary planners. In addition the agents were enabled to observe the current load of the system and to plan their activities accordingly. Using this approach it was discovered that the use of applets led to serious problems (because of different Java versions supported by different browsers and hardware platforms). In addition it turned out that memory leaks (which do happen in Java!) over time led to crashes of the agent host. Monitoring the system and restarting it periodically before memory consumption reached critical levels ensured a minimal degree of stability. Unfortunately, the usage of the system peaked on weekends before assignment-deadlines, which resulted several times in crashes at the time of biggest need. The students reacted to this instability by avoiding the tool.

3.3 The Second and Third Version

The next implementation of the I-Help 1-on-1 architecture (which underpinned both deployments 2 and 3) represented a complete re-implementation of all parts. CORBA was adopted as an object sharing protocol, since it promised the best standard and the easiest way to ensure a scalable system. This version of the system consisted of an agent for each user and a user host, a database connection and servlet engine for communication. The servlets ensured the connection of the clients with the other parts of the implementation and replaced the ComServer. In addition a user host was introduced that was responsible for handling all user data and also served as a cache for user specific web pages. Each module was implemented in a way that one main process (master) controlled various sub-processes e.g. there was one database connection main process, which controlled several database connection sub-processes. This technique ensured scalability by having several agent hosts and database connection processes. By spreading the processes over several machines resource conflicts were avoided. Results showed that this was the first stable version, which was able to serve a large number of users (up to 400). When more than 400 users were given personal agents, the CORBA object-request broker could not support the load. This limited the scalability of the system.

3.4 Current Version

The fourth implementation of the I-Help 1-on-1 architecture (and the version that will underlie deployment 4) is still undergoing tests and improvements. It involves a fully distributed multi-processor implementation with automatic load balancing across many processors. New CORBA brokers are spawned automatically as required. New processes spawn as needed on under-utilised processors. If one processor fails, the entire set of agents that it supported will migrate onto a new processor without interruption. This implementation also offers a rule-based expert system shell on board each agent, permitting the agents to be "programmed" in more flexible ways.

3.5 Lessons

One of the most interesting aspects of the development of I-Help 1-on-1 was the growth of code. The first implementation was 40 KB in size, the second 182 KB, and the third reached 460 KB. The current version, which is still undergoing tests and improvements, is 583 KB at the moment.

We believe, that it would have been completely impossible to achieve the functionality and scalability of the system without a multi-agent architecture. The amount of communications involved in the system makes a load on a centralized component very high. We explored off-the-shelf FIPA-conform agent frameworks, but they turned out to be too limited, involving one process per agent, thus making scalability to thousands of agents an impossible goal. Thus, we created our own multi-agent architecture, and this has proven to be critical to our success in getting 400 distinct personal and application agents working at the same time. In fact, the architecture is an important ingredient to our future plans for this system. As we incorporate more and more I-Help functionality into the multi-agent paradigm, it becomes easier to modify a particular agent's capability and watch its effects on the system.

There is a down side to agents, however. The nature of emergent behaviour resulting from large numbers of interacting, autonomous agents means that any notion of "correct" behaviour is very difficult to define. This suggests there may be no way to predict whether or not a system will scale up without building it first. In fact even after it is built and tested with simulated workloads, it is sometimes hard to predict the kind of workload that real users might apply. Furthermore, simulated workloads that represent realistic situations with multi-user distributed systems are themselves very time-consuming and difficult to build. Often the deployment itself works as the first real load test, so on the first day, when hundreds of students simultaneously start logging on, there is a real risk of a bad surprise.

Another software engineering lesson we learned is that a system in constant evolution must be carefully managed during major deployments. Change management and version control are important issues. There is a great temptation to apply partially tested hot-fixes to code in the running environment. This has caused embarrassment to our developers on many occasions and caused confusion to our users when new features (or new bugs) or subtle changes began to appear without adequate explanation. One of the goals of experimental work with deployed systems is to compare functionality by offering different versions to different sub-groups of users. For example, two different agent negotiation algorithms were being used in deployment 3 of I-Help 1-on-1. The difference in behaviour between the two algorithms would be imperceptible to users, but would provide different candidate helpers for a given situation. Adding this kind of new functionality is relatively simple if the system is well designed. Clearly, version management is crucial in all of these situations.

4. Deployment Lessons

We discuss four deployments of I-Help in classes at the University of Saskatchewan: 1. Sept.-Dec. 1999 (using I-Help v.1); 2. Jan.-Apr. 2000; 3. Sept.-Dec. 2000 (using I-Help v.2) ; 4. Jan.-Apr. 2001 (using the current I-Help version). Deployment 1 of I-Help 1-on-1 used a synchronous chat environment. At that time, I-Help sought the single best helper, according to their knowledge of the topic. The first deployment was available to 100 students, but there was very little usage. The reasons for this are technical (the system was able to support about 50 personal agents; in addition, there were problems with the network speed, as well as social reasons, as will be explained later in the paper.

In deployment 2, synchronous/asynchronous messaging replaced the chat, because the previous version was dependent on the selected helper being online, and willing to engage in the help session. For the same reason, I-Help located the top five potential helpers to increase the likelihood of a quick response. In addition to knowledge level, the learner helpfulness (as evaluated by previous helpees) and eagerness (online activity) were also modelled, and this information was used in matching partners. Learners could also create a 'friends' list – people from whom they would particularly like to receive help, and to whom they would offer a discount in the event that they required help. Users could similarly construct 'banned' lists – people with whom they did not wish to interact. Topics could also be banned. The price for help was calculated centrally by the matchmaker depending on the difficulty of the topic and the amount of knowledgeable users currently on-line. The number of personal agents that could be supported was scaled up to about 200. In deployment 2 I-Help was available to 322 first year computer science students for almost three weeks. Of these, 76 individuals registered to use the system. 62% of them actually used the peer-help search – some extensively; others rarely. There were 86 help requests in total over this three-week period.

In deployment 3, a negotiation mechanism was incorporated in the I-Help personal agents. The system was available to 251 second year engineering students. Of these, only two used the systems few times a week, 13 weekly, 31 seldom, and 205 – very rarely.

In the remainder of this section we discuss the factors and issues influencing the usage of I-Help.

4.1 Technical Factors

Technical factors had a large impact on I-Help usage. One of the reasons for the relatively low level of usage in deployment 1 was the slow response time of the system, especially off campus, due to slow network connections during this period. It must be pointed out that the slow response was due to reasons independent of the system (the local phone company was upgrading the network connection to campus). The coincidence of this maintenance with the introduction of the system to the course was unfortunate. Many students tried to log into the system, after endless waiting tried to log-in again, and when this failed too, they never tried using the system again!

The speed of connection is important, and so is the type / power of computer used. For example, the students using I-Help in deployment 3 usually accessed the system from a lab where the software required for their course assignments (Visual C++) is installed, rather than from their own computers at home. Unfortunately, the lab is using very old and slow computers (Pentium I). Running Visual C++ simultaneously with a browser consumes the processor power entirely, which slows down the performance in both I-Help and the programming environment. These two examples show the critical importance of such “low level” technical factors for the usage of the system.

4.2 Social Factors

A number of social factors affect I-Help usage. The choice of group had a strong influence on the level of use. Often smaller or more cohesive groups do not need the system. The first deployment of I-Help was with 3rd year students who knew each other well, had established multiple ways of interacting in course and in the labs, and hence did not find any need to login to the system. The reasons for this choice were purely pragmatic: time until the beginning of term was short and implementation for this course required the least adaptation effort, as the domain representation and student modelling were already developed. A similar effect appeared in the third deployment, with a large group (3 parallel sections) of second year Engineering students. Due to the culture of the College of Engineering, involving much group work and extra-curricular activities, students knew each other well and had established knowledge networks. They shared laboratory space so there was ready access to face-to-face help. 68% of those who stated in the deployment 3 questionnaires that they had never logged on were from this group. (Other reasons for low use were technical – poor computers in their labs, as mentioned before).

Having knowledge-level differences within a group also encourages I-Help usage. If all the students are at approximately the same level of knowledge, it is less likely that the selection of competent helpers will be effective.

Motivation is another social issue of importance. Our effort to motivate students to offer help led to the introduction of an I-Help economy underlying I-Help. The I-Help economy is intended to create a dynamic help market, which is important not only for encouraging a reasonable level of help requests and help responses, but also for load-balancing among helpers. The main idea is that those who request help have to pay (in I-Help credit units, a virtual currency) and those who give help get paid for the effort. The reasoning mechanism for negotiation [3] incorporated in deployment 3 to facilitate the selling and buying of help allowed a dynamic calculation of the price for help depending on the priorities of the helper and helpee.

Has the economy worked? In deployments 1 and 3 the amount of use of I-Help was minimal, suggesting the economy was not particularly motivating. I-Help was more extensively used in deployment 2, but it is not clear that the economy was the motivating factor. Respondents to the questionnaire administered after deployment 2 were evenly split as to whether they found the virtual currency motivating. Some mentioned that it would be good to be able to exchange the accumulated help-currency for marks towards their final grade in the course. Two people were particularly negative, stating that the currency was stupid! One problem may be that the currency exchange of I-Help credit units into things of value in the real world is not favourable (minimal prizes have been given for top helpers). Another problem may be that rewarding students solely on the level of their bank account does not take into account the quality of the help. It might be important also to take peer evaluations of helpfulness into account, and to see whether users have banned helpers. For example, in deployment 2, one student who was involved in many help sessions in the role of helper (17), left 5 of his helpees with an unanswered question – i.e. he abandoned them during an ongoing discussion.

Finally, perhaps the currency has to be converted into other things than material goods. Several students revealed their main motivation for posting answers on I-Help Pub (a public discussion forum linked which is also a part of I-Help) to be "glory", that they became recognised as “authorities” among their peers. Some students mentioned that they hoped by posting on I-Help Pub to attract the attention of the instructor, another form of recognition. Perhaps I-Help needs to map the currency onto fame and social prestige, as some adaptive web-sites (www.thevines.com, or www.slash.org) already do. In fact, it seems to be generally recognised that social recognition is an efficient reward system also in many newsgroups and on the Internet for the developers of free software [4]. Though our data is inconclusive, we believe that some form of reward is useful to stimulate student participation. The crucial question is the choice of the real world equivalent - the reward should be based on the social values of the group.

The results suggest that I-Help should not necessarily be used by all students. It may not be the most effective method of obtaining help for some. For example, users registered in deployment 2 of I-Help who did not make use of I-Help gave the following reasons: asked friends; preferred face-to-face interaction; preferred working in a small group; asked lab assistant; asked teacher; checked textbook/references; preferred solving own problems; gave help in person; never needed help. These are all valid reasons for not using I-Help.

5. Conclusions

In this paper we outline the multi-agent architecture of I-Help, a distributed Internet based system for peer-help and we focused on discussing the lessons learned from several deployments of the system. As these lessons show, developing a scalable, dependable and usable multi-agent system is a complex endeavor, which can fail in many different ways. However, as virtual learning environments span the web, and physical presence becomes expensive or impossible, we believe that multi-agent environments will be the right way to support distributed learner communities.

Acknowledgement This work has been supported by the Telelearning Network of Centers of Excellence under Project 6.28.

References

[1] Kostuik K., Vassileva J. (1999) Free Market Control for a Multi-Agent Based Peer Help Environment, in Proceedings of the Workshop on Agents for Electronic Commerce and Managing the Internet-Enabled Supply Chain, held in association with the 3rd International Conference on Autonomous Agents (Agents '99), Seattle, May 1-5, 1999.

[2] McCalla, G., Vassileva, J., Greer, J. & Bull, S. Active Learner Modelling, in G. Gauthier, C. Frasson & K. VanLehn (eds), Intelligent Tutoring Systems, Springer-Verlag, Berlin Heidelberg, 2000, 53-62.

[3] Mudgal, C. & Vassileva, J. Bilateral Negotiation with Incomplete and Uncertain Information, in Klusch & Kershberg (eds.) Proceedings of Workshop on Cooperative Information Agents, Springer-Verlag, 2000, 107-118.

[4] Raymond, E. Homesteading the Noosphere, the Cathedral, and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary, O'Reilly and Associates, 1999, available at www.tuxedo.org/~esr/writings.

[5] Vassileva, J., Greer, J., McCalla, G., Deters, R., Zapata, D., Mudgal, C. & Grant, S. A Multi-Agent Design of a Peer-Help Environment, in S. Lajoie & M. Vivet (eds), Artificial Intelligence in Education, IOS Press, Amsterdam, 1999, 38-45.

[6] Vassileva, J. Goal-based Autonomous Social Agents Supporting Adaptation and Teaching in a Distributed Environment, Proceedings of Third International Conference on Intelligent Tutoring Systems, San Antonio, Texas, 1998, 564-573.

[7] Winter, M. (1999) The Role of Trust and Security Mechanisms in an Agent-Based Peer-Help Environment, Autonomous Agents '99, Workshop on Deception, Trust, and Fraud in Agent Societies, Seattle WA, 139-149.