

A Case Study of Agent Programmability in an Online Learning Environment

A Thesis Submitted to the College of
Graduate Studies and Research
In Partial Fulfillment
For the Requirements for the Degree of Master of Science
in the
Department of Computer Science
University of Saskatchewan
Saskatoon

By
Yang Cao

© Copyright Yang Cao, August, 2004. All rights reserved.

PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for Master of Science degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Computer Science

University of Saskatchewan

Saskatoon, Saskatchewan,

S7N 5A9

ABSTRACT

Software agents are well-suited to assisting users with routine, repetitive, and time-consuming tasks in various educational environments. In order to achieve complex tasks effectively, humans and agents sometimes need to work together. However, some issues in human agent interaction have not been solved properly, such as delegation, trust and privacy. 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 humans and agents.

The objectives of this research are to investigate how easy it might be for a user to program his/her agent, how users behave when given the ability to program their agents, whether access to necessary help resources can be improved, and whether such a system can facilitate collaborative learning. Studying users' concerns about their privacy and how an online learning environment can be built to protect users' privacy are also interesting issues to us.

In this thesis 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. The systems were built on top of a multi-agent collaborative learning environment that enables a user to program his or her agent to communicate with other agents and to monitor the activities of other users and their agents. These systems for end user programmable agents were evaluated and compared. The result demonstrated that an end-user programming environment is able to meet users' individual needs on awareness information, facilitate the information exchange among the users, and enhance the communication between users within a virtual learning environment. This research provides a platform for investigating concerns over user privacy caused by agent programmability.

ACKNOWLEDGEMENTS

First I would like to thank my supervisor, Professor Jim Greer, for his guidance and support throughout this study. I really appreciate the encouragements you provided in my study, work, and life and I believe these encouragements will benefit me in the rest of my life. Special thanks to the members of supervisory committee, Professors John Cooke, Julita Vassileva, and the external examiner Len Proctor, for their criticism and suggestions. Thanks as well to the entire faculty, staff and students of the Department of Computer Science at the University of Saskatchewan who provided me with assistance and support throughout this study.

I would also like to thank my husband Hongyu Qiao and daughter Dan Qiao for their understanding, encouragement, and support.

Finally, thanks to the University of Saskatchewan for providing a Graduate Teaching Fellowship to support this study.

TABLE OF CONTENTS

ABSTRACT	ii
ACKNOWLEDGEMENTS	iii
TABLE OF CONTENTS	iv
LIST OF FIGURES	vii
LIST OF TABLES	viii
1. INTRODUCTION	1
1.1 I-Help	1
1.2 Activity Awareness	2
1.3 The Goals of the Thesis	3
2. BACKGROUND	5
2.1 Agent-based System	5
2.1.1 What Is an Agent?.....	5
2.1.2 Why Use agents?.....	6
2.2 Issues in Human-Agent Interaction	8
2.2.1 Delegating Tasks and Authority	8
2.2.2 Instructing Agents to Act and React.....	9
2.2.3 Sharing context	10
2.2.4 Dialogue Issues.....	11
2.3 Approaches for Building Intelligent Agent Systems	12
2.3.1 Knowledge-based Approach.....	12
2.3.2 Machine Learning Approach.....	12
2.3.3 End-user Programming	14
2.4 Agent Programmability	15
2.4.1 Scripting and Form Filling	16
2.4.2 Programming by Demonstration.....	17
2.5 Research on Agent in Learning Environment	18
2.5.1 I-Help System.....	19
2.5.2 Awareness Issues in the Current I-Help System.....	24
2.6 Risk & Privacy Protection	24
2.7 Summary	26

3. DESIGN OF PROGRAMMABLE AGENTS IN I-HELP	28
3.1 I-Help Architecture	28
3.1.1 Logic View of the System Architecture of I-Help	29
3.1.2 Development View of I-Help	32
3.1.3 I-Help Agents	33
3.1.4 Communication among the Agents	34
3.2 Design of Agent Programmability	35
3.2.1 How a User Programs His or Her Agent.....	35
3.2.2 System Architecture	37
3.3 Functionality of the Agent Programming Environment	38
3.3.1 Example Scenarios	39
3.3.2 User Interfaces of ARMS.....	40
3.3.3 User Interface of the CLIPS-based Rule Environment.....	43
4. IMPLEMENTATIONS OF PROGRAMMABLE AGENTS	45
4.1 ARMS Approach.....	45
4.1.1 Design of Rules.....	45
4.1.2 Design of the Cycle Check	48
4.1.3 Structure of ARMS	53
4.2 CLIPS Approach.....	54
4.2.1 Structure of a CLIPS/JESS rule	54
4.2.2 Bottleneck of the Implementation	56
4.2.3 Structure of the CLIPS-Based Agents.....	57
4.2.4 Creating Rules in the CLIPS-based Approach.....	58
4.3 Summary.....	59
5. EVALUATIONS AND RESULTS	60
5.1 The Research Issues and Objectives	60
5.2 Case Studies Design	60
5.3 Usability Study on ARMS Approach.....	61
5.3.1 Experiment Procedure	61
5.3.2 Results.....	62
5.3.3 Discussion	65
5.4 Comparative Usability Study on CLIPS vs. ARMS	66
5.4.1 Experiment Procedure	66
5.4.2 Results.....	67
5.4.3 Discussion	70
5.5 Summary.....	73
6. CONCLUSIONS	74
6.1 Summary of Thesis Work	74

6.2 Research Contributions	75
6.3 Future Research	76
6.4 Conclusion	78
REFERENCES	79
APPENDIX A: Materials Used in the ARMS Usability Study.....	86
APPENDIX B: The Questions in ARMS Usability Study.....	92
APPENDIX C: An Interview Form for First ARMS Usability Study.....	93
APPENDIX D: The Case Study Consent Form for the First ARMS Usability Study.....	98
APPENDIX E: Materials Used in the Comparative Study.....	99
APPENDIX F: An Interview Form for Comparative Study.....	103
APPENDIX G: The Case Study Consent Form for the Comparative Study.....	106

LIST OF FIGURES

Figure 2.1 An agent co-operates with the user on the task	6
Figure 2.2 I-Help Public Discussions Forum.....	20
Figure 2.3 Preferences of Asking and Offering Help	22
Figure 2.4 I-Help Chat Tool.....	23
Figure 3.1 The multi-agent architecture of I-Help	29
Figure 3.2 I-Help System Conceptual Architecture	30
Figure 3.3 I-Help Personal Agent Interface	31
Figure 3.4 DICE Subsystem Conceptual Architecture	32
Figure 3.5 The 5 layers of I-Help private discussion.....	33
Figure 3.6 Architecture of I-Help End User Programming Environment	37
Figure 3.7 Examples of usage of the end user environment	40
Figure 3.8 Rule Management Interface.....	41
Figure 3.9 Condition Specification.....	42
Figure 3.10 Action Specification Interface	43
Figure 3.11 An Interface for Login Notification Template	44
Figure 4.1 The situations of rule a triggers rule b	49
Figure 4.2 Rule Graph.....	50
Figure 4.3 Rule Graph with conditional triggers.....	53
Figure 4.4 Structure of ARMS	54
Figure 4.5 Structure of CLIPS based Agent	58
Figure 4.6 A Sample Rule Template.....	59
Figure 5.1 Performance for each student	63

LIST OF TABLES

Table 2.1 Privacy concerns of Internet users	25
Table 4.1 Formation of the conditions in the rule.....	46
Table 5.1 Result of Questionnaire of ARMS	64
Table 5.2 Result of survey on Login event	67
Table 5.3 Comparison on ARMS and CLIPS approach.....	69
Table 5.4 Results of general questions	70

CHAPTER 1

INTRODUCTION

Software agents will soon proliferate in human organizations, education and society (Greer et al., 2000; Johnson et al., 2000; Payne et al., 2002), helping users with information gathering, activity scheduling, email management, and individual and collaborative learning. 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” (Jennings et al., 1999). 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 issues in human-agent interaction may be more generally described by the following four categories (Dickinson, 1998): 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. 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.

For the purposes of this research, the focus has been on delegating tasks to an agent and specifically within the bounds of a multi-agent learning environment named I-Help.

1.1 I-Help

The IHelp system (Greer et al., 1998) 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 with each other. There are two components in the version of I-Help used here: the public discussion area where students share questions and answers within various course forums, and the one-to-one discussion component where private conversations can be conducted between pairs of people. I-Help is built on a multi-agent architecture where each person in the learning environment is supplied with a personal agent. The personal agents are designed to monitor their user's activity and construct a learner model, and to assist learners in locating useful help resources (human and electronic).

Both I-Help components have been used in computer science courses at the University of Saskatchewan. The students found the I-Help system useful and helpful. 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, and whether users have read a particular message, etc. Unfortunately, neither the WWW nor the current I-Help system supports this degree of awareness of users' activities.

1.2 Activity Awareness

Awareness is the information about other possible collaborators who may be around, whether they are available, what they are doing, and where they are working. In Computer-Supported Collaborative Learning (CSCL), awareness is essential for effective collaboration and it plays an important role in augmenting collaboration opportunities naturally and efficiently (Gutwin et al., 1995). In the physical collaborative learning environment, awareness allows learners to implicitly maintain information about the others' interactions with common problems and corresponding tasks. Ideally the learners can see, hear and even feel the presence and actions of the other. Unfortunately these abilities are hard to achieve in web-based learning environments.

More research has been done to support basic characteristics of group work and awareness (Kurhila, 2002; Ogata et al., 1998; Dieberger, 1997; Munro, et al., 1999; Wexelblat, 1999). However, a number of interesting research questions haven't been solved, such as whether every user should trigger an interaction history, how to deal with different users' needs about awareness, and how to cope with changing interests.

For this thesis research, a human-agent interface was built on top of I-Help, where the users can monitor some events happening in this learning environment by programming their personal agents to watch for particular events or situations. A set of user activities in the I-Help environment (event stream) could be detected by the agents, such as when any user *logs in to* or *logs out from* the system, any user *reads any message* in the discussion forum and any *messages have been sent* between any two users. For example, same events might include "Nancy has just signed in to the system"; "John has posted a question on some public discussion forum"; "George has read a posting in the public discussion forum"; "Nancy has sent a request for help to Fred; or Nancy has just received a message from Susan".

Exposing users' actions or activities may make the system somewhat transparent and the users able to accomplish their respective goals more efficiently. However, while surveillance can be used for good purposes it also can be used for bad purposes. It may result in undesirable consequences and some users may be concerned about their privacy. A user may also wish to know who is watching him/her, what they are watching, and how to protect himself/herself from stalking or inappropriate surveillance.

1.3 The Goals of the Thesis

For the I-Help online learning environment to provide effective and efficient usage, users should be able to acquire knowledge about individuals and events happening in the system. In order to meet users' individual needs, facilitate the information exchange among the users, and enhance the communication between users within the virtual learning environment, an end-user programming environment in I-Help was proposed. Two variations for agent programmability were built on top of the I-Help system to enable users to monitor and analyze the events happening in the I-Help system.

This research was 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, and whether the overall performance of the system will be affected with agent programmability.

The specific questions to be answered by this thesis were:

1. How easy or hard can it be for users to program their agents?
2. Will people feel that agent programmability is helpful?
3. Will users try to destroy the system by malicious use of agents and how can one protect the system?
4. What are the users' concerns about their privacy?
5. Will the capability of I-Help improve with agent programmability?
6. Can 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?

The remainder of this thesis is organized as follows: Chapter 2 provides the background information on agent based system and learning environment. Chapter 3 describes the design of programmable agents in the context of I-Help. The multi-agent architecture of the I-Help system is introduced first, followed by the high level design of the agent programmability, functionality and benefits and risks of the end user programming environment. In chapter 4, two variations of implementation for agent programmability in I-Help are introduced. One variation is to add to each agent in I-Help a simpler customized rule system, which is called Agent Rule Management System (ARMS). Another variation is implementing CLIPS (C Language Integrated Production System) based agents that involve connecting a rule based expert system shell to each personal agent in I-Help. Chapter 5 describes the experiments and results. Chapter 6 gives the conclusions of this research.

CHAPTER 2

BACKGROUND

This chapter explores some of the aspects of agent-based systems, issues in human agent interaction, and some of the ways other researchers have addressed these issues. This chapter also reviews some of agent research in learning environments. The I-Help system that provides a platform for this research is also analysed in more detail.

2.1 Agent-based System

The idea of employing software agents to perform some computer-based tasks for users was first introduced by Nicholas Negroponte (Negroponte, 1970) and Alan Kay (Kay, 1984). Much research has been done in the design and implementation of agents. Examples of some agent-based systems include Meeting Schedule Agent (Kozierok and Maes, 1993), Rcal-RETSINA Calendar Agent (Payne et al., 2002), SwiftFile (Segal and Kephart, 2000), Yenta (Foner, 1997), and online shopping recommendation agent (Haubl and Trifts, 2000). This section reviews some properties of an agent, and the benefits of using agents.

2.1.1 What Is an Agent?

There is no universally accepted definition for the term agent. Actually, there is a great deal of debate and controversy about what an agent exactly is. Wooldridge et al. describe an agent 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” (Wooldridge et al., 1995). In general, an agent has at least some of the following properties: autonomy, reactivity, pro-activity, social ability.

- **Autonomy** refers to the fact that agents can act without the initiative of humans, and they have controls over their actions and internal state.
- **Reactivity** means that agents can sense their environment and respond in time to the changes in the environment.

- **Pro-activity** implies that the agents not only simply respond to their environment, but also take the initiative to pursue goal-directed behavior.
- **Social ability** means that agents can interact with other agents through cooperation and negotiation.

Some researchers, especially those working in AI, generally refer to agents using concepts that are usually applied to humans, such as knowledge, belief, and obligation (Shoham,1993).

2.1.2 Why Use agents?

The significant contribution of software agents is that an agent can act on the user's behalf while the user is doing something else, leading to a significant decrease of human effort in routine work. This is consistent with Maes's notion that the agent behaves as a personal assistant which cooperates with the user on the task. The user is able to "bypass the agent". Figure 2.1 (Maes, 1994) describes the collaborative relationship between a user and his/her agent.

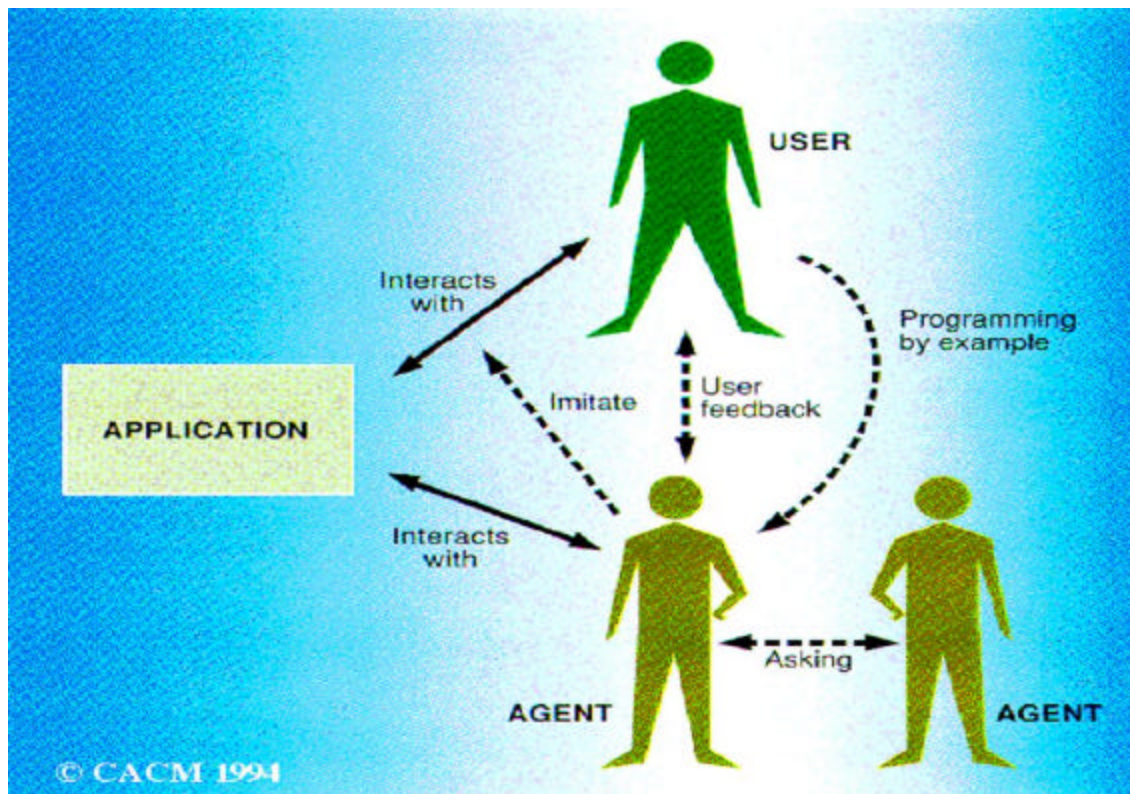


Figure 2.1 An agent co-operates with the user on the task (copyright@ACM)

Computer systems now proliferate in human organizations, information technology, and education society, helping users with information gathering, activity scheduling, email management, entertainment, individual and collaborative learning, etc. They are being used by distributed communities comprising people with varying abilities and needs. Direct manipulation (Shneiderman, 1983), the current dominant metaphor of human computer interaction, requires the user to initiate all tasks explicitly and to take care of sequences of actions in detail. For example, when one copies a file to another folder, first one must open a folder, copy the file, open the destination folder, paste the file, and close all folders. With the agent metaphor, people should be able to communicate with their agents in terms of the work or effects they want to accomplish, rather than being bothered with the specific steps it must take to satisfy their needs. In the previous example, to ask an agent to sort files, probably one would only need to tell the agent to copy a certain file to certain folder. Then the agent will do the detailed work for the user.

Another advantage of using agents is that the agent can work on tedious, repetitive tasks without losing attention and they act and react in situations more quickly than the user could. For example, a web search agent could be tasked with watching for new information on a web site, such as finding some new articles in a particular area for its owner. The search agent will repeatedly visit the site, monitoring arrival of new articles, 24 hours a day, seven days a week without getting tired. Also the speed of agents' reactions is very fast. Agents' reaction times to new events can be measured in milliseconds. The agents are suited to helping users to purchase goods in electronic markets because of this feature as they monitor new events and compare available prices quickly and without losing attention.

The last major advantage is that the agent can adapt to changing circumstances or user preferences. The agent is given a minimum of background knowledge, and it can acquire the knowledge it needs by learning from its user or environment. Maes (Maes, 1994) claims that a learning agent acquires its competence from four different sources. The agent can gradually learn how to act in a new situation either by watching how the user performs tasks, by considering the user's direct and indirect feedback, by receiving

explicit instructions from the user, or by learning from other more experienced agents that perform the same tasks for other users.

2.2 Issues in Human-Agent Interaction

Section 2.1 lists some properties of software agents that enable an agent to act as a broker. An agent with such properties could independently enter into negotiations, help achieve the user's goals in an unpredictable environment, and communicate effectively with the user. However, it is also these properties, particularly autonomy and pro-activity that raise some difficult issues for the agent interface designer. These may be more generally described by the following groups of issues (Dickinson, 1998): delegating tasks and authority, instructing agents to act and react, sharing context, and dialogue issues.

2.2.1 Delegating Tasks and Authority

Delegating a task to an agent means that the agent will act on the user's behalf to perform a task that otherwise would have to be performed by the user. In more detail, it means that the agent accepts a goal from the user, autonomously senses the environment, and acts to pursue this goal.

A trust issue occurs when a user delegates tasks to agents. If the user doesn't know exactly what an agent does in response to a user's goal, a user might be reluctant to delegate a task to the agent. Milewski and Lewis (Milewski and Lewis, 1997) indicate that delegation may not be an easy or a natural behavior for humans. Similarly, Shneiderman (Shneiderman and Maes, 1997) raises the worry that "using agents implies giving up all control". Shneiderman suggests that agent promoters might shift some attention to showing users what is happening so that they can monitor and supervise the performance of agents. Obviously, this approach might compromise the agent's autonomy. Delegation to agents, just like delegation to human, always involves some risks. The risks include that the intentions could be misunderstood and the task might not be performed correctly. But these risks do not stop us from delegating many tasks to people that we work with, and the result is that we are able to do more by collaborating with others (Lieberman and Selker, 2003).

Authority is another issue when users delegate tasks to agents. Different users may want to specify different degrees of authority to the agents, especially when the agent has been given the task of negotiation. For example, in a market some users may like to give an agent authority to decide whether to buy a product at a higher price because its higher quality is better than buying a cheaper one. Other users may give less authority to the agent so the users can have more control in some situations. Sometimes users may not want the agent to have the same authority in different situations. However, the degree of an agent's authority may be encoded at design time by programmers, and thus it may not be possible for a user to vary the agent's authority at runtime.

When an agent is negotiating with other agents, the user may change his/her mind or maybe there is a disagreement between the user and his or her agent. What can a user do in this situation, can he/she interrupt his/her agent or cancel the negotiation? In some cases, an agreement might be reached between the buyer agent and seller agent. If the users don't like an agent's deal, can agents countermand the agreement? Who will be responsible for an agent's actions and transactions? These issues are more social and organizational than technical, but they must be solved in agent-based systems.

2.2.2 Instructing Agents to Act and React

If autonomous agents are to act on their users' behalf, human users must give instructions to them. The issue is how a user instructs his or her agent and how an agent understands the domain specific knowledge.

One solution is that the domain specified knowledge (domain model) should be explicitly or implicitly encoded into the agent's implementation and interface so that the agent will be able to understand and fulfill some tasks in a particular domain. However, a user may delegate a task to an agent using some concept with which the agent is unfamiliar. Alternatively there could be some new concept that arises. In these cases the agent can not understand without the introduction of a new explanation. It requires the designer of the agent to anticipate all possible aspects and interrelationships of the dialogue, and allow sufficient expressiveness in the dialogue for a user to perform the task effectively. Obviously such a demand is unrealistic.

The agents should be able to react appropriately to new situations. This involves characterizing the new situation, and then making a decision about what to do, or what the user wants exactly. This knowledge also depends on input from the user, but it seems unreasonable for a user to provide all the information needed at the beginning of the task. For example, a mail agent which assists the user with email may learn to prioritize, delete, forward, sort and archive mail messages on the behalf of the user. When a message has come from a new person or with an un-interpretable subject, when the user delegates the task to a mail agent the outcome may be unpredictable. Then how can an agent react to it and how can the user tell the agent what to do exactly?

2.2.3 Sharing context

Human beings obtain a large amount of contextual and background information during their daily lives. The information can help people to make decisions, such as recognizing a situation, and deciding what to do in the current situation. Agents acting on a user's behalf must make similar decisions, so they must possess some of this knowledge.

The contextual information consists of two parts: One part is domain-specific background knowledge about the application and user, which are referred as the domain model and user model respectively. The knowledge is relatively stable and usually can be entered into the agent's knowledge base at design and implementation time. Another part is more dynamic, and may vary in different users and situations.

For example, the word “busy” carries different context information in different environments and different times. When in the office, “busy” may mean that one is meeting with someone, reading some papers, writing a technical report, or talking on the phone. However, reading magazines or talking on the phone at home might mean that one is relaxing. Busy-ness is a property of the user's context determined by the user. It is difficult to rigidly define this in a meaningful or convenient way. If one delegates a task to his or her agent, such as “If I am busy, answer some emails or answer a phone for me”, the agent may be confused about the word “busy” and need better context. The agent should share a common context with the user in order to act properly on the user's behalf. The issue is how an agent can share this knowledge with his or her user dynamically.

2.2.4 Dialogue Issues

There are two common categories of human-computer dialogue. The dominant category is: human users take initiative and act as a master to issue commands and queries to the computer, while the computer serves as a slave to perform the tasks and respond to user. Another category is: the computer controls the structure and direction of the dialogue, such as in tutoring system. Mixed-initiative (Novick and Sutton, 1997; Horvitz, 1999; Boicu et al., 2000) has been gradually employed in the dialogues between human and agent. It means that sometimes the user has control, and sometimes the computer (agent) has control.

Besides initiative, other properties of human-agent dialogues need to be addressed, such as: Should agents use facial expressions? What are naturalistic styles for dialogues between human and an intelligent, social agent? What language will be used when a user interacts with his or her agent? Does it have to be spoken or typed natural language? Does it need to show emotional tone in dialogue? The belief that humans will be able to interact with computers in conversational speech has long been “a favorite subject in science fiction, reflecting persistent belief that spoken dialogue would be the most natural and powerful user interface to computers” (Allen et al., 2001). Speech recognition and speech synthesis technologies have been available for nearly two decades. However, due to technical limitations, their application has been limited to a few success stories. Meanwhile, over the same two decades, a revolution in consumer electronics and computing devices has dramatically increased the market need for Spoken Language Interfaces in order to simplify UI and free-up the hands and eyes. Whether anthropomorphism is a good or bad thing has yet to be established.

Finally, culture also affects acceptable and effective dialogue structures. The difficulty for designing international dialogue is not only language, character sets and layout, but also the considerations of different meanings in different cultures. The very basis of the social interface may not be easily applied in different cultures (O’Neill-Brown, 1997). For example, facial expressions may have different meanings in different cultures or in different countries. In most countries, nodding the head means yes and shaking the head

means no, while in some countries, it has very different meaning: nodding means no and shaking means yes.

2.3 Approaches to Building Intelligent Agent Systems

A number of researchers and observers are already concerned about agent construction issues and working on solutions to address them. In this section, different options for building a human-agent interface are introduced along with existing systems built using these techniques.

2.3.1 Knowledge-based Approach

The knowledge-based approach requires designers to encode domain model and user model in advance. In the run-time, the agent uses its knowledge to infer the user's plans and take actions. UCEgo (Chin, 1991) is an interface agent designed according to this approach. The system assists users with the Unix operating system.

The knowledge-based approach is suitable for an application that involves a substantial amount of repetitive behaviour for the users. Unfortunately, since a large amount of application-specific and domain-specific knowledge needs to be entered into the agent's knowledge base, this requires a great deal of work from the knowledge engineer. Another shortage of this approach is that it cannot be readily personalized to individual users' habits and changing goals. There is also a problem with trust. The users might feel some loss of control since they don't know the way an agent "thinks" or works.

2.3.2 Machine Learning Approach

With the machine learning approach, instead of the user or a knowledge engineer explicitly programming the rules, an agent can acquire knowledge by watching the user's behaviour and detecting patterns and regularities. The agent predicts how a new situation should be handled by finding the most similar previously seen situation. Maes (Maes, 1994) claims that a learning agent can gradually learn how to act in a new situation: by watching the user, by considering feedback, by receiving instructions, or by learning from other agents.

The classic learning agent is the mail or news agent that learns to prioritize, delete, forward, and notify a user of incoming mail. For example, MAXIMS (Lashkari et al., 1994) performs all these functions by observing how a user deals with email using the machine learning techniques. The main learning technique used by MAXIMS is memory-based reasoning (Stanfill and Waltz, 1986). The agent memorizes all of the situation-action pairs generated as the user performs actions. When a new situation occurs, the agent compares the new situation with the memorized situations and tries to find close matches. Other systems of this sort include a learning agent MAGI (Payne and Edwards, 1997), and a news-story categorizer (Gustafson, 1998).

Recently electronic profiling became popular in the area of electronic commerce. The technique of learning user preferences in order to build a profile has been used sporadically in autonomous agent development (Libeman, 1997). However, it deserves individual (user) attention in developing an electronic profile. In order to reduce the burden in completing complicated questionnaires, the Apt Decision agent (Shearin and Liebman, 2001) uses an alternative approach. In the rental real estate domain, users provide a small number of criteria in the initial interaction, receive a display of sample apartments, and then react to any feature of any apartment independently, in any order. The agent uses interactive learning techniques to build a profile of user preferences, which can then be saved and used in further retrievals. Because the user's actions in specifying preferences are also used by the agent to create a profile, the result is an agent that builds a profile without redundant or unnecessary effort on the user's part.

The agents discussed above have used learning to anticipate the intentions of an individual user. A class of systems does just the opposite (Maes, 1994; Resnick et al., 1994). They use “social filtering”. Some agents act as learning agents by comparing a user's profiles with other user's profiles and estimating preferences on this basis. The Ringo (Maes, 1994) music-recommendation system is one example. In this system, agents accept recommendations from the other correlated agents (their users have similar musical tastes). For example, if user A and user B have similar musical tastes, and user A found a song is very good which user B has not yet evaluated, then that song is recommended to user B. Context-aware Annotation Proxy-based System (CAPS) (Sharon

et al., 2002) uses collaborative filtering helping a user browsing the web that is based on the experience of the community, i.e., members in the same organization.

The machine learning approach requires less effort from end-users, agent designers and developers. In addition, the learning agent can easily adapt to the user and be customized to individual preferences and habits. However, the basic learning model had several weaknesses: the agent normally has a slow learning curve, therefore it cannot provide useful assistance until it gets enough examples. Furthermore, the agent cannot propose advice in truly new situations, since it would not be able to find any similar examples or precedents.

2.3.3 End-user Programming

In the end-user programming approach, the system has "semi-autonomous agents" which consist of a set of user-programmed rules that contain information about certain tasks. Lai and Malone's Object-Lens system (Lai et al., 1998) is the foundational research using this approach. Object Lens allows users to represent information about people, tasks, products, messages, etc. by defining and modifying templates for various objects. By creating semi-autonomous agents, users can specify rules for automatically processing this information according to their preference. Subsequent systems have been built on this idea, such as SOFTBOT (Etzioni and Weld, 1994), Apple Data Detectors (Nardi et al., 1998), and Stagecast (Smith et al., 2000), the detail of these examples can be found in section 2.4.

The main advantage of this end-user programming approach is that the user can understand what the agent does and how the agent does it. Although the user turns over some control of tasks to the agent, the user can have control back whenever he or she wants. However, it has a number of disadvantages (Lashkari et al., 1994): users have to recognize the opportunity for delegating tasks to an agent, take the initiative to state the rules, specify the rules in textual or graphical languages, and maintain the rules over time as their preference changes.

2.4 Agent Programmability

Many of the routine tasks which users delegate to agents involve relatively complex sequences of behaviours, such as extracting particular information from a web site, monitoring a news group for a recent topic, or performing some set of conditional actions. Users will have specific requirements for their agents about the tasks and the requirements or users' preferences could change from time to time. By analyzing the different approaches for building agents, we found out that a simple knowledge-based approach is not suitable for agent personalization. The agents may have inductive-learning capabilities, however, it is not always possible for agents to perceive user's intent. For example, I may want to be alerted when I get an email message from my boss with the subject "urgent" and I never have similar situation in the past. Therefore, there is no base of examples for a learning approach to work. In addition, the users want their intentions to take effect immediately, so there is no time for agents to accumulate samples and find the pattern. In these cases, agents' actions cannot be learned by observing and must explicitly be requested. The problem of these desires can be solved directly by end-user programming of the intelligent agents.

The end-user programming approach can provide an environment for the users where they can create specific features about which they care and serious users will work around missing features. End-user programming is also an effective approach because it offers significant flexibility to users and it is relatively easy for users to update the rules when their needs change. Some of the issues in human-agent interaction, such as who takes responsibility for the agent's behavior, how to give instruction to agents and the degree to which users can trust their agents, can be solved.

The most commonly encountered instructable agents of this sort are the notification agents found at many web sites. These agents typically use a form-based interface to allow a user to enter an event that he/she wants to be notified about, such as a sale on a particular product. When the event occurs, the agent is triggered, and notification is sent by e-mail.

Available approaches for more end-user programming include high-level scripting languages, forms, direct-manipulation interfaces and programming by demonstration.

2.4.1 Scripting and Form Filling

The script can be seen as a kind of application programming interface (API) that allows users to specify the agent's behavior at a certain level of abstraction. A straightforward approach is to equip the agent with a library of manually authored scripts that determine what the agent might do in a certain situation. At run time, the remaining task is to choose from the library a suitable script that meets the constraints of the current situation and at the same time helps to accomplish a given task.

Unfortunately, the problems with manually authored scripts and script libraries are: the author has to anticipate all possible situations and tasks; the scripts must allow for sufficient variations to avoid agents behaving in a very predictable and similar way; furthermore, the manual scripting of presentation agents can become quite complex and error prone because of synchronization issues. To avoid extensive script writing but, nevertheless, to enable a rich and flexible agent behavior, one can use a generative mechanism that composes scripts according to a set of composition rules.

The PPP (personalized plan-based presenter) Project (Andre and Rist, 1996) addressed the automated generation of instructions for the operation of technical devices. The work of Andre et al. (Andre et al., 1999) formalizes action sequences for composing multimedia material and designing scripts for presenting this material to the user as operators of a planning system. The plan operators allow users to specify spatial and temporal layout constraints for the presentation segments corresponding to the single acts. A planning operator refers to a complex communicative goal (for example, to describe a technical device), whereas the expressions of the body of the operator indicate which acts have to be executed to achieve this goal. The input of the presentation planner might be a complex presentation goal. To accomplish this goal, the planner looks for operators whose headers subsume it. If such an operator is found, all expressions in the body of the operator are set up as new sub-goals.

Using a scripting language requires more effort from the end users, but programming in a scripting language is powerful and the users can develop motivation to automate certain tasks. A notable approach is combining the power of scripting and form filling. SOFTBOT (Etzioni and Weld, 1994) is an example of this approach. The user interacts

through a form-based interface with a planning program that takes the user's goal as input and searches a library of action schema to generate a sequence of actions for achieving the goal. Apple Data Detectors (Nardi et al., 1998) use the similar approach, which works on the user's own machine helping the user to take actions on the structured information found in everyday documents. The user making a selection from the given data achieves the job of supplying data to the parameters of the scripts. Terveen and Murray's Agent Manager system (Terveen and Murray, 1996) works in the domain of message processing. Users instruct their agents to process on a particular message by specifying a set of rule conditions and actions. Users specify rule conditions by restricting the values of the message type, sender, recipient, subject, and the date/time of arrival. Similarly, users select rule actions by choosing from a list of options, such as Move Message, Forward message, Notify by paging, Notify with popup alert, etc.

2.4.2 Programming by Demonstration

Programming by demonstration (Cypher, 1993) also appears a good candidate for instructing agents. Programming by demonstration enables a user to instruct the system to "Watch what I do" and the system to abstract this observed action sequence in order to make it applicable to a wider range of situations. This is similar in spirit to over-the-shoulder learning but "not subject to the pitfalls of inductive learning".

With programming by demonstration, users do not need to learn a conventional programming language or scripting language. A typical example of this class is Stagecast (Smith et al., 2000), a system intended to enable its user to program a kind of videos games. KidSim (Smith et al., 1997) is an example in the programming of animation. KidSim allows children to create their own simulations in which characters move around in a two dimensional world. They create their own characters, and they create rules that specify how the characters are to behave and interact. Bauer et al. (Bauer et al., 2000) also demonstrated a sample programming by demonstration (PbD) dialog aiming at building the wrapper for Personalized Information Services (Bauer and Dengler, 1999). The training dialog starts with the user marking the desired portion of the document. Then the system suggests a number of actions that are expected to increase the estimated

wrapper quality. The user accepts one of these suggestions or decides to do something completely different and so forth until a seemingly good wrapper was produced.

2.5 Research on Agent in Learning Environment

Agents have become popular additions to interactive learning environments. In general, a 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.

Animated pedagogical agents (Johnson et al., 2000) 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 (Rickel and Johnson, 1999). 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 (Rickel and Johnson, 2000).

AUTOTUTOR and ATLAS are two other successful tutoring systems. AUTOTUTOR (Graesser et al., 1999) 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 (VanLehn et al., 2000) 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 (Downes, 1998). 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 (Greer, et al., 1998; Pressley et al., 1992). This provides a sense of belonging, a sense of feeling part of the community. In the following subsections an example of agent based online collaborative learning environment, I-Help, is introduced, followed by an activities awareness issue in this learning environment.

2.5.1 I-Help System

The IHelp (Greer et al., 1998) 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 (Greer et al., 2000). There are two main components in the current I-Help system: the public discussion component and the one-to-one private discussion component.

Public Discussion

Public discussion forums are also known as bulletin boards or newsgroups. In the public discussion forums, learners can post questions, discuss 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.

Figure 2.2 shows an initial interface of the public discussion forum when a user logs into the I-Help Public Discussions. Then the user will be able to perform the tasks we described above, such as read posting or generate a new posting, etc.

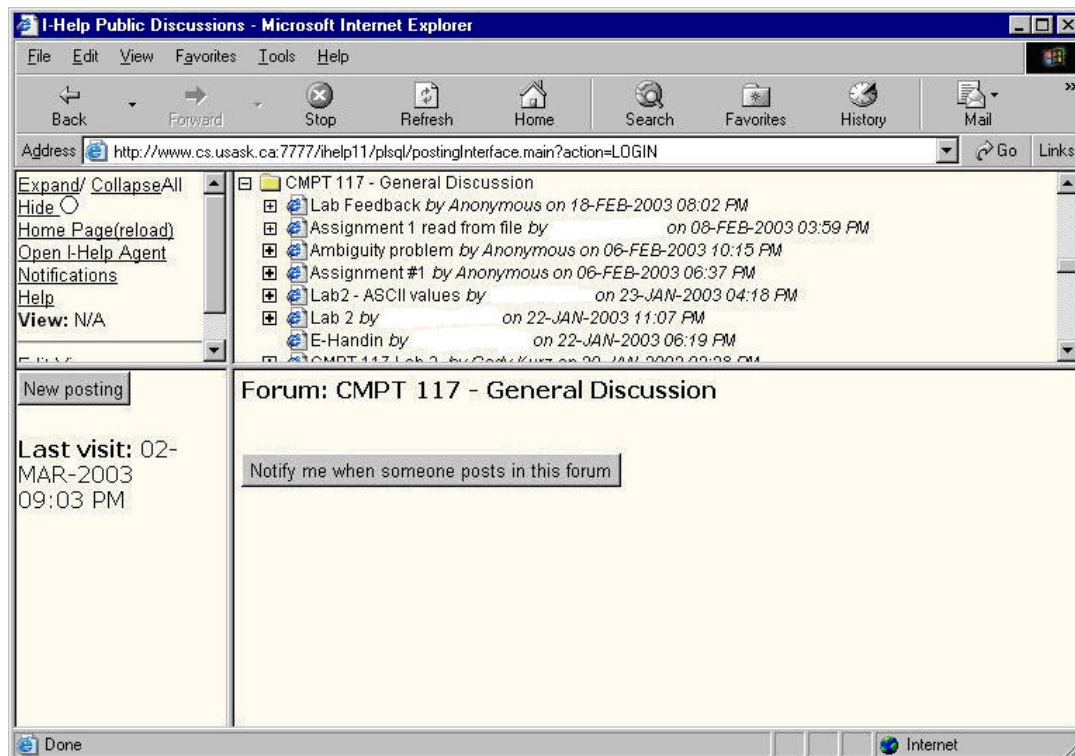


Figure 2.2 I-Help Public Discussions Forum

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 (Vassileva et al., 1999). 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 IHelp system. Users' activities which are used to measure student participation in IHelp 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 (Mudgal and Vassileva, 2000). The agent determines to accept an offer or reject an offer by calculating a utility function with the following factors: money, current goal of the user, the relationship between the users, risk attitude, and perceived utility function and factors of the other agents (Mudgal and Vassileva, 2000).

The Matchmaker agent is an 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. Figure 2.3 shows how an I-Help user can tell the agent about his/her preferences of asking and offering help. 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.

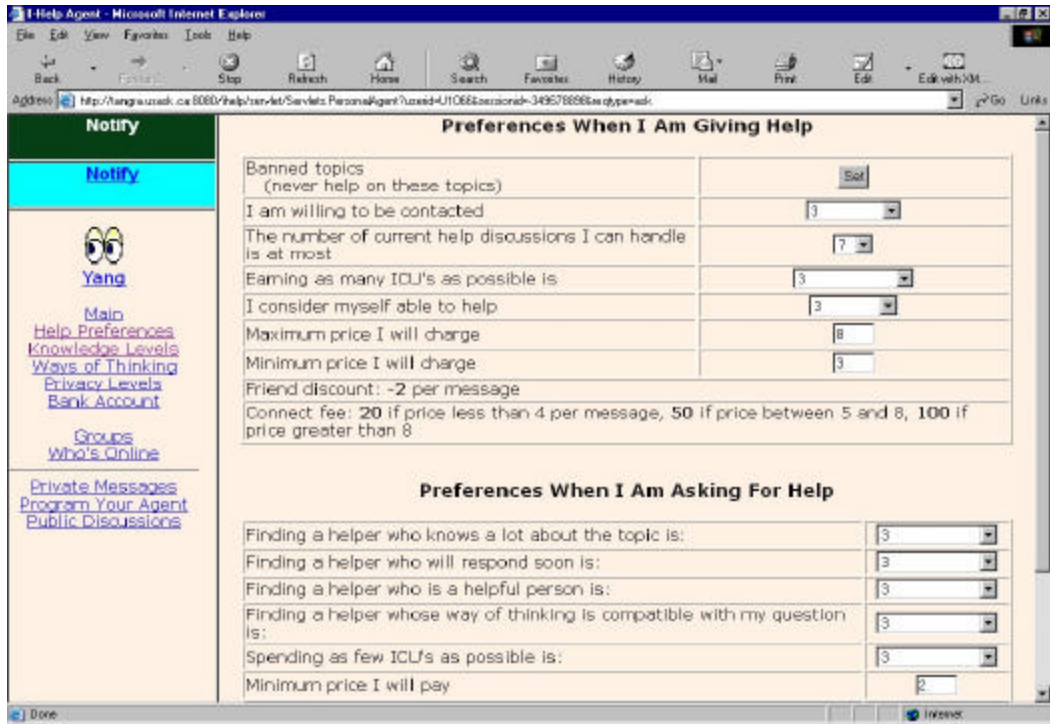


Figure 2.3 Preferences of Asking and Offering Help

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.

In order to illustrate the functionality of I-Help one-to-one private discussions, a scenario is described below. Imagining a first year university student has a question and doesn't know much about the other students. He/she can delegate the task of locating help to his/her personal agent. The following is the sequence of events:

- A learner provides his/her agent with the information about the question, such as question type, topic, and content of the question, etc.
- The agent forwards the information along with the user's help preferences to the matchmaker.

- The matchmaker generates a ranked list of the users who are able to offer help according to users' knowledge level, availability, helpfulness, eagerness to help, and cognitive style, etc.
- The learner's agent negotiates with agents of the potential helpers using its negotiation techniques.
- If the negotiation succeeds, the agent of the potential helper notifies its "owner" that there is a help request waiting.
- Once the helper accepts the request, the two users can start to communicate via a simple chat tool (Figure 2.4).
- After they complete their conversation, each learner is provided with an evaluation form to evaluate their partner.

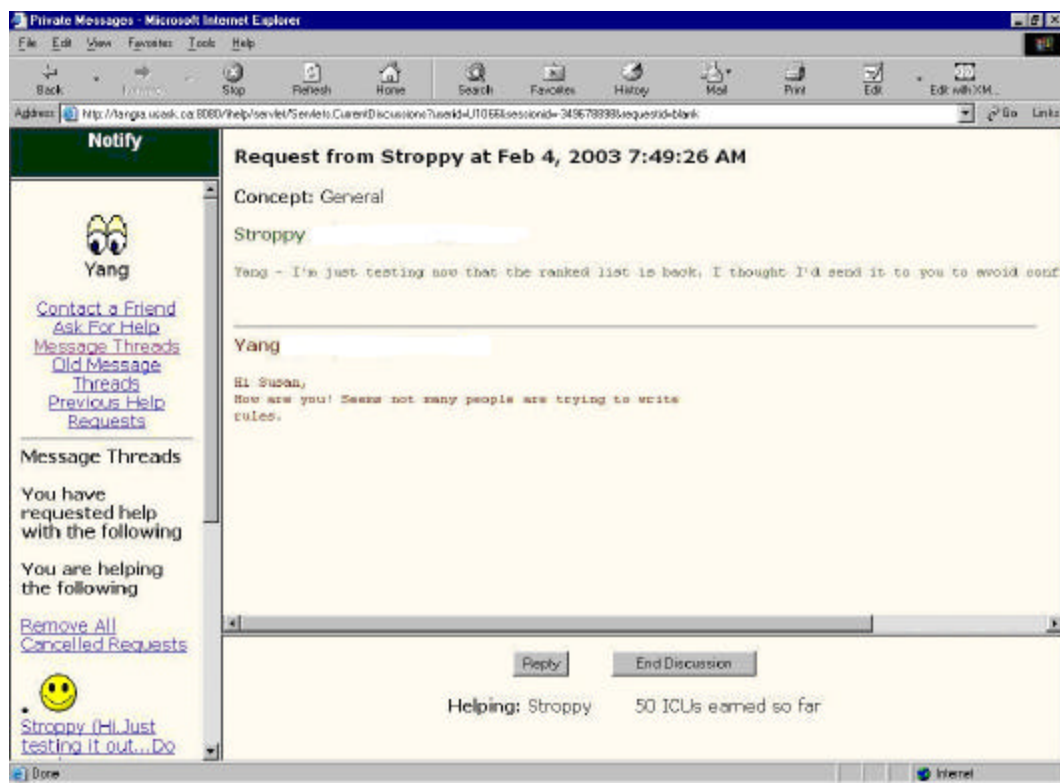


Figure 2.4 I-Help Chat Tool

2.5.2 Awareness Issues in the Current I-Help System

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" (Greer et al., 2001).

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 IHelp demonstrates that awareness of other learners' activities facilitates both individual learning and collaborative learning (Cao and Greer, 2003 a). Other research (Jermann et al., 2001; Schichter et al., 1998) also shows 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.

2.6 Risk & Privacy Protection

We already know the benefit of awareness of others' activities from the previous section. However, while surveillance can be used for good purposes it also can be used for bad purposes. It may result in undesirable consequences such as "cyber-stalking" (Petherick,

1999). In addition, not all the people might like the idea of everyone being able to see everything they are doing in the system. Table 2.1 is an overview of the surveys on privacy concerns of Internet users. The data in table 2.1 shows that the users have concern about their privacy when they are surfing on the Internet.

Table 2.1 Privacy concerns of Internet users

What users are concerned about	Percentage of Users
Being (very) concerned about threats to their privacy when using the Internet	81% (Westin and Maurici, 1998) 87% (Cranor et al., 1999)
Being extremely or very concerned about divulging personal information online	69% (Culnan and Milne, 2001) 74% (AARP, 2000)
Being (extremely) concerned about being tracked online	54% (Pew Internet Project, 2000) 77% (AARP, 2000)

Privacy protection is known as "the right of individuals to control the collection, use and dissemination of their personal information that is held by others" (EPIC, 2000). Many countries and researchers are now developing privacy regulations and privacy protection for electronic data and on-line activities. The Platform for Privacy Preferences Project (P3P, 2000) is an industry standard that facilitates users to gain more control over the use of personal information on Web sites. P3P allows Web sites to publish their privacy policies about how a site handles personal information about its users in a machine-readable format. P3P enables browsers to "read" the privacy policy and compare it to the consumer's own privacy preferences. If the policy and information requested by a Web site matches the user's preferences then the information is sent without disturbing the user, otherwise, the user is notified and given the opportunity to enter the requested information or leave the site.

Privacy critics (Ackerman and Cranor, 1999) are semi-autonomous agents used in combination with P3P sites to help users protect their private information. A critic-base

environment can have hundreds of different critics where each checks on a different facet of a problem domain and user goal. For example, one critic may check the Privacy Consumer Group's site for additional information about this site, another critic may double-check any human-readable text that a site provides to see whether it is the same as the P3P policy.

In I-Help, there are following features of users' information:

- Consist of sensitive personal information like birthday, email, password, etc. and activity information, such as log on to the system, read a posting and so on.
- The activity information including who logs into the system and who posts a message, etc. was obtained by the system by tracking users' activities. The users will use this information when they acquire the information on other's activities. These activities information may be used to facilitate their individual and collaborative learning.

A preliminary user study on activity awareness and privacy concerns in I-Help (Cao and Greer, 2003 a) showed that surveillance of part of the users' activities is not a big concern for most of the learners and they are willing to disclose information about themselves. However, some users raised concerns when people other than a tutor or a friend were watching them on certain events, such as sending messages, and reading messages. Detailed information can be found in Chapter 5.

2.7 Summary

In this chapter, we first broadly reviewed some of the aspects of agent-based systems, the issues in human agent interaction, and different approaches for building agents. Then some agent research and issues in learning environment were explored and I-Help particularly was studied in more detail.

By analyzing the methods of building agents, we found all of them have their own strengths and weaknesses. Some approaches can easily adapt to individual preferences but could not solve users' changing interests very well and quickly. Others may provide more powers for the users but require a big effort from the end users. It should be noted

that the key problems in human agent interaction, namely delegating tasks & authority, instructing agents, and privacy & security, are still unsolved.

An ideal agent based system would accomplish the following:

- Users have control over their agents
- Users are able to instruct their agents easily
- Users are comfortable with their privacy and security
- The system should easily adapt to individual preferences and changing interests

The next chapter describes the design of two agent programming extensions to I-Help that were developed to accomplish these goals.

CHAPTER 3

DESIGN OF PROGRAMMABLE AGENTS IN I-HELP

In this thesis, two alternative approaches are employed to build user agent programming environments on top of the I-Help system. These end user programming environments in I-Help allow users to monitor some events happening in the I-Help learning environment by programming their personal agents. This chapter first introduces the agent platform and multi-agent architecture of the I-Help system, followed by a high level design of the agent programmability, functionality and benefits and risks of the end user programming environment.

3.1 I-Help Agent Architecture

The version of the I-Help system considered in this research was developed by other researchers in the Advanced Research in Intelligent Educational Systems (ARIES) Lab of the Department of Computer Science, University of Saskatchewan. It was built with a multi-agent architecture. In this architecture there are two types of agents: personal agents and application agents, which monitor user activity and assist learners in locating help resources (human and electronic).

Figure 3.1, adopted from (Vassileva et al., 1999), illustrates the organization of I-Help. There is one personal agent for each user and one application agent for each type of application. The I-Help system is implemented in Java and the agents communicate by sending KQML-like messages via CORBA. The various agents can be hosted on the same or different servers and simply register themselves on the CORBA name server to communicate with each other. The following sections explore the different views of the system architecture, the various types of I-Help agents and the communication between agents. "4+1 views" (Kruchten, 1995) is one of the approaches used by software engineers in analyzing and maintaining big systems. "4+1 views" explores the architecture of the big systems from different points of view, which include logic view, process view, physical view, development view and scenarios. In the following section, two of "4+1" views: logical view and development view are used to explain the I-Help architecture.

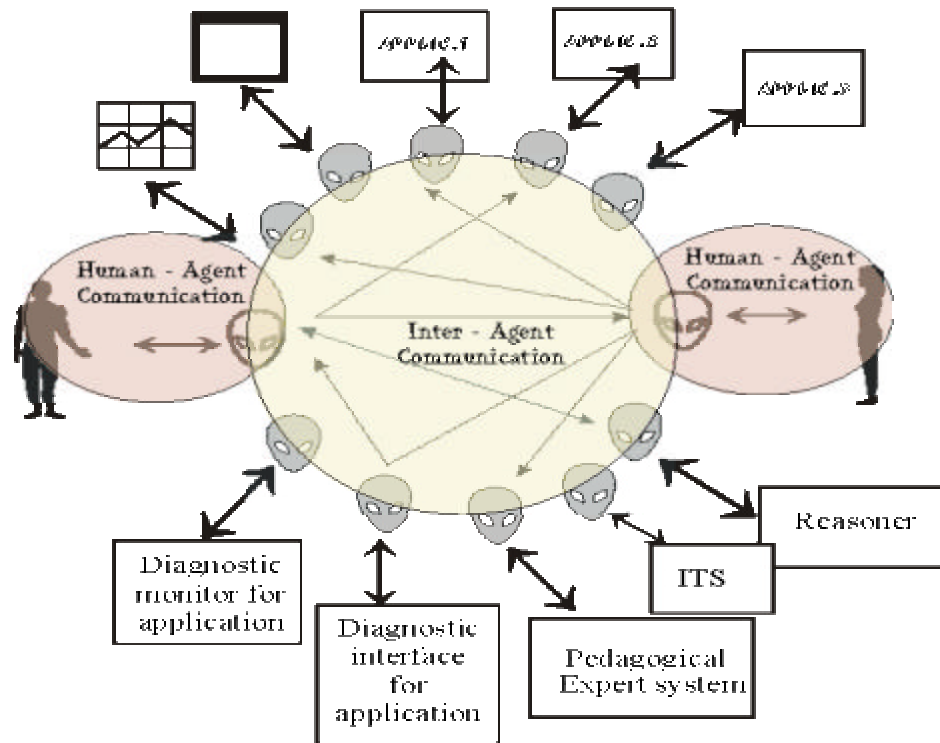


Figure 3. 1 The multi-agent architecture of I-Help

(reprinted with permission from Vassileva et al.,1999)

The grey faces represent application agents, the white ones represent personal agents. Applications are shown as rectangular boxes.

3.1.1 Logic View of the System Architecture of I-Help

Based on knowledge of distributed systems and agent based systems and the existing I-Help documentation and source code, we arrived at the conceptual architecture shown at its highest level of abstraction in Figure 3.2. The diagram is drawn using the UML standard. The four major subsystems are described in the following paragraphs:

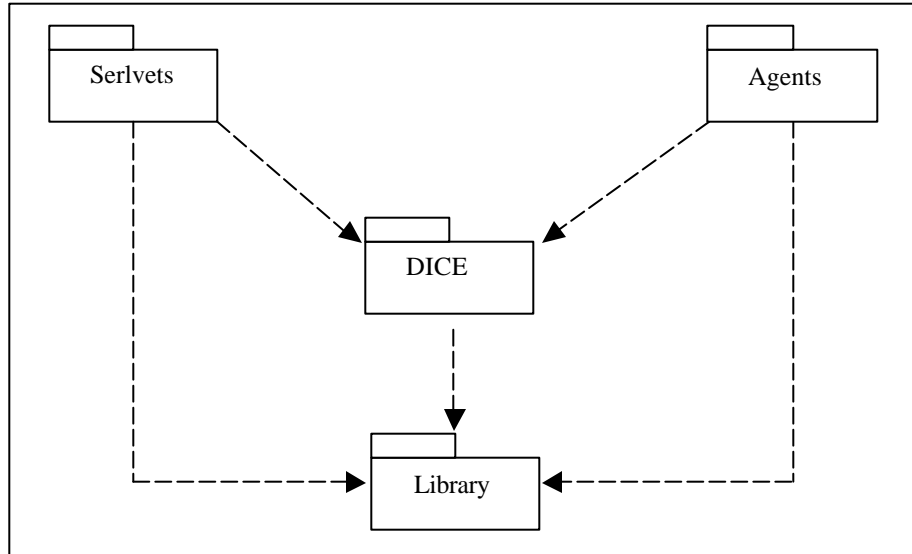


Figure 3.2 I-Help System Conceptual Architecture

- **User Interface** (the package named as Serlvets):

The users interact with the system via a set of Html pages through web browsers. The user interface is composed of static and dynamically created html pages. Information is sent between a user and his/her personal agent via Java Servlets. The Servlets used are: LoginIHelp, PasswordCheck, CheckMessages, GetMessages, FindHelp, Evaluate, Knowledge, Preference, RemoveRequest, DisplayQuestion, DiscussionList, EndDiscussion, etc.



Figure 3.3 I-Help Personal Agent Interface

For example, the LoginIHelp servlet and PasswordCheck servlet handle the login interaction. LoginIHelp servlet gets the login id and password from the user and sends the information to the PersonalAgent and automatically loads PasswordCheck. PasswordCheck servlet checks whether the password was correct. If password is correct, PasswordCheck servlet loads the main agent page for that user (Figure 3.3), if not it returns an error message to the user.

- ***Agents***

The agents are the most important component in the I-Help private discussion. The detailed explanation of I-Help agents is given in section 3.1.3.

- ***DICE***

The DICE subsystem (Deters, 2000) is the infrastructure for providing the environment for communications between agents and message passing between

agents and Servlets. It contains four subsystems: UserProxy, Host, Communication, and Basic. The Host is used by Agents subsystem to allocate the agents. The UserProxy serves as a mailbox to get and store messages which will be used by Servlets and Agents subsystems. They all need the functionality of the Communication and Basic subsystems (See Figure 3.4).

- **Library**

The Library subsystem contains routines that are used throughout the IHelp private discussion system.

Each of the above subsystems has additional sub-subsystems hierarchically nested within it. The relations shown in Figure 3.2 and Figure 3.4 are 'depends-on' relationships. For example, the Agents subsystem depends on the DICE subsystem for communication support or to communicate with the Servlets subsystem. All of the subsystems depend on the Library subsystem.

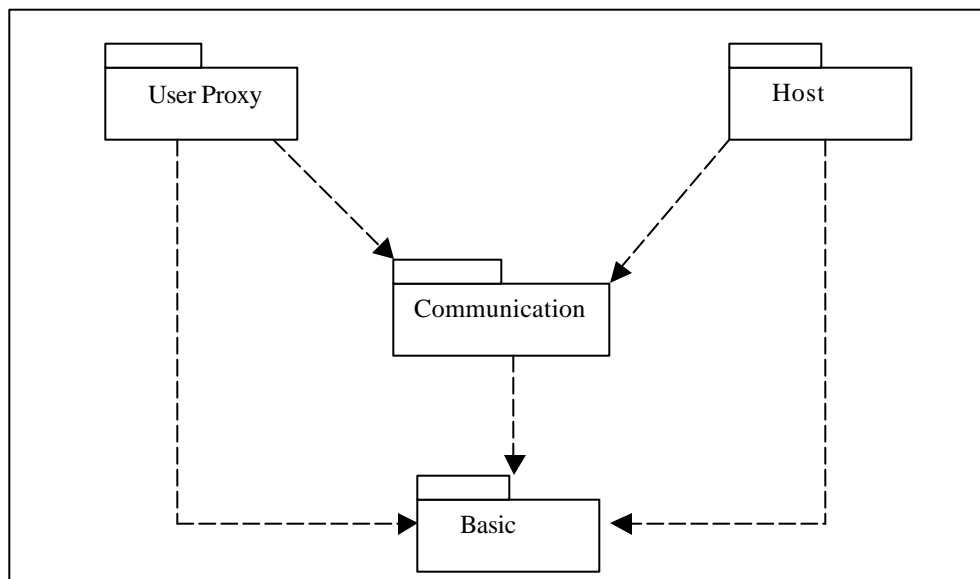


Figure 3. 4 DICE Subsystem Conceptual Architecture

3.1.2 Development View of I-Help

Figure 3.5 represents the system organization of the I-Help system in five layers. Layers 1 and 2 constitute a domain-independent distributed infrastructure that is common across the system components and shields them from variations in hardware platform, operating system, or off-the shelf products such as the database management system. Layer 3 adds

I-Help development framework to form a domain-specific software architecture. Using this framework a palette of functionality is built in layer 4. Layer 5 is very customer-and product-dependent, and contains most of the user-interface and interfaces with the external systems.

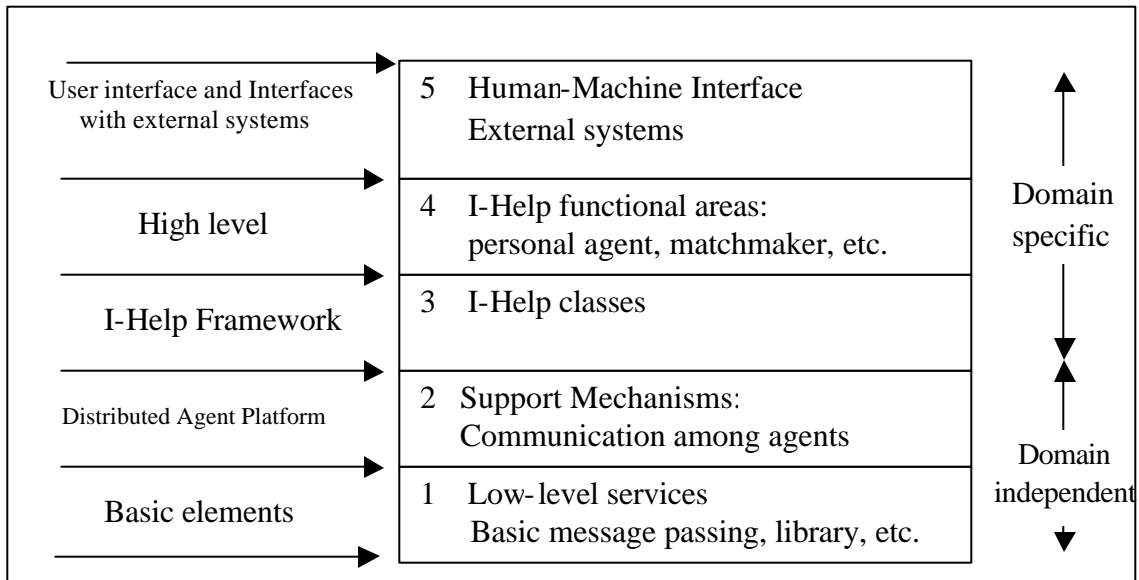


Figure 3. 5 The 5 layers of I-Help private discussion

3.1.3 I-Help Agents

❖ Personal Agents:

Personal agents are representatives of the users in I-Help. They maintain user models and tasks to be performed. The user model contains all of the characteristics of a certain user, such as the alias, friends, banned people lists, banned topics, email address, knowledge profile, preferences about how the agent should negotiate for help, etc. Some characteristics are set explicitly by the user. Some agents, such as diagnostic agents also maintain partial user models by monitoring user activities. All interactions between a user and the system are filtered through a personal agent. The personal agent either interprets the task and processes the information or collaborates with an appropriate agent to fulfill the task.

Each agent consists of four main classes: PersonalAgent.java, UserModel.java, NegotiationComponent.java and AgentFace.java. The PersonalAgent.java does the majority of the interaction between other personal agents and applications agents. UserModel.java contains all of the characteristics of the users described above. NegotiationComponent.java handles the negotiation with other agents for help requests. AgentFace.java formats all of the output that goes to the user. By separating the output, it makes it simple to replace the user interface without having to modify the agent's logic and attributes.

❖ **Application Agents:**

Application agents manage the resources around particular applications. Some of the most important agents are briefly discussed below.

- The matchmaker application agent is the agent that ranks potential helpers and performs matchmaking based on certain criteria. The set of criteria includes willingness, availability, the user's knowledge level, eagerness rating, helpfulness, etc. The matchmaker acquires the user models from the personal agents and from diagnostic agents who create partial user models.
- The DB Agent is an application agent that handles all writes and reads operations from the database. Whenever the agents need information or needs to update the database they send messages to DB agent.
- Diagnostic agents model characteristics of users in certain fields, such as eagerness, helpness, or knowledge on some topics by means of monitoring user activities, users' self evaluations, or other relevant information includes grade. There are several types of diagnostic agents currently implemented and a new diagnostic agent can be added to the system at any time.

3.1.4 Communication among the Agents

The agents in I-Help work together co-operatively to accomplish complex goals and use local information and knowledge to manage local resources and handle requests from peer agents. They communicate with each other using an expressive agent communication language-- a subset of **Knowledge Query and Manipulation Language**

(Finin et al., 1997). **Common Object Request Broker Architecture --CORBA** (Raj, 1998) is used as an infrastructure that enables agents to locate one another in the distributed environment and facilitates message passing. Before messages can be sent, each agent must register itself on the CORBA name server so that other agents can find it. In general, each message contains sender, receiver, subject of message, content of message, etc. An agent makes a decision on how to process and respond to the message based on the subject and content of the message.

CORBA was designed to be platform and language independent. Therefore, CORBA objects can run on any platform, be located anywhere on the network, and can be written in any language that has Interface Definition Language (IDL) mappings.

The agent platform and multi-agent architecture of I-Help was designed and implemented by the other researchers in the ARIES lab. This thesis work extended their implementation and made agent programmable so that the users in the I-Help can program their personal agent to monitor others' activities. In the next section, the design of agent programmability is described.

3.2 Design of Agent Programmability

This section describes how a user programs his or her agent and the high level architecture of the system.

3.2.1 How Users Program Their Agents

An end user programming environment allows users to program their agents so that the agents can monitor certain events happening in the system and take some actions accordingly.

Events frequently occurring in the I-Help system might include: a user has just signed in to the system, a user has just received a message from another user, a user has sent a request for help to a particular user, etc. For the purposes of the I-Help system, events have been broken into seven main categories:

- Logging in/out of the system

- Changing preferences
- Viewing pages
- Responding to a help request/posting messages
- Requesting information
- Updating system attributes (evaluations, prestige, etc.)
- Updating privacy settings

The user can tell his/her agent what actions to perform when these events happen. A set of actions for the agent include:

- Notify the owner about the event
- Email the owner with some given subject and content
- Send an email to other users with some given subject and content
- Communicate with other agents on behalf of its owner.

These triggers and actions describe the kind of programming functionality a user might wish to have available. The following example shows how a user instructs his/her agent explicitly about what action/actions should be taken under a certain condition. The hypothetical user, Yang, provides a rule called *read-message*, which is used to program her agent to contact Sam, if Sam has logged in to the system within the past 30 minutes and if Sam has read a particular message within the past 30 minutes.

Rule name: read-message

If

User Sam logs in to the Private Discussion within the past 30 minutes *and*

User Sam reads the message with message id 19291 in the Public discussion forum within the past 30 minutes

Then

Notify Sam with subject: discussion and content: "...we need to talk"

3.2.2 System Architecture

In order to build an environment allowing the user to program his or her agent, the human-agent interface, Rule Supervisor Agent, Rule Management module, Rule Cycle Detector modules were developed as well as modifications and additions made to the personal agents and DB agent. Figure 3.6 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.

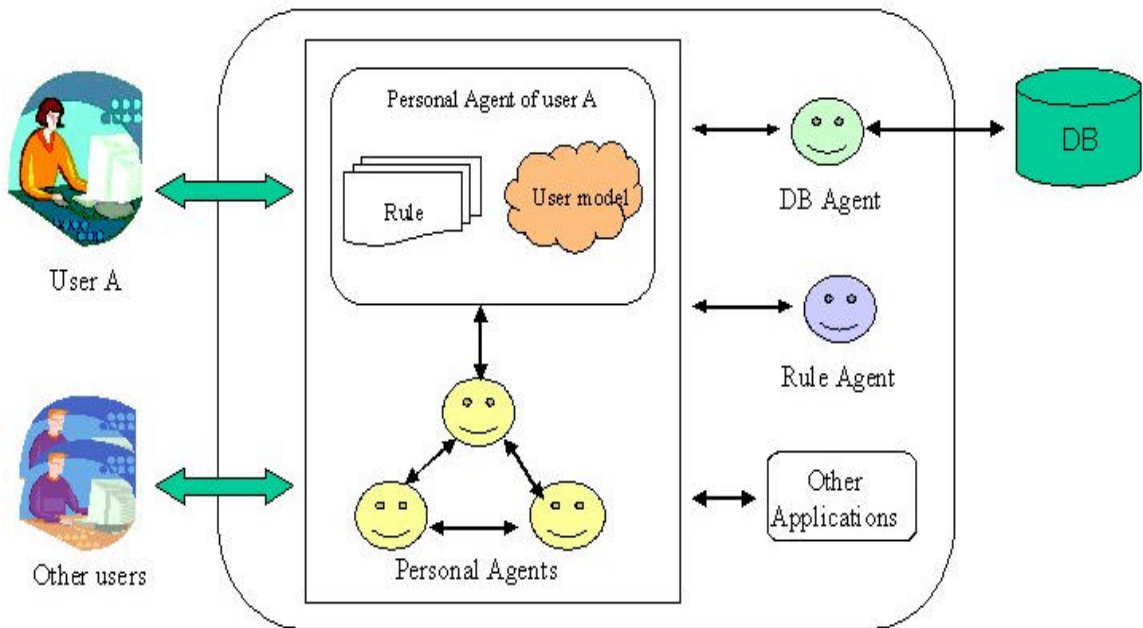


Figure 3. 6 Architecture of I-Help End User Programming Environment

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. The interactions between a user and his/her agent are through a set of user interfaces including presentation of information, option selections, and form- filling. Additional graphical user interfaces were designed and implemented to allow the users to enter their rules or fill the template form. For each type of event the users are able to set conditions and actions that should be taken by the agent. Other kinds of new graphical user interfaces and modifications to existing graphical user interfaces were designed and implemented for displaying messages to the user, such as giving a notification signal to a user and

displaying messages about events in which the user is interested. The function of each module is briefly introduced below.

- 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. The personal agent achieves these functions by working with other agents and applications.
- 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.

Two alternative approaches for implementing an end user programming environment were employed in the context of IHelp agents. Detailed information on these two approaches and the implementations is introduced in next section and Chapter 4.

3.3 Functionality of the Agent Programming Environment

The I-Help end user programming environment is a Web-based system where the users interact with the system through normal web browsers. Two approaches were employed to implement the agent programming environment. One approach is to add to each agent in I-Help a simple customized rule system, which we called the Agent Rule Management System (ARMS). Another approach is implementing CLIPS-based agents that involve connecting the CLIPS rule-based expert system shell to each personal agent in I-Help. In

this section some examples of usage of the system and related user interfaces are described and sample user interfaces are presented.

3.3.1 Example Scenarios

Figure 3.7 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 3.7 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".

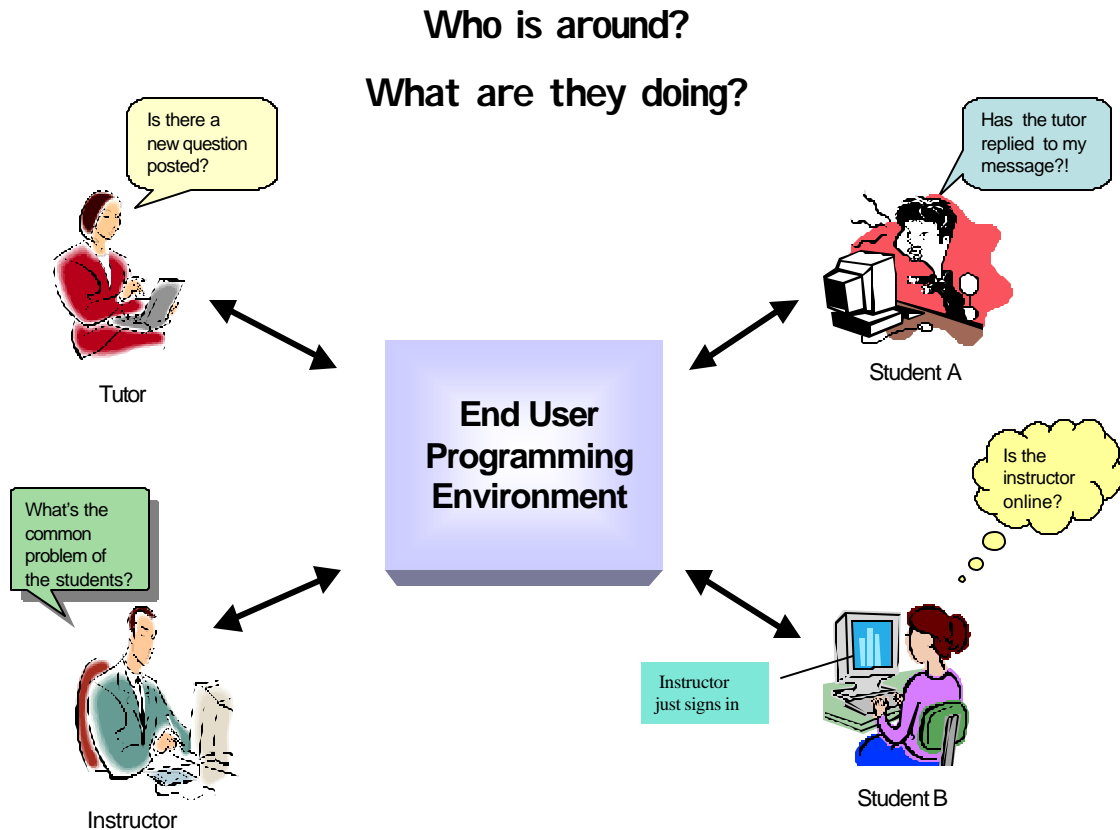


Figure 3. 7 Examples of usage of the end user environment

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 detailed information about interfaces for editing a rule and checking notification messages is discussed in the next section.

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.

3.3.2 User Interfaces of ARMS

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 3.8).

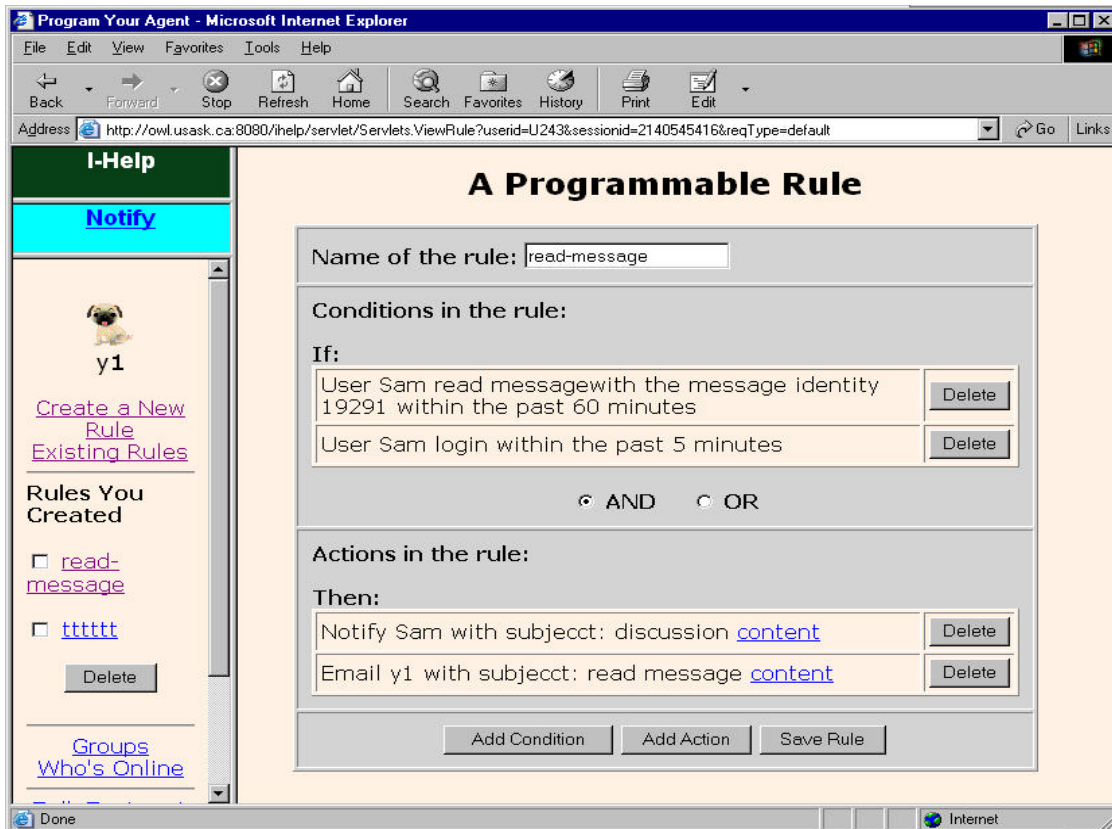


Figure 3. 8 Rule Management Interface

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 3.8 shows that there is a new notification message for the user. The index frame (left part in Figure 3.8) 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 3.8), condition specification (Figure 3.9), and action specification interfaces (Figure 3.10).

Each rule contains three parts as described previously. The users are able to add a new condition to the rule by clicking on the “Add Condition” button in the rule editor and this will open the condition specification window for the users.

Users specify rule conditions by selections on event type, the user, and other parameters involved. Figure 3.9 shows a condition describing “User Sam has read any message

within the past 2 minutes". Users can add an action in a similar way. Figure 3.10 presents an action which notifies Tom's agent with subject discussion and content: "Sorry, I couldn'tuntil Sep30, 2002". The conditions and actions will be displayed as an understandable English sentence in the rule editor after they are added to the rule. The functions of the rule editor also include the deletion of conditions /actions, and saving function.



Figure 3. 9 Condition Specification

Once the conditions of the rule are satisfied, the agent will perform the actions specified in the rule. The current available actions are emailing the agent's owner and other users, as well as notifying users or agents by placing notification messages in agents' notification boxes. When there is a new notification message in an agent's message box, the agent will signal his/her owner. The user is able to check the notification messages by clicking on the notification signal bar. This will open a window with the notification messages displayed.

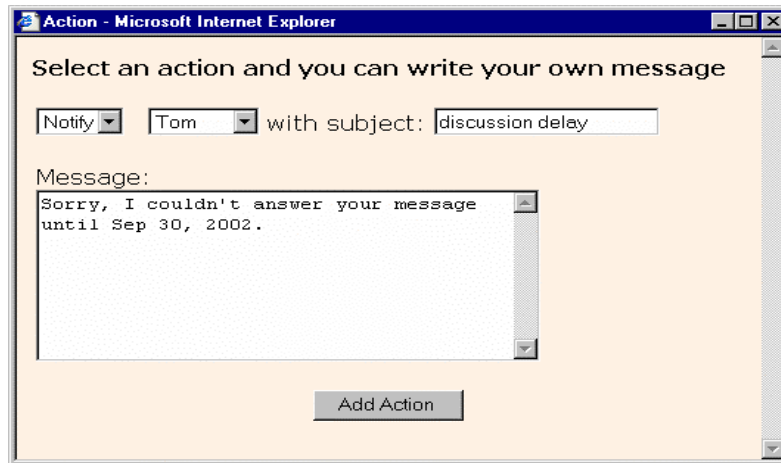


Figure 3. 10 Action Specification Interface

3.3.3 User Interface of the CLIPS-based Rule Environment

The rule management interface of the CLIPS-based rule environment is similar to Figure 3.8, 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. Similarly to the 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 3.11 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 3.11). 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. The detailed information on how to create a rule and the predefined functions provided by the system are described in Chapter 4.

```

(defrule loginNotification

(User who )
(login ?y ?within&: (< ?within 2 minutes))

(test (eq ?y ?z))
=>

(bind ?nagent (fetch Notifyagent))
(bind ?notice (call ?nagent createNotice "login" ?y ))
(call ?nagent sendNotice ?notice send to whom ))

```

Figure 3.11 An 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 compromise 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. While it will be a big challenge to track errors when users write rules completely by themselves, one approach to solve this problem is providing users with a suitable editor/compiler of CLIPS on the client side that enables the users to configure a set of syntactically correct rules. Another approach is checking for syntax errors on the server side after a user submits a rule to the system and informing the user about syntax errors.

CHAPTER 4

IMPLEMENTATIONS OF PROGRAMMABLE AGENTS

In this chapter, two variations of implementation for agent programmability in I-Help are introduced. 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.

4.1 ARMS Approach

The Agent Rule Management System (ARMS) is a simple customized rule system which allows users to instruct their agents to monitor certain events happening in the system and take some actions accordingly. With the ARMS-based agents, the users are able to configure rules by filling or selecting values from a set of given options in web based user interfaces.

4.1.1 Design of the Rules

Each user can provide multiple rules for his or her agent. We have engineered the system (Cao and Greer, 2003 b) so that rules are simple triggers. That is, all rules in the repository of an individual agent fire independently of each other. This is achieved by restricting the set of conditions and actions the user can elect to use in a rule.

Each rule contains a rule name, a set of conditions (If section), and a set of actions (Then section) that an agent will perform. Each condition has three or four parameters: a user, activity type, and other parameters specific to the activity type. Each action corresponds to a notification - a user/agent who needs to be notified, the notification subject, and the message content. Table 4.1 presents the various formations associated with different types of activities.

Table 4.1 Formation of the conditions in the rule

	User	Activity type	Parameters	
Login	user A	logs in	within the past N minutes	—
Logout	user A	logs out	within the past N minutes	—
Read Message	user A	read message	with message identity Y.	within the past N minutes
Send Message	user A	Send message	to user B	Within the past N minutes
Send Notification	user A	notify	user B	Within the past N minutes

If there is more than one condition in a rule, the user can specify how the conditions are connected. For simplicity, we decided to restrict the user from having access to full Boolean conditions and rather to join all the conditional clauses of a rule with a single “AND” or a single “OR”. If “AND” is used then all the conditions must be satisfied at the same moment for the rule to succeed. If the conditions are combined by “OR”, then the rule will be fired if any condition is met. The following are two examples of the combination of the conditions:

e.g.1.

If user Tom has logged into the system within the past 2 minutes

and

user Sam has read any message within the past 50 minutes

then

notify me.

e.g.2.

If user Tom has logged into the system within the past 5 minutes

or

Sam has logged into the system within the past 2 minutes

or

user John has logged into the system within the past 5 minutes

then

notify me”.

If a user wants to specify a rule that involves all users, unlike Object-Lens system (Lai, et al., 1998) the rule does not need to include the names of all the users in the option list. The user can simply select “all users” or “any user” from the given list. The agent will find the real meanings of “all users” and “any user”.

The rules are applied in the order in which they appear in the agent’s rule set. Each rule is restricted to fire at most once in a given context. Once fired, the rule sleeps until the time limit of the rule expires. Each condition has a time parameter specified by the user (see the last parameter in Table 3-1). The system generates the rule time limit according to the time parameters of all conditions in the rule.

The following formula illustrates how the rule time limit is defined. The minimum time parameter will be chosen as a rule time limit if the operator of the conditions is under “AND”. In contrast, the maximum time parameter will be selected under the operator "OR". For example, in the example 1 described above, the time parameters of the two conditions are 2 minutes and 50 minutes. Since the logic operator is “AND”, then 2 minutes will be chosen as the rule time limit. That means the agent will check this rule every two minutes.

$$\text{TimeLimit} | T_r = \begin{cases} \text{Min}\{T_{\text{cond}1}, T_{\text{cond}2}, \dots, T_{\text{cond}k}\} & \text{if Joining-Operator } r = \text{"AND"}; \\ \text{Max}\{T_{\text{cond}1}, T_{\text{cond}2}, \dots, T_{\text{cond}k}\} & \text{if Joining-Operator } r = \text{"OR"}; \end{cases}$$

Although the rules are mutually independent, special configurations of rules may still cause runaway behaviour within the system. For example, user John has a rule called rule #1 and user Ben has a rule called rule #2. The detailed information for the rules are illustrated below:

rule #1:

"if user Ben sends a message to user John, then send a message to user Ben".

rule #2:

"if user John sends a message to user Ben, then send a message to user John".

In this situation, if either user John or Ben sends a message to the other, there will be infinite message passing between these two users. Even a single rule may cause a similar problem. For example, "if user Yang has a new message, then send a message to Yang." will cause a self cycle.

Nardi's studies of spreadsheets (Nardi, et al., 1998) shows that cycles involving a single rule are rather simple to locate, however, dealing with interaction among rules is much harder. In our system the interactions among rules are checked wherever a rule is generated or modified. This will be explained in detail in Section 4.1.2.

A rule becomes active at once after it is created and checked for validity. Each rule has a status to record the last check time and time limit. The system updates the current status whenever a condition or an action is added, deleted, or the logical operator is changed. When a rule is updated, the validity on the interactions among the rules must be rechecked immediately.

4.1.2 Design of the Cycle Check

Whenever a user creates or updates a rule, the system will automatically try to detect interactions among existing rules, and help users to create an effective set of rules for their agents. This involves following chains of rule actions that result in sending messages to agents that would trigger firing of another rule. If a cycle is detected, the newly added rule is problematic. In order to conduct a cycle check, first we need to define what the trigger means here.

The rule A triggers rule B iff the actions of the rule A are same as the conditions of rule B or the action set of the rule A subsumes the condition set of rule B.

Figure 4.1 illustrates the situations when rule a triggers rule b.

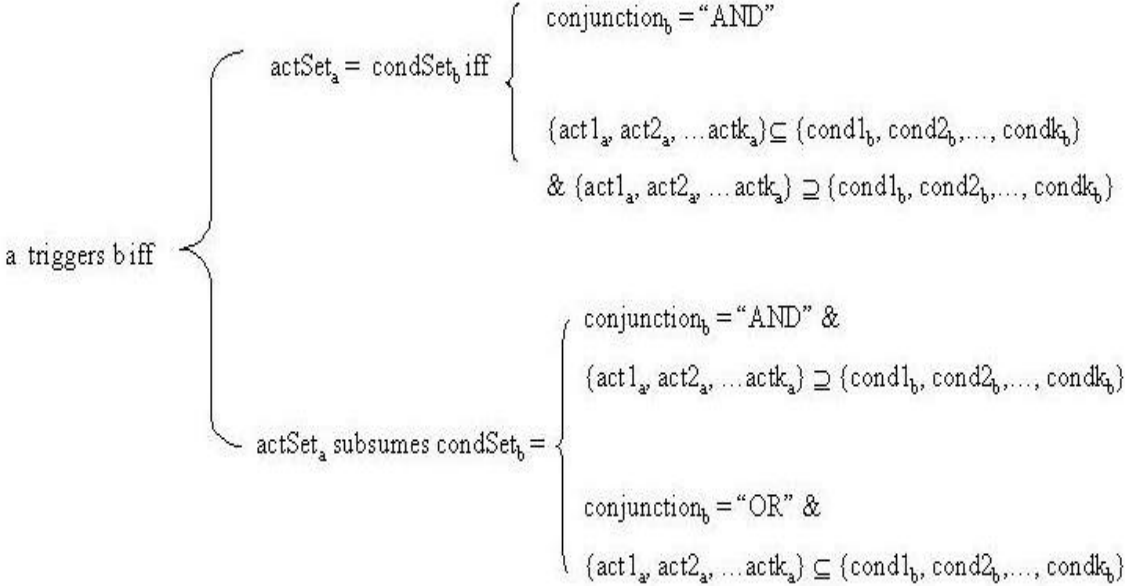


Figure 4.1 The situations of rule a triggers rule b

Idea of Cycle Check

A directed graph is employed to store and check possible cycles among the rules. The nodes in the graph present rules and the arcs between two nodes presents the relationship between the two rules. If rule A triggers rule B then there will be an arc from node A to node B. Figure 4.2 shows an example of rule graph. The graph contains 5 nodes that presented 5 rules generated by user Andrew, Ben, Danny, and Ellen.

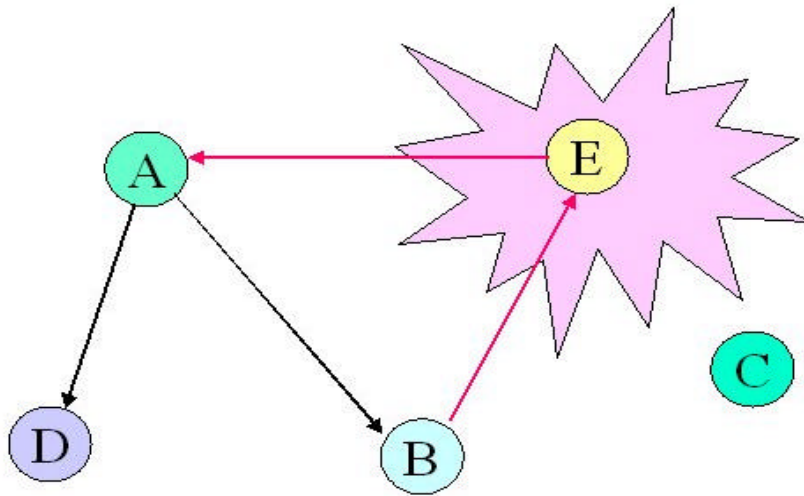


Figure 4.2 Rule Graph

Cycle check is composed of two main steps: graph creation and cycle detection. A scenario of adding a new rule to the rule repository describes the idea of cycle check (Figure 4.2). For example, image that there are four rules in the repository initially: Andrew has two rules: rule A and rule C, Ben has one rule named rule B, and Danny has one rule called rule D. The following are the contents for each rule.

Rule A: If user Ellen notifies user Andrew then notify Ben.

Rule B: If user Andrew notifies user Ben then notify Ellen.

Rule D: If user Andrew notifies user Ben then notify Danny.

Rule C: If any user login then notify Andrew.

These rules tell us that the action of rule A is same as the condition of rule B and rule D. The arc from node A to node B is then added to the graph according to the definition of trigger, similarly for arc $A \rightarrow D$. The rule C is isolated as rule does not trigger any rules or is triggered by any other rules. We can see that there is no cycle currently. Assuming user E configures a new rule called rule E. The detail for the rule is displayed as follow:

Rule E: If user Ben notifies user Ellen, then notify Andrew.

In Figure 4.2, we can see that adding rule E results in adding two new arcs: $B \rightarrow E$ and $E \rightarrow A$ to the graph. These new arcs combine with original arc $A \rightarrow B$ resulting a cycle in the graph. This indicates that rule E is problematic and either the conditions or the actions of the rule E must be reconfigured. While rule E could be left intact if some other rule were revised, this is not feasible in our system since the other rule may be “owned” by some other personal agent.

Algorithm

The algorithm for the cycle detection is illustrated by a procedure `cycleDetect` and a function `cycleSearch`:

```

procedure cycleDetect (Graph G)
    for each node  $v \in G$  do
        mark[v]  $\leftarrow$  not visited
        active [v]  $\leftarrow$  F
    pnum  $\leftarrow$  0
    for each  $v \in G$  do
        if mark [v]  $\neq$  visited then
            cycleSearch [v]

```

```

function cycleSearch ( Node v)
    pnum  $\leftarrow$  pnum +1
    prenum [v]  $\leftarrow$  pnum
    mark [v]  $\leftarrow$  visited
    active [v]  $\leftarrow$  T
    for each node w adjacent to v do
        if mark[w]  $\neq$  visited do

```

```

        parent [w] ← v
        cycleSearch (w)
    else if prenum[w] < prenum [v] and active [w] then
        output ( 'cycle found' )
        return true
    active [v] ← F
    return false

```

These algorithms work well in cycle detecting. However there are some limitations in cycle checking. There is no way in principle to check for all infinite loops. For example, imagine that there are four rules in the repository (Figure 4.3): Andrew has a rule called rule A, Ben has one rule named rule B, Danny has one rule called rule D, and Elaine has a rule called rule E. The following are the contents for each rule.

Rule A: If I receive a message from Elaine, then notify user Ben and Danny.

Rule B: If user Andrew notifies me then notify Ellen.

Rule D: If user Andrew notifies me then notify Ellen.

Rule E: If both user Ben and user Danny send message to me, then notify Andrew.

The arcs from node E to node A (E-> A), A to D (A-> D) and A to B (A->B) should be added to the graph according to the definition of trigger. Rule E shows that Rule B or rule D is only partial of the trigger for rule E. We call this situation as conditional trigger. In the current implementation we do not consider conditional triggers and these rules can be added to rule repository. While this permits possible cycles, it was considered that such cycles would be rare.

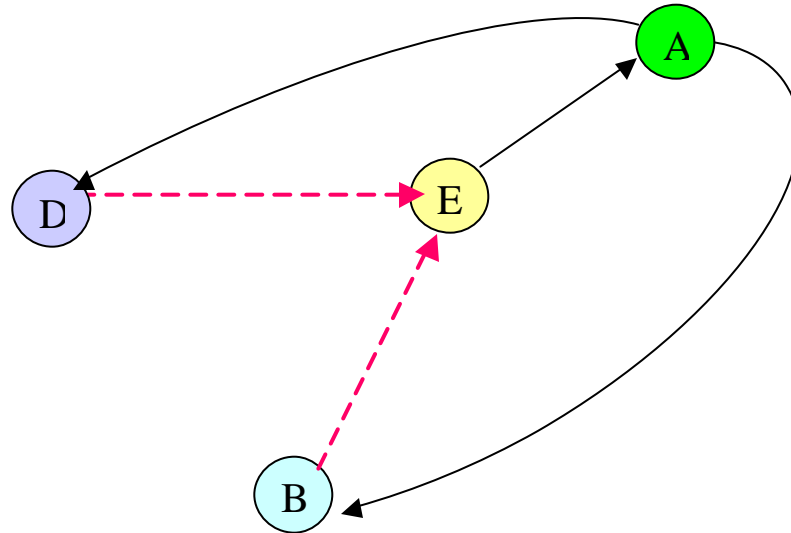


Figure 4.3 Rule Graph with conditional triggers

4.1.3 Structure of ARMS

Figure 4.4 represents the architecture of the I-Help ARMS. Each PersonalAgent has one RuleManagement that is a module for managing the rules for an individual user. RuleCycleDetector is a component that does the actual work for checking the validity on the interactions among the rules. AgentFace formats all of the information that goes from the personal agents to their users. The proxy is served as a message queue to store messages which will be used by servlets to display information to users. The functionalities of Personal Agent, DB Agent, RuleAgent, and User Interface components can be found in previous section-- 3.2.2. Personal Agent, DB Agent, and RuleAgent all inherit from TaskBase which is a generic agent that contains generic agent functionalities, such as message passing between agents.

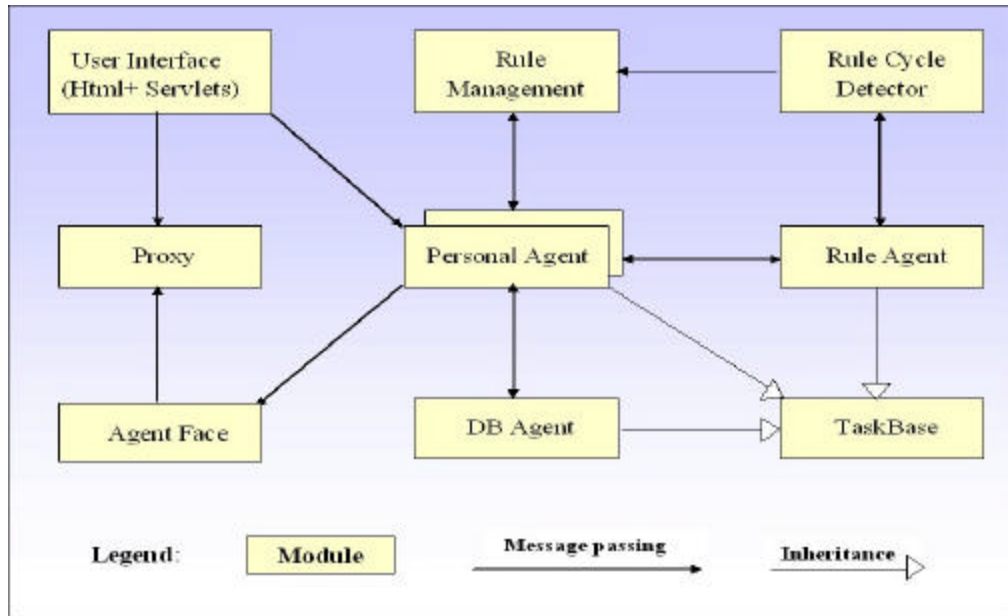


Figure 4.4 Structure of ARMS

4.2 CLIPS Approach

As described in section 3.2.1, the users are able to instruct their agents to monitor certain events happening in the system and take some actions accordingly. In order to evaluate the suitability of ARMS, we built a second implementation relying on an “off the shell” rule interpreter, CLIPS. With the CLIPS-based agents, the users instruct the agents by writing rules in JESS (Java Expert System Shell), which is a Java implementation of CLIPS. We made it possible for users to construct arbitrary JESS rules to control their agents as well as to fill the rule templates, which are provided by the system.

4.2.1 Structure of a CLIPS/JESS rule

CLIPS is a productive development and delivery expert system tool which provides a complete environment for the construction of rule-based and/or object-based expert systems. CLIPS can be modified or tailored to meet a user's specific needs fairly easily. CLIPS provides a cohesive tool for handling a wide variety of knowledge with support for three different programming paradigms: rule-based, object-oriented and procedural. Rule-based programming is one of the most commonly used techniques for developing

expert systems. In this programming paradigm, rules are used to represent heuristics, or "rules of thumb," which specify a set of actions to be performed for a given situation.

A rule is composed of an *if* portion and a *then* portion. The *if* portion of a rule is a series of patterns which specify the facts (or data) which cause the rule to be applicable. The *then* portion of a rule is the set of actions to be executed when the rule is applicable. The actions of applicable rules are executed when the inference engine is instructed to begin execution. The inference engine selects a rule and then the actions of the selected rule are executed (which may affect the list of applicable rules by adding or removing facts). The inference engine then selects another rule and executes its actions. This process continues until no applicable rules remain.

JESS is a rule engine and scripting environment for CLIPS written in the Java language. Using JESS, one can build Java applets and applications that have the capacity to "reason" using knowledge supplied in the form of declarative rules. JESS uses the Rete (Latin for *net*) algorithm to process rules, "a very efficient mechanism for solving the difficult many-to-many matching problem" (Forgy, 1982). The Rete algorithm remembers past test results across iterations of the rule loop. Only new facts are tested against any rule left-hand-side (LHSs).

The following is a simple Jess rule example:

```
(defrule library-rule-1
  (book (name ?X) (status late) (borrower ?Y))
  (borrower (name ?Y) (address ?Z))
  =>
  (send-late-notice ?X ?Y ?Z))
```

Note that this syntax is identical to the syntax used by CLIPS. This rule might be translated into psueudo-English as follows:

Library rule #1:

If

a late book exists, with name X, borrowed by someone named Y

and

that borrower's address is known to be Z

then

send a late notice to Y at Z about the book X.

The book and borrower entities would be found in the knowledge base. The knowledge base is therefore a kind of database of bits of factual knowledge about the world. Actions like send-late-notice can be defined in user-written functions in the Jess language (deffunctions) or in Java (Userfunctions).

4.2.2 Bottleneck of the Implementation

One of the challenges in designing and implementing CLIPS-based agents is to restrict a rule to be fired at most once in a given context. Two possible techniques to address this problem are briefly described in the following. In this thesis work, the first approach is used.

- Record the time for each rule to be checked, a rule will not be checked again until the time limit of the rule expires. For example, the rule can be checked every two minutes.
- Set a context variable for each rule and every time the context variable needs to be considered when a rule is to be fired. After the rule is fired the status of the context variable will be changed (eg. change to false). Define another rule to reset the context variable according to the time parameters of the conditions in the original rule.

Each user is able to provide as many rules to their agents as they wish. In the current system, a sample set of *if clauses (events)* and *then clauses (actions)* are available to users as listed in section 3.2.1. There are several parameters associate with each type of event. Users could make a simple rule by filling the value in a rule template or they could make complex rules by combining several events/actions. If they so choose, the users are able to define their rules from scratch without using any events/actions provided by the system.

Basically these rules are designed to be more or less independent of each other. However, the users could configure a rule to trigger other rules. This makes it a challenge in handling the interactions among the rules. For example, in some situations a group of rules might trigger each other resulting in an infinite trigger loop. Therefore, the system checks the validation on rules actions before executing them. Checking validity of rules actions is achieved by tracking the chains of rule actions and conditions. If a cycle is found, then the latest added rule needs to be fixed. Of course there is no way in principle to check for all infinite loops.

4.2.3 Structure of the CLIPS-Based Agents

Figure 4.5 shows the structure of CLIP-Based agents (DB Agent and MatchMaker are not shown in the diagram). The rules are entered from a human-agent interface and then passed to the Personal Agent. Notification Manager serves as a coordinator that cooperates with each Personal Agent, retrieves the event stream, and works with the JESS Engine. A JESS Engine instance and some predefined functions are also created in the Notification Manager class. JESS is a powerful Java scripting environment, from which one can create Java objects and call Java methods. It serves as middleware between Java classes and CLIPS rules. The JESS instance processes the rules using data provided by the Event Stream. If the facts (current situations) are matched with the conditions in the rule, then the rule will be fired and the action specified in the rule will be executed.

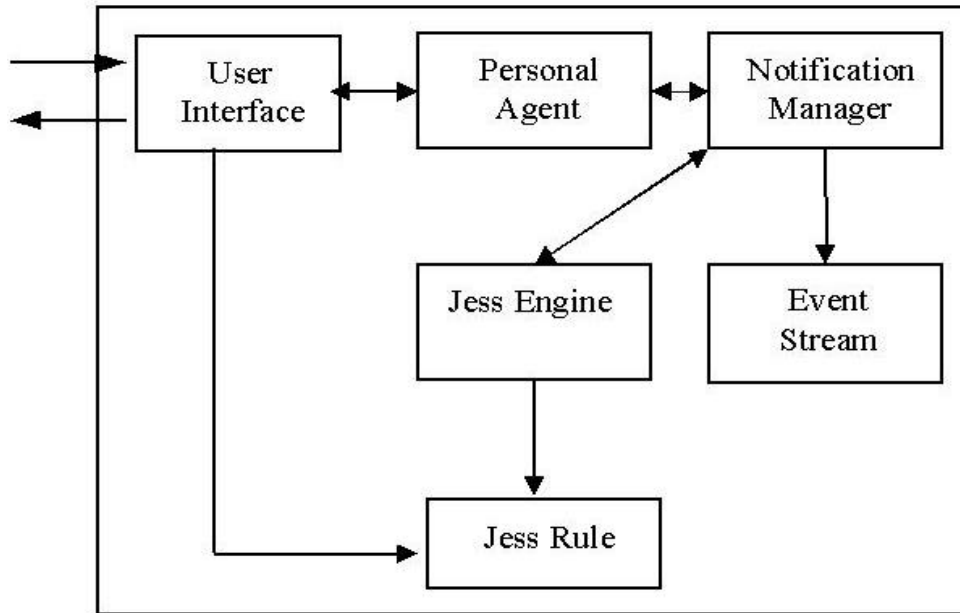


Figure 4.5 Structure of CLIPS based Agent

4.2.4 Creating Rules in the CLIPS-based Approach

As described in section 3.3.3 the users are able to configure a rule by selecting and filling the value in a template as well as to make complex rules by combining several events/actions. The system provides a set of predefined functions for the user to employ in their rules. We implemented some Java classes and procedures/functions that are available for users. For example, Notifyagent and Notice are two Java classes, while createNotice and sendNotice are Java methods in these classes. The users are able to create an instance of Java class and call the methods in that class from CLIPS rule.

Figure 4.6 is a sample rule called loginNotification that was generated by a user. In line 7, the user created an instance of a Java class Notifyagent which is similar as creating an instance in Java:

```
Notifyagent nagent = new Notifyagent ( );
```

In line 8, the user created a “Notice” object by calling a method of class Notifyagent.

The method entitled as createNotice with four parameters including a String “login”, and three variables “?y”, “?date”, and “?time”. The interface of this method is shown below:

```
public Notice CreateNotice ( String type, String user, Date date, Time time)
```

In line 9 the user called the method `sendNotice` in `Notifyagent` with two parameters: a `Notice` object and the name of receiver.

The system provides a simple Java API for users' references. Of course the users have the option to define their own rules without using any functions provided by the system.

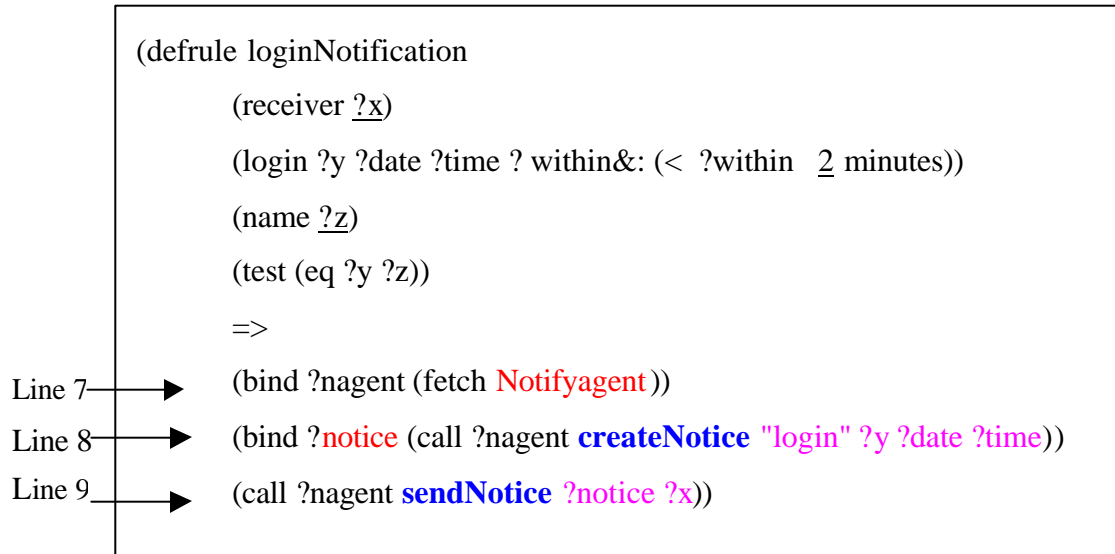


Figure 4.6 A Sample Rule Template

4.3 Summary

Two implementation for user programmable agents were presented in this chapter. The ARMS approach was built as a simple and easy to use mechanism where users could add capacity to their agents. The CLIPS approach provides a more powerful agent programming system, but is more complex and potentially less robust because users can more easily write buggy rules or cause infinite loops.

CHAPTER 5

EVALUATIONS AND RESULTS

This chapter explains the case studies used to evaluate IHelp agent programmability. Two experiments were conducted, including a usability experiment on the ARMS enabled programmable agent environment and a comparative user study on the ARMS approach versus the CLIPS approach. The result of the usability experiment and the comparison between these two approaches were analyzed. This chapter first summarizes the objectives this thesis seeks to address. The rest of the chapter describes the case studies and the analysis of the results.

5.1 The Research Issues and Objectives

The research done for this Master's thesis had several aims. In summary, the main objectives were to investigate the following questions:

1. How easy or hard can it be for users to program their agents?
2. Will people feel that agent programmability is helpful?
3. Will users try to destroy the system by malicious use of agents and how can one protect the system?
4. What are the users' concerns about their privacy?
5. Will the capability of I-Help improve with agent programmability?
6. Can 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?

5.2 Case Studies Design

❖ Usability Study on ARMS

An experiment on usability of the ARMS approach 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
- ◆ Whether the students had better performance in a learning task with agent programmability than no programmability support

❖ **Comparative Usability Study On Two Approaches**

One of the objectives of the second user study 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. Based on this objective, a comparative usability study on the two approaches was conducted. A group of people compared the two interfaces to see which interface they would like to work with, faking the back end and not considering the usefulness of the resulting programs in terms of smarter agent actions.

5.3 Usability Study on ARMS Approach

5.3.1 Experiment Procedure

A two-hour usage study of the ARMS system was conducted with fifteen university students from different disciplines acting as experimental subjects. Subjects' activities included attending a training session, using the ARMS to fulfil information seeking task, and completing an exit questionnaire on feelings about the system and privacy concerns.

At the beginning of the training session, brief information on I-Help public discussion and private discussion were introduced. Each subject was given a demonstration of the

ARMS system, which described what kind of activities they can monitor and how to receive notifications, how to use the system. (see Appendix A for detail).

During the experiment, three subjects were appointed as tutors (experts) and the rest were assigned the role of information seekers. The information seekers were asked to complete a quiz based on information they found, which required collaboration with others. The subjects could post questions on the IHelp public discussion forum or they could get help from other subjects through the IHelp one-to-one messaging system. They were periodically prompted to connect (log in) or disconnect (log out) from their agent. There were no face-to-face communications among the participants. A participant had to find out when others were online, whether they had read his/her messages and whether they had replied to his/her messages.

The experiment was replicated with similar tasks (see Appendix B for detail) in two situations: with and without agent programmability. When agents were to be programmed, the participants were asked to initially create a rule set and were permitted to edit or add rules as the experiment progressed. When programmable agents were not available, the participants could query the system (check status of who is online and look at tables of who read what). In each situation participants were given the same amount of time to complete the task.

The subjects' activity logs were recorded for later analysis. The achievement scores (maximum 20 points) for subjects under each treatment condition were collected and used to measure performance/ achievement.

After the participants fulfilled some tasks using the system, they 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 questions, which included views on surveillance, privacy concerns, and questions on using the IHelp rule management system. A copy of the interview sheet can be found in Appendix C.

5.3.2 Results

Under the condition with agent programmability support, most of the subjects generated 2 to 4 rules and each rule took them about 1 minute to configure. We observed that most

rules are appropriate for facilitating their access to the necessary resources that include tutor and electronic resources. During the experiment no one configured a rule to intentionally harm others or the system (i.e. denial of service attacks, etc.). A few potential rule cycles were caught because of the careless use of “all users” and “any user” in the rule. For example, the rule "If anyone notifies me then notify all users" causes a cycle.

The subjects’ achievement scores are analysed in Figure 5.1. It demonstrated that 10 subjects had better achievement with agent programmability support. The mean scores of the subjects with and without using agent programmability are 11.95 and 9.47, respectively. There is a significant difference between the achievement under two situations (paired t-test, $t=4.24, t=3.81, P<0.01$). However, there are differences in improvement in achievement for individuals. The score increased from between 1.2 and 6 points on the 20-point scale.

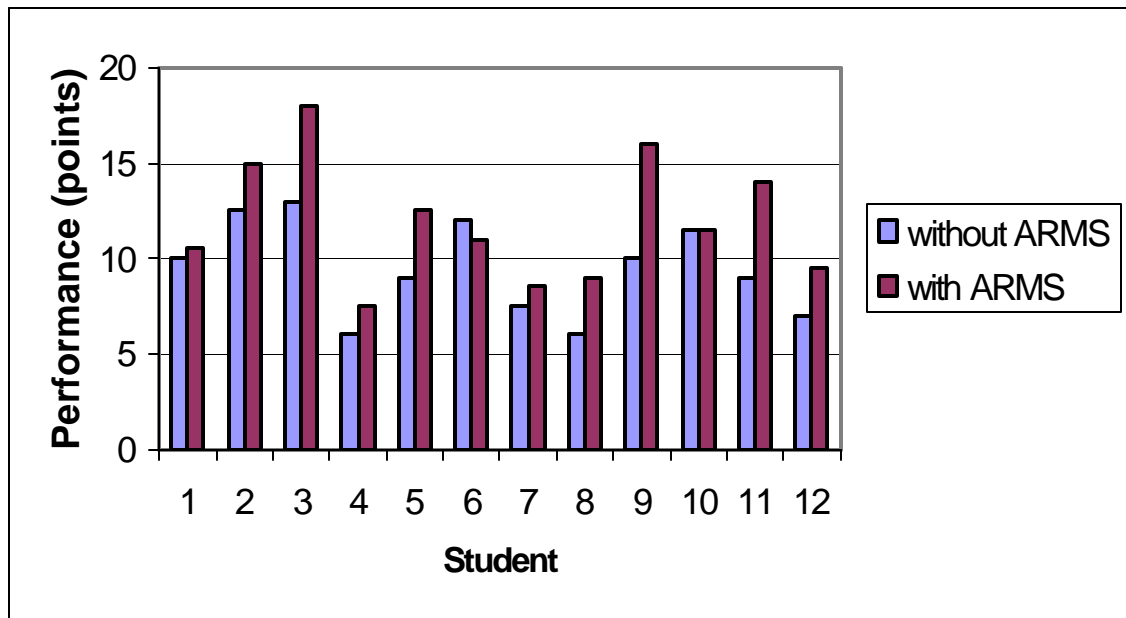


Figure 5.1 Performance for each student

The results of the questionnaire filled out by the participants are summarized in Table 5.1:

Table 5.1 Result of Questionnaire on ARMS

1. When, what, whom and whether the people like to watch others:			
login, read message, and send message activities		just for curiosity	
92% watch instructor/tutor, and knowledgeable students.	8% watch any person	None	
2. How easy is it for users to program their agents (configure a rule):			
easy to understand and operate without help	easy to operate with a little help	confusing or very hard to operate	
75%	25%	None	
3. Do programmable agents facilitate learners in accessing necessary resources:			
very useful	helpful to some extent	a little helpful or useless	
80%	20%	None	
4. How do users prefer to find out the status of events happening in the system:			
program their agents i.e. generating a set of rules.	check by themselves for some events (i.e. logged in).	send email asking	
80%	20%	None	
5. How would the users like to use the system in the future:			
More than before with programmable agents	Same as before	Less than before	
100%	None	None	
6. People's privacy concerns when someone watches their activities?			
When someone watches my	I do not mind	Depends who it is	Big concern
Log in	72%	28%	None
Send message	30%	50%	20%
Read message	48.3%	34.7%	16.7%

5.3.3 Discussion

We believe that one reason for the improvement in achievement is that agent programmability facilitates the subjects' access to necessary resources (human and electronic) and interaction among subjects in IHelp by providing information about whether a person is available and communicative. Another reason is that people can better attend to their work when there is agent programmability support, since they do not need to explicitly check on events such as whether their questions are answered in the public discussion forum or whether their messages have replies. With agent programmability, they can program their agents to inform them when a particular event happens. The knowledge level of the group members and their willingness to provide help may also affect the individual's achievement.

Table 5.1 shows that surveillance of users' activities is not a big concern for most of the subjects and they are willing to disclose information about themselves. This result may be because of the perceived benefits of the effects of the programmable agents. During the interview, several participants mentioned that there is not much privacy concern in an online learning environment as long as no grade or other personal information is released. They also indicated they would feel more comfortable about releasing more information after experiencing the benefits of this system. This is encouraging, as the system would not be useful if most people were interested in others' activities but kept themselves private. However, 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.

Allowing users to view one another's activities in the system may make the system somewhat transparent and the users will be able to accomplish their respective goals more efficiently. However, while surveillance can be used for good purposes it also can be used for bad purposes. It may result in undesirable consequences and some users may be concerned about their privacy. The result in Table 5.1 indicates that there is concern about privacy on some events. Therefore it will be desirable to allow users to restrict who has access to some events. It is also essential for users to know who are watching, what they are watching, and how to protect themselves from stalking or inappropriate surveillance.

5.4 Comparative Usability Study on CLIPS vs. ARMS

5.4.1 Experiment Procedure

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 staffs 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 IHelp 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 IHelp, 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 (see Appendix E for detail).

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. A copy of the questionnaire on usability of the systems and the comparison is in the Appendix F.

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 IHelp agent programming environment and whether surveillance issue and privacy concern might prevent them from using the system in future. The interview sheet can be found in Appendix F.

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

5.4.2 Results

The results of the questionnaire filled out by the participants were analysed. Table 5.2 summarises of the result on surveillance.

Table 5.2 Result of survey on surveillance

Situation	% of users who would like to watch others' events			
	Login status	Message reading	Message sending	
when I post a question on public discussion forum	10%	20%	20%	
when I ask for help	40%	30%	30%	
when I want to discuss a question posted online with others	70%	50%	50%	
during a group discussion	30%	40%	60%	
when I want to know when/ how long a person is online	20%	N/A	N/A	
	% of users who would like to watch others' events			
Whom to watch	Login status	Message reading	Message sending	
			by whom	to whom
instructor	40%	40%	30%	10%
tutor	30%	10%	20%	N/A
the knowledgeable students they knew	40%	10%	20%	10%
group members	40%	50%	30%	30%
any one	30%	40%	50%	20%

Participants seemed most interested in watching the events of group members or participants with whom they actively share information. In addition, half of the participants indicated that they would like to be notified any time someone replied to one of their actions.

Results of privacy survey questions:

A number of questions about privacy preferences were answered by the participants. The following responses indicate that they had some concerns about others watching their actions, but that the concerns were not too serious.

- 60 % of the participants sometimes cared about other people being aware their login activity, 20 % never cared about it and 20% always were concerned.
- 10% of the people felt little uncomfortable when the message receiver watching their activities of sending messages, 90% of the people felt comfortable and none of the people felt nervous or angry.
- 10% of the participants felt nervous when the people other than the receiver watching their activities of sending messages, 90% felt just a little bit uncomfortable
- 40% of the participants sometimes cared about other people being aware their reading messages activity, 30% never cared about it and 30% always were concerned.

Result of comparison on interfaces of two approaches:

Table 5.3 summarizes the comparisons of the two approaches.

Table 5.3 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%

Three other questions were used in comparing the two approaches and the result are summarized as follows:

- 30% of the users would prefer to watch more events than the system currently provided.
- 90% the participants would like to be able to generate some rules that are not provided by the rule template. Of these, about half would like to do so using an interface like ARMS.
- 40% of the participants felt they might be tempted to write some rules that might cause harm to the system or other users might find worrisome.

Results of general questions on using I-Help programmable agents (Table 5.4):

Table 5.4 Results of general questions (percentage of subjects)

How the users felt about the usefulness of programmable agents in e-learning environment:			
Very useful	Helpful to some extent	Slightly helpful	Useless

20%	70%	10%	None
How do users prefer to find out the status of events happening in the system:			
Program their agents by creating simple rules	Check for some events (i.e. logged in) by themselves	Send email asking	Instruct by complex rules
60%	30%	10%	None
How would the users like to use the system in the future:			
More than before	Same as before	Less than before	
50%	50%	None	
Would privacy concerns prevent them from using the system:			
Not at all	To some extent	Yes	
40%	50%	10%	

In the interview, the subjects indicated the following activities would be useful:

- 1) On-line chatting
- 2) Maybe for tutor or instructor to watch participation of particular student in theme discussions
- 3) A privacy filter to block some users from watching you

5.4.3 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. The results demonstrate that students had better performance in the situation with awareness support. The subjects 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 to 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 the same as now if programmable agents were available. No one said that it was useless or they would stop using it. However the some of the results of this study were not so positive as the ones in the ARMS usability study. The following illustrate the differences between the answers to the questions on the usefulness of the programmable agents and how this will affect their usage in the future.

- In the ARMS usability study, 80% of the people felt it was very useful and 20% of the people felt it was helpful to some extent, but in the comparative study, only 20% of the people said it was very useful while 70% of people said it would be helpful to some extent.
- In the ARMS usability study 100% of the people said they would use the I-Help system more with programmable agents, while in comparative study, 50% of the people said they would use the system more than before and 50% of the people said they would use the system same as before.

One reason for the users' attitudes being more positive in ARMS usability study may be because of the actual use of the system and the perceived benefits of the effects of the programmable agents. The subjects felt that the programmable agents in ARMS provided great opportunities in facilitating them to communicate / cooperate effectively and efficiently. We suspect that the people might feel the programmable agents more useful if they had actually used them to fulfil some tasks in the comparative experiment. Another reason for the less positive result in CLIPS may be because of the complicated syntax in configuring rules with CLIPS approach. Some of the people felt it would not be so simple to program their agents using the CLIPS approach.

Table 5.3 shows the comparison result of the two approaches. For the question "How easy is it to program the agent (configure a rule)", 70% of the people chose "easy to understand/operate without help" and 30% of the people chose "it's easy with a little

help” for ARMS approach. No one felt ARMS was confusing or hard to understand/operate. For the CLIPS approach, the majority of people (60%) felt it was confusing to understand/operate without help and 40% felt it would be easy to understand/operate with little or no help. The numbers illustrate that both approaches can be understood and operated by users, however ARMS is easier to be understood/operated than CLIPS.

For the tasks that are available in both approaches, 90% of the subjects preferred to use ARMS and only 10% preferred to use CLIPS. People prefer to use an interface that is easy to understand and operate.

Fewer of the subjects (30%) had a desire to monitor more events than the system provided. When asked to generate rules that are not in the rule template, most of the subjects (70%) prefer to find a way using the ARMS approach or using predefined functions in the CLIPS approach. Only 20% of the people would like to write their own rules. This result may arise because of the complicated syntax of CLIPS and it also depends on who the subject is. If the subjects are from computer science or have some experiences in computer programming, they might feel CLIPS is not too hard.

When comparing which approach has more power, 80% of the subjects felt CLIPS would be more powerful than ARMS.

An interesting observation is that compared to the answers in the first ARMS usability study, more subjects 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 said “I would like to see how I can affect the system by writing my own rules”.

The belief on the powerfulness of the CLIPS approach also raised greater concern about system security and users’ privacy. In comparing system security and users’ privacy, the majority of people felt that ARMS was more secure than CLIPS and no one felt that ARMS would be dangerous. On the other hand, 30% of the people felt that the CLIPS approach might be dangerous. In answer to the question “do privacy concerns prevent you from using the system”, 50% of the people indicated that privacy concerns would

discourage them from using the system to some extent. This number is much higher than the corresponding response in the first ARMS usability study. Their concerns might be caused by the belief of the powerfulness of the CLIPS and less control over other users' agents.

In the interview, one user suggested that a user should have a privacy filter to block certain users from watching him/her. Actually this is similar as the idea we stated in section 5.3.3. It is desirable to allow users to restrict who has access to some events. It is also essential for users to know and control who may be watching, what they are watching, and how to protect themselves from stalking or inappropriate surveillance. "Making it available for tutors or instructors to watch participation of particular student in theme discussions will be very useful and practical in the online learning environment", one of the users stated in the interview. These valuable suggestions and comments will form directions for our future research.

5.5 Summary

The results from two studies on user programmable agents were presented in this chapter. The results show that the both the ARMS and CLIPS approaches are useful and helpful, but they each has their own strengths. The CLIPS version was seen to be more powerful but harder to use. The ARMS approach was considered to be easy to use and provides sufficient functionality to the users. Subjects preferred to use the ARMS approach over CLIPS.

CHAPTER 6

CONCLUSIONS

6.1 Summary of Thesis Work

Software agents provide great opportunities to assist users with routine, repetitive, and time-consuming tasks in various educational environments. An agent could enter into negotiations, acting independently to help achieve the user's goals in an unpredictable environment, and communicate with the user because of its autonomy, reactivity, and pro-activity properties. However, it is also these properties, particularly autonomy that raises significant challenges in human-agent interaction. This thesis explores some of the aspects of agent-based systems and issues in human agent interaction, and develops an end user programming environment that focus on delegating tasks to an agent. This thesis also presents the detailed design and the implementation of the end user programming environment within the bounds of a multi-agent learning environment named I-Help.

The I-Help system is an online learning environment that provides just in time help for students over the internet. There are public discussion forums and private discussion rooms for the students getting /asking questions, and sharing their knowledge. The programmable agents were developed to facilitate the information exchange among the users, and enhance the communication between users within the virtual learning environment.

In this thesis 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. One of the approaches is called ARMS that is to add to each agent in I-Help a simple customized rule system. ARMS was built as a mechanism where users could add capacity to their agents. In order to evaluate the suitability of ARMS, we built a second implementation, CLIPS-based agents that involve connecting a rule based expert system shell to each personal agent in I-Help. The CLIPS approach provides a more powerful agent programming system, but is more complex and potentially less robust because users can more easily write buggy rules or cause infinite loops.

Two experiments with human subjects were designed and conducted that included a usability experiment on ARMS enabled programmable agent environment and a comparative user study between the ARMS approach and the CLIPS approach. A usability experiment on ARMS agent environment was designed to observe the behaviors of subjects during the experiments, and to evaluate the helpfulness and difficulty of agent programmability. A comparative study between two approaches was designed 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 or a simpler customized rule system. In both experiments the users were asked for their concerns on system security and users' privacy.

6.2 Research Contributions

The primary research contribution of this thesis is the creation of an end user programming environment that supports individual and collaborative learning in a collaborative learning environment. The result of the ARMS usability study demonstrates that: (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.

The second contribution is that it provides a platform for finding an optimal solution for human agent interaction, which especially focuses on delegating tasks, instructing agents and trust between the human and agents. In this thesis we proposed two approaches for building end user programming environment and let users to chose which approach has more power, which one they prefer to employ and so on. The result of the usability study and comparative study shows that (1) the programmable agent is an optimal solution for delegating tasks to the agent and (2) the people prefer to use an easier way to instruct and delegate tasks.

The third contribution of this thesis is that it 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.

Finally, this thesis has a contribution in customizing individual needs based on activity information. The agent programmability approach solved the problems of how to deal with different users' needs about awareness, and how to cope with changing interests. We predict that awareness customization will augment collaboration opportunities naturally and efficiently in online collaborative learning and working environments.

6.3 Future Research

In the future, more work/functionality needs to be done to the I-Help agent programming environment. Long-term plans for I-Help include making more system events available to users, developing programmable anti-spy agents, and making I-Help programmable agents part of other integrated online virtual communities.

❖ Making More System Events Available to Public

More system events need to be made available so that users could use this information to increase their awareness of others in order to cooperate in the learning community. One useful event might be the update on knowledge level. The update can be done by two ways: the users update their knowledge level by themselves or the system update a user's knowledge level according to his/her performance factors, such as grades in relevant courses, helpee's feedback, and self-assessment, all contribute to a more complete model of learner. It is important to make this information up to date, so that the students can get accurate information about knowledgeable peers and adjust their target helpers to ask for help in time.

Another useful event for users is “when someone answer my question posted in the public discussion forum”. In our studies, users would like to be notified as soon as their questions are answered. A useful event for instructor or tutor might be providing a way for them to watch participation of particular students in theme discussions.

❖ **Developing Programmable Anti-spy Agents**

By analyzing the features of the privacy concern in I-Help and related research (PISA, 2000; Kobsa, 2001, 2002; Schreck, 2000) we found several actions should be taken in order to protect users' privacy:

- The users in I-Help should know about the information that is collected and how it is used. The information about their user identification, password, birthday, email address, etc should not be exposed to other users. Some of the activities information of the I-Help users might be shared with other users within the I-Help scope. This should include logging into the system, reading a message, sending a message, and so on.
- The users should have control over their information. They should be able to choose what activities they want to share with others and with whom they wish to share the information. The research on user privacy in I-Help is underway and a privacy server for an agent-based learning environment is being built in I-Help. In order to protect the user's privacy, each user will be able to set up their privacy preference. Each personal agent maintains the information about who has the privilege to know the owner's activities and provides this information to the Privacy Server. When someone wants to know the others' activities in the system, the personal agent attempts to retrieve the information from the event stream. The Privacy Server then creates a special event stream for the agent that contains only those events the agent has a right to see (Kettel, 2003).
- It is desirable to allow users to restrict who has access to some events. It is also essential for users to know and control who is watching, what they are watching, and to protect themselves from stalking or inappropriate surveillance. One interesting direction for our future research is the development of a programmable

anti-spy agent. It 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.

❖ **Integrate with Other Online Virtual Communities**

The programmable agents work very well in the IHelp learning environment to help users to become aware of system events and users' online activity and to locate and access help resources. 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 being developed at the University of Saskatchewan. The agents will help users on acquiring system events and other users' activities. The programmable agents also will be useful in restricting the flow of information through the system to the agents or applications that are looking to model the user. Agent programming techniques can be adapted to other virtual communities.

6.4 Conclusion

This research has demonstrated that our agent programming environment is able to meet users' individual needs for awareness information, facilitate information exchange among users, and enhance the communication between users within the virtual learning environment. Two variations for agent programmability were built on top of the IHelp system and the comparative study shows that the Agent Rule Management System (ARMS) is a preferred solution for delegating task to the agent.

This research work 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. Further research should be done to make the system more useful and secure.

REFERENCES

- AARP (2000). AARP National Survey on Consumer Preparedness and E-Commerce: A Survey of Computer Users Age 45 and Older.
<http://research.aarp.org/consume/ecommerce.pdf>
- Ackerman, M. S., and Cranor, L. (1999). "Privacy Critics: UI Components to Safeguard Users' Privacy", In Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'99), 258-259.
- Allen, J.F., Byron, D.K., Dzikovska, M., Ferguson, G., Galescu, L., and Stent, A.(2001) Toward Conversational Human-Computer Interaction. *AI Magazine* 22(4), 27-37.
- Andre, E., and Rist, T. (1996) Coping with Temporal Constraints in Multimedia Presentation Planning. In Proceedings of the Eighth National Conference on Artificial Intelligence, Menlo Park, Calif.: American Association for Artificial Intelligence. 142-147.
- Andre, E., Rist, T., and Muller, J. (1999). Employing AI Methods to Control the Behavior of Animated Interface Agents. *Applied Artificial Intelligence* 13 (4-5), 415-448.
- Bauer, M., and Dengler, D. (1999). InfoBeans – Configuration of Personalized Services. In Maybury, M., ed., *Proceeding of the 1999 International Conference of Intelligent User Interfaces (IUI' 99)*, 153-156.
- Bauer, M., Dengler, D., and Paul, G. (2000) Communication between Trainer and Agent in Programming by Demonstration. *Proceedings of the AAI Fall Symposium 2000*, North Falmouth, Massachusetts, USA.
<http://www.dfki.de/~bauer/fs2000/proceedings.html>
- Boicu, M., Marcu, D.m Bowman, M., and Tecuci, T (2000) A Mixed-Initiative Approach to Teaching Agents to Do Things. *Proceedings of the AAI Fall Symposium 2000*, North Falmouth, Massachusetts, USA.
<http://www.dfki.de/~bauer/fs2000/Proceedings/marcu.pdf>
- Cao, Y., & Greer, J (2003 a). Supporting Awareness to Facilitate Collaborative Learning in an Online Learning Environment. *Proceedings of the Computer -Supported Collaborative Learning (CSCL 2003)*, Bergen, Norway. 183-187.
- Cao, Y., & Greer, J. (2003b) Agent Programmability in a Multi-Agent Learning Environment. *Proceedings of the 11th International Conference on Artificial Intelligence in Education*, Sydney, Australia. 297-304.
- Chin, D.(1991). *Intelligent Interfaces as Agents*. In *Intelligent User Interfaces*. J. Sullivan and S. Tyler (eds). ACM Press, New York. 177-206.

- Cranor, L.F., Reagle, J., and Ackerman, M.S. (1999). Beyond Concern: Understanding Net Users' attitudes About Online Privacy, Technical Report, TR 99.4.3, AT&T Labs - Research. <http://www.research.att.com/resources/trs/TRs/99/99.4/99.3/report.htm>
- Culnan, M.J., and Milne, G.R.(2001) The Culnan-Milne Survey on Consumers & Online Privacay Notices: Summary of Responses. <http://www.ftc.gov/bcp/workshops/glb/supporting/culnan-milne.pdf>
- Cypher, A. Eager (1993): Programming Repetitive Tasks by Demonstration. In Watch What I Do: Programming by Demonstration. MIT Press, Cambridge, MA, 205-217.
- Dickinson, L.(1998) Human-Agent Communication. <http://www.hpl.hp.com/techreports/98/HPL-98-130.pdf>
- Deters, R. (2000) Developing and deploying a multi-agent system, Proceedings of Autonomous Agents'2000, Barcelona, Spain. 175-176.
- Dieberger, A. (1997) Supporting Social Navigation on the World Wide Web. International Journal of Human-Computer Studies, 46 (6), 805-825.
- Downs, S. (1998) The future of online learning. <http://www.atl.ualberta.ca/downes/future/home.html>
- EPIC (2000) "Pretty Poor Privacy: An Assessment of P3P and Internet Privacy", "Electronic Privacy Information Center", June 2000. <http://www.epic.org/reports/prettypoorprivacy.html>
- Etzioni, O., and Weld, D. A.(1994). Softbot-Based Interface to the Internet. Communications of the ACM 37(7), 72-76.
- Finin, T., Labrou, Y., and Mayfield, J.(1997) KQML as an Agent Communication Language, MIT Press. Cambridge, MA, USA. 291-316.
- Foner, L. N. (1997)Yenta: A Multi-Agent, Referral-Based Matchmaking System. In The First International Conference on Autonomous Agents (Agents '97), Marina del Rey, CA. 301-307.
- Forgy, C. (1982). Rete: A fast algorithm for the many patterns / many objects match problem. Artificial Intelligence, 19 (1), 17-37.
- 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.

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.

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.

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

Gustafson, T., Schafer, J.B., and Konstan, J.(1998) Agents in Their Midst: Evaluating User Adaptation to Agent-Assisted Interfaces. In Proceeding of the 1998 International Conference on Intelligent User Interfaces. New York: Association of Computing Machinery. 163-170.

Gutwin, C., Stark, G., and Greenberg, S. (1995) Support for Workspace Awareness in Educational Groupware. Schnase, J. L., Cunnius, E.L. (eds.) Computer Support for Collaborative Learning. Proceedings of CSCL'95. The First International Conference on Computer Support for Collaborative Learning, New York: Lawrence Erlbaum Associates, 147-156.

Haubl, G., and Trifts, V. (2000). Consumer decision making in online shopping environments: The effects of interactive decision aids. Marketing Science, 19(1), 4-21.

Horvitz, E. (1999) Principles of Mixed-Initiative User Interfaces. Proceedings of CHI '99, ACM SIGCHI Conference on Human Factors in Computing Systems, Pittsburgh, PA, ACM Press. 159-166.

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.

Jennings, N.R., Faratin, P., Norman, T.J., OBrien, P., and Odgers, B. (1999) Autonomous Agents for Business Process Management : in International Journal of Applied Artificial Intelligence., 14(2),145-189.

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-cscl/Papers/197.pdf>

Kay, A.(1984) Computer Software. In: Scientific American. 251(3), 41-47.

- Kettel, L.(2003) A Privacy Server for an Agent-based Learning Environment. Master Thesis, Department of Computer Science, University of Saskatchewan, Canada.
- Kobsa, A. (2001) Tailoring Privacy to Users' Needs (Invited Keynote). In M. Bauer, P. J. Gmytrasiewicz and J. Vassileva, eds.: User Modeling 2001: 8th International Conference. Berlin - Heidelberg: Springer Verlag, 303-313.
<http://www.ics.uci.edu/~kobsa/papers/2001-UM01-kobsa.pdf>"© Springer Verlag
- Kobsa, A. (2002) Personalized Hypermedia and International Privacy. Communications of the ACM 45(5), 64-67.
<http://www.ics.uci.edu/~kobsa/papers/2002-CACM-kobsa.pdf>.
- Kozierok, R. and Maes, P.(1993) A Learning Interface Agent for Scheduling Meetings, ACM SIGCHI International Workshop on Intelligent User Interfaces, ACM, Orlando, Florida. 81-88.
- Kruchten, P. (1995) Architectural Blueprints—The “4+1” View Model of Software Architecture. IEEE Software 12(6), 42-50.
- Kurhila, J., Miettinen, M., Nokelainen, P., and Tirri, H.(2002). Educo- A Collaborative Learning Environment based on Social Navigation. in Proceedings of the Elearn 2002 Conference, Montreal, Canada, 1738-1741.
- Lai, K-Y., Malone, T.W., and Yu, K-C. (1998) Object Lens: A 'spreadsheet' for cooperative work, ACM Transactions on Office Information Systems, 6(4), 332-353.
- Lashkari, Y., Metral, M., & Maes, P. (1994) Collaborative Interface Agents, AAAI's 94, 444-449.
- Lieberman, H. (1997). Autonomous Interface Agents. In Proceedings of CHI 1997, 67-74.
- Lieberman, H., and Selker, T. (2003) Agents for the User Interface, in Handbook of Agent Technology, Jeffrey Bradshaw, ed., MIT Press.
http://web.media.mit.edu/~lieber/Publications/Agents_for_UI.pdf
- Maes, P. (1994). Agents that reduce work and information overload. Communications of the ACM 37(7), 30-40.
- Milewski, A.E. and Lewis, S.H. (1997). Delegating to software agents. International Journal of Human-Computer Studies, 46(4), 485 -500.
- 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.

- Munro, A.J., Sanger, D.J., and Benyon, D.R. (1999). Personal and Social Navigation of Information Space, 1st edition. Springer: London. 284pages.
- Nardi, B. A., Miller, J. R. and Wright, D. J. (1998) Collaborative, Programmable Intelligent Agents. Communications of the ACM. 41(3), 96-104.
- Negroponte, N.(1970). The Architecture Machine; Towards a More Human Environment, MIT Press.
- Novick, DG., and Sutton, S. (1997) Mixed initiative dialogue. In Proceedings of the 1997 AAAI Symposium. AAAI Press, 114-116.
- Ogata, H., Matsukuma, R., and Yano, Y. (1998). Supporting Awareness for Augmenting Participation in Collaborative Learning. Proceedings of ED-Media 98, Germany, 1040-1045.
- O'Neill-Brown, P. (1997). Setting the stage for the culturally adaptive agent. In 1997 AAAI Fall Symposium. AAAI Press, 93-97.
- Payne, T. R., and Edwards, P. (1997) Interface Agents That Learn: An Investigation of Learning Issues in a Mail Agent Interface. Applied Artificial Intelligence 11(1), 1-32.
- Payne, T.R., Singh, R., and Sycara, K. (2002) Calendar Agents on the Semantic Web. IEEE Intelligent Systems, 17(3), 84-86.
- P3P (2000) Platform for Privacy Preferences project: see <http://www.w3.org/P3P/>
- Petherick, W. (1999) Cyber-Stalking: Obsessional Pursuit and the Digital Criminal. In The Crime Library. <http://www.crimelibrary.com/criminology/cyberstalking/>.
- Pew Internet & American Life Project (2000). Trust and Privacy Online: Why Americans Want to Rewrite the Rules. <http://www.pewinternet.org>
- PISA (2000) Privacy Incorporated Software Agent project:
<http://www.tno.nl/instit/fel/pisa/>.
- 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.
- Raj, G.S. (1998) Common Object Request Broker Architecture.
<http://my.execpc.com/~gopalan/corba/corba.html>
- Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl,J.(1994) GROUPLENS: An Open Architecture for Collaborative Filtering of NETNEWS. In Proceedings of the

- ACM 1994 Conference on Computer-Supported Collaborative Work, New York: Association of Computing Machinery, 175-186.
- 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.
- Rickel, J., and Johnson, W.L. (2000). Task-Oriented Collaboration with Embodied Agents in Virtual Worlds. In *Embodied Conversational Agents*, eds.J. Cassell, J. Sullivan, S. Prevost, and E. Churchill. Cambridge, Mass.: MIT Press, 95-122.
- 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*. Berlin:Springer-Verlag. 77-93.
- Schreck, J. (2000) Security and Privacy in User Models. PhD Thesis, Department of Mathematics and Computer Science, University of Essen, Germany.
<http://www.jschreck.de/sapium/>
- Segal, R. and Kephart, J.O. (2000) Incremental Learning in SwiftFile. In *Proceedings of the Seventh International Conference on Machine Learning*, 863-870.
- Sharon, T., Lieberman, H., and Selker, T. (2002) Searching the Web with a Little Help from Your Friends, *ACM Conference on Computer-Supported Cooperative Work*, New Orleans. http://web.media.mit.edu/~lieber/Publications/Little_Help.pdf
- Shearin, S., and Lieberman, H. (2001) Intelligent Profiling by Example, *Proceedings of the International Conference on Intelligent User Interfaces (IUI 2001)*, Sante Fe, NM.
<http://web.media.mit.edu/~lieber/Lieberary/Apt-Decision/Apt-Decision.html>
- Shneiderman, B. (1983) Direct Manipulation: A Step Beyond Programming Languages, *IEEE Computer*, 16(8), 57-69.
- Shneiderman, B., and Maes, P. (1997) Direct Manipulation vs. Interface Agents. *Interactions* 4 (6), 42-61.
- Shoham, Y. (1993) Agent-oriented programming. *Artificial Intelligence*, 60(1), 51-92.
- Smith, D. C., Cypher, A., and Spohrer, J. (1997) KIDSIM: Programming Agents without a Programming Language. In *software Agents*, ed. J. Bradshaw, Menlo Park, California. AAAI Press. 165-190.
- Smith, D. C., Cypher, A., and Tesler, L. (2000) Novice Programming Comes of Age. *Commun. ACM* 43(3), 75-81.

Stanfill, C. and Waltz, D.(1986) Toward memory-based reasoning, *Comm. ACM* 29(12), 1212-1228.

Terveen, L.G., and Murray L.T.(1996) Helping users program their personal agents. In *proceedings of ACM CHI 96 Conference on Human Factors in Computer Systems*. Vancouver, 355-361.

Westin, A.F. and Maurici, D.(1998). *E-Commerce & Privacy: What the Net Users Want, Privacy & American Business*, and PricewaterhouseCoopers LLP, New York.
<http://www.pweglobal.com/gx/eng/svcs/privacy/images/E-Commerce.pdf>

Wexelblat, A. (1999). *Footprints: Interaction History for Digital Objects*, in MIT Media Lab. MIT: Cambridge, MA.

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.

Vassileva, 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.

Appendix A: Materials used in the ARMS Usability Study

In this document, brief information on I-Help public discussion and private discussion is introduced, followed by a demonstration of the ARMS system, which describes what kind of activities the users can monitor and how to receive notifications, how to use the system, and so on.

A User Study on I-Help Programmable Agents, Surveillance and Privacy in Online Learning Environment

1. I-Help one-to-one discussion:

The I-Help one-to-one discussion system is a "peer help" system where the students share their knowledge with each other. In this system users are represented by "personal agents". The "personal agents" are designed to monitor their users' activity in this online learning environment, and to assist learners in locating help resources (human and electronic).

2. Objective of the study:

For I-Help online learning environment to provide effective and efficient usage, users should be able to acquire the knowledge about individuals and the events happening in the system. In order to meet users' individual needs, facilitate the information exchange among the users, and enhance the communication between users within the virtual learning environment, an end-user programming environment ARMS (Agent Rule Management System) in I-Help is proposed. The I-Help ARMS allows users to monitor and analyze the events happening in the I-Help system by programming their agents. The agent acts on the user's behalf to execute a rule once the conditions of the rule are satisfied.

This research is to investigate how users behave when given the ability of programming their agents, what are the users' concerns about their privacy and how agent-based systems can be built to protect users' privacy, and whether the overall performance of the system will be affected with agent programmability.

The specific questions to be answered by this study are:

- How easy for users to program their agents?
- Will people feel helpful of agent programmability?
- Will users try to destroy the system and how to protect the system?
- What are the users' concerns about their privacy?
- Will I-Help capability improve with agent programmability?

3. How and what to Surveil?

With the I-Help programmable agents environment, a user is able to configure a simple “rule” to program his or her agent to communicate with other agents and to monitor the activities of other users and their agents. A rule will be similar as the one in Figure 1.

The following activities in this online environment are available for all users to watch:

- **Login:** when someone login to the I-Help one-to-one discussion
- **Logout:** when someone log our from the I-Help one-to-one discussion
- **ReadMessage:** when someone read a message posted on the public discussion forum
- **SendMessage:** when a person sends a message to another person within I-Help one-to-one

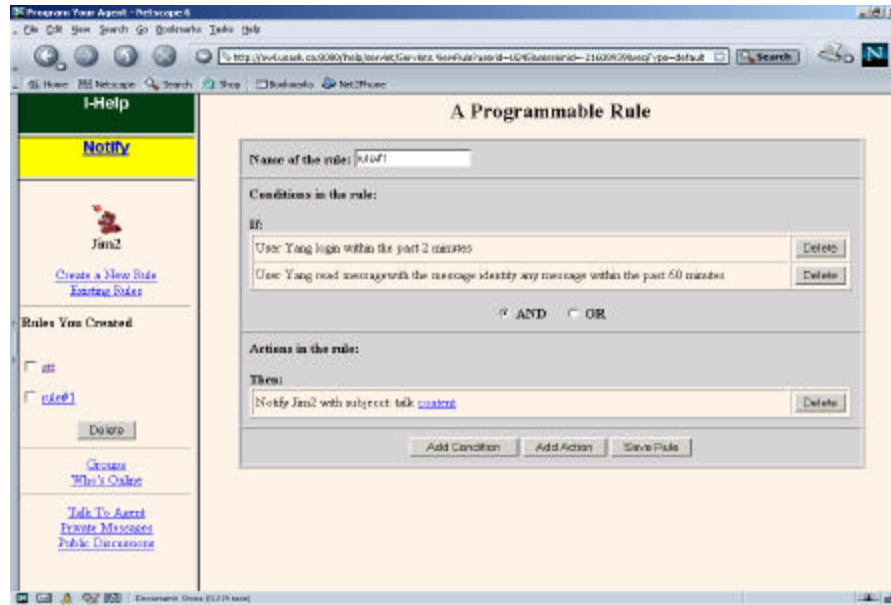


Figure 1. Rule Management Interface

Each rule contains a rule name, a set of conditions (If section), and a set of actions (Then section). Each condition has three or four parts: a user, activity type, and other parameters involved while each action has notification type, a user which need to be notified, notification subject, and content. The number of parameters and the context for parameters will vary from different activity types. Table 1 presents the various formations associated with different type of activities.

Table1. Formation of the conditions in the rule

	User	Activity type	Parameters	
Login	user A	logs in	within the past 2 minutes	_____
Logout	user A	logs out	within the past 2 minutes	_____
Read Message	user A	read message	with message identity	within the past 30 minutes
Send Message	user A	Send message	to user B	Within the past 30 minutes
Send Notification	user A	notify	user B	Within the past 5 minutes

A user can generate and modify a rule through a set of domain specific user interfaces. For example, the right section of the Figure 2 shows an example rule named read-message:

Rule name: read-message

If

User Sam reads the message with message identity 19291 within the past 60 minutes *and*
User Sam logs in to the system within the past 5 minutes

Then

Notify Sam with subject: discussion and content: "...we need to talk"

Email Yang with subject: read-message and content: "...Sam is ready to talk now..."

4. System Demonstration

The I-Help Agent Rule Management System currently works in the I-Help system helping users to monitor system events and others' activities and to respond according to users' preferences.

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).

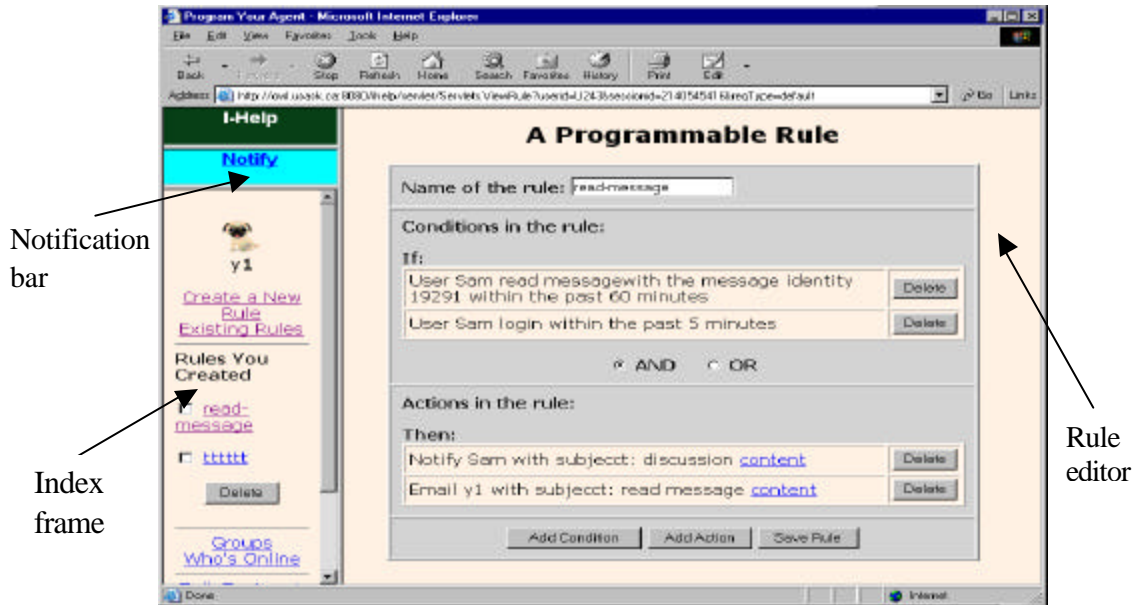


Figure 2: Rule Management Interface

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 Figure2) 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 rule are performed in the rule editor (the right part in Figure 2), condition specification (Figure 3), and action specification interfaces (Figure4).



Figure 3: Condition Specification

Each rule contains three parts as described previously. The users are able to add a new condition to the rule by clicking on the “Add Condition” button in the rule editor and this will open the condition specification window for the users.

Users specify rule conditions by selections on event type, the user, and other parameters involved. Figure 3 shows a condition describing “User Sam has read any message within the past 2 minutes”. Users can add an action in a similar way. Figure 4 presents an action which notifies Tom’s agent with subject discussion and content: “Sorry, I couldn’tuntil Sep30, 2002”. The conditions and actions will be displayed as an understandable English sentence in the rule editor after they are added to the rule. The functions of the rule editor also include the deletion of conditions /actions, and saving function.

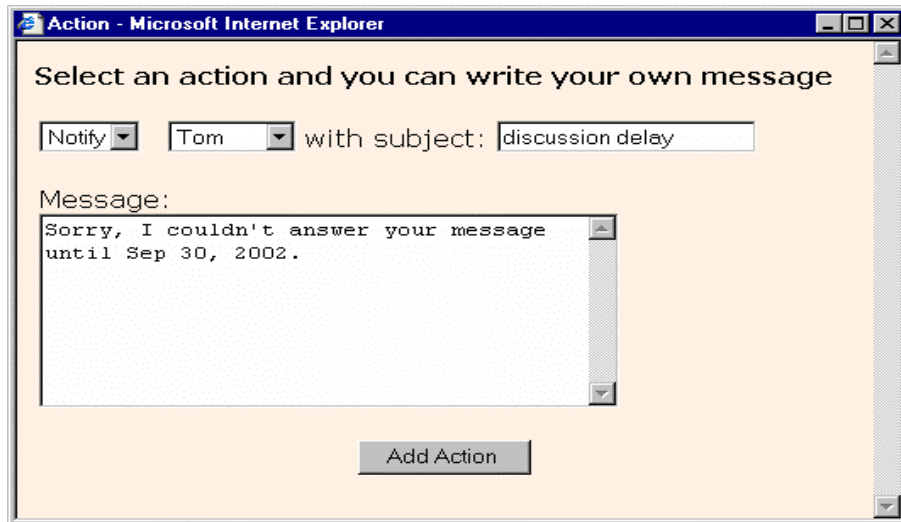


Figure 4: Action Specification Interface

Appendix B: The Questions in ARMS Usability Study

1. What is tutor George's favorite website?
2. Find the author for the article "Searching the Web with a Little Help from Your Friends"
3. What is the tutor Jennifer's favorite color?
4. Type a specific paragraph of a given newspaper
5. Schedule your group's meeting time
6. When is the time for meeting with tutor Eric?
7. One of the tutors or students knows a secrete answer to this question, what is the secrete answer?
8. Write a paragraph to describe the students in the class. One sentence is written for each student. Ask them what they would like to write for themselves or you can find it somewhere in the public discussion forum.
9. How many times people read the posting identity 1929 on public discussion forum?
10. Find the article "Intelligent Profiling by Example" online and find the page number for the phrase "profile expansion".

Appendix C: An Interview Form for First ARMS Usability Study

This document consists of a set of questions that were asked in the interview for first ARMS usability study, which includes views on surveillance, privacy concerns, and general questions on using the I-Help rule management system.

A User Study on I-Help Programmable Agents, Surveillance and Privacy in Online Learning Environment

Surveillance :

Login event:

1. In which situation you like to watch login event?

- a. I just post a question on public discussion forum
- b. I want to ask for help
- c. Want to discuss a question posted online with others
- d. Group discussion for group project
- e. Want to know when/ how long a person is online

2. Whom do you prefer to watch?

- a. instructor
- b. tutor
- c. knowledgeable students I know
- d. group members
- e. any one

Logout event:

1. In which situation you like to watch logout event?

- a. I just post a question on public discussion forum
- b. I want to ask for help
- c. Want to discuss a question posted online with others
- d. Group discussion for group project
- e. Want to know when/ how long a person is online

2. Whom do you prefer to watch?

- f. instructor
- g. tutor
- h. knowledgeable students I know
- i. group members
- j. any one

Read Message:

1. In which situation you like to watch read message event?

- a. I just post a question on public discussion forum
- b. I want to ask for help
- c. Want to discuss a question with others
- d. Group discussion for group project

2. Whom do you prefer to watch?

- a. instructor
- b. tutor
- c. knowledgeable students I know
- d. group members
- e. any one

Send Message:

1. In which situation you like to watch send message event?

- a. I just post a question on public discussion forum
- b. I want to ask for help
- c. Want to discuss a question with others
- d. Group discussion for group project

2. The message is sent by whom:

- e. instructor
- f. tutor
- g. knowledgeable students I know
- h. group members
- i. any one

3. The message is sent to whom:

- j. instructor
- k. tutor
- l. knowledgeable students I know
- m. group members
- n. any one

Privacy:

1. Do you like other people know when you log in to the system?

- a. not at all
- b. sometimes
- c. always

2. Do you care who watch your activities of sending messages?

2.1 if the receiver watches you:

- a. comfortable
- b. little bit uncomfortable
- c. nervous
- d. angry

1.2 if some person other than receiver watch you:

- a. comfortable
- b. little bit uncomfortable
- c. nervous
- d. angry

2. Do you care who serviel your activities of reading messages?

- a. not at all
- b. sometimes
- c. always

General questions on using the rule management system

1. Which way do you prefer when you want to know some events happening in the system?

- a. Check by myself for some events such as whether a person log in to the system.
- b. Send email directly for some events such as whether a person read a question or not.
- c. Program my agent by generating a set of simple rules.
- d. Instruct my agent by complex rules.

2. How easy to program your agent(configure a rule) ?

- a. Easy to understand /operate without help
- b. It's easy with a little help
- c. It's confusing to understand/operate without help
- d. It's very hard to understand/operate even with help

3. Do you feel it is useful in e-learning environment?

- a. Very useful
- b. It is helpful in some extent
- c. Little helpful
- d. Useless

3. What other activities will be useful?

4. How will you use the system when you have ability to program your agent? Why? (despite privacy concern)

- a. More than before
- b. Same as before
- c. Less than before

Because:

- 5. Will privacy concern prevent you from using the system?**
- a. Yes
 - b. in some extent
 - c. no at all

Appendix D: The Case Study Consent Form

A User Study on I-Help Programmable Agents, Surveillance and Privacy in Online Learning Environment

The I-Help one-to-one discussion system is a “peer help” system where the students share their knowledge with each other. The aim of this research is to provide an end-user programming environment in I-Help that facilitates the information exchange among the users, and enhances the communication between users within the virtual learning environment. This user study is to investigate whether such a system can facilitate collaborative learning, how users behave when given the ability of programming their agents, and what are the users’ concerns about the system security and their privacy

You will be asked to evaluate the feasibility of end user programmable agents and privacy and security of the system. The experiment included attending an introduction session, using the ARMS to fulfil a learning task, completing an exit questionnaire on feelings about the system and security and privacy concerns. The amount of time required for the entire experiment around 2 hours. Your test results will be used for evaluation of the I-Help end user programming environment, and for no other purpose. Test results will be kept confidential and your name will not be used for any purpose during the study: alias will be used instead.

Subject Consent:

I agree to be a subject for the I-Help end user programming environments study, according to the conditions described above.

Name:

Address:

Phone:

Email:

Signature:

Date:

Ethic approval: BSC #2001-198

Appendix E: Materials used in the Comparative Study

In this document, brief information on I-Help public discussion and private discussion is introduced, followed by demonstrations of the ARMS the CLIPS systems, which describe what kind of activities the users can monitor and how to respond, how to use each system respectively, with both the ARMS and CLIPS user interfaces .

A Comparative Study on Two approaches (the ARMS vs. CLIPS) of I-Help Agent Programming Environments

1. I-Help one-to-one discussion:

[text presented identical as show in Appendix A]

2. Objective of the study:

For I-Help online learning environment to provide effective and efficient usage, users should be able to acquire the knowledge about individuals and the events happening in the system. In order to meet users' individual needs, facilitate the information exchange among the users, and enhance the communication between users within the virtual learning environment, an end-user programming environment in I-Help is proposed. The I-Help programmable agent environment allows users to monitor and analyze the events happening in the IHelp system by programming their agents. The agent acts on the user's behalf to execute a rule once the conditions of the rule are satisfied.

This research is to investigate how users behave when given the ability of programming their agents, what are the users' concerns about the system security and their privacy and whether the overall performance of the system will be affected with agent programmability.

The specific questions to be answered by this study are:

- How easy for users to program their agents?
- Will people feel that agent programmability is helpful?
- Will users try to destroy the system and how to protect the system?

- What are the users' concerns about system security and their privacy?
- Can 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?

3.How and what to watch?

[text presented identical as show in Appendix A]

4. User Interfaces

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 1).

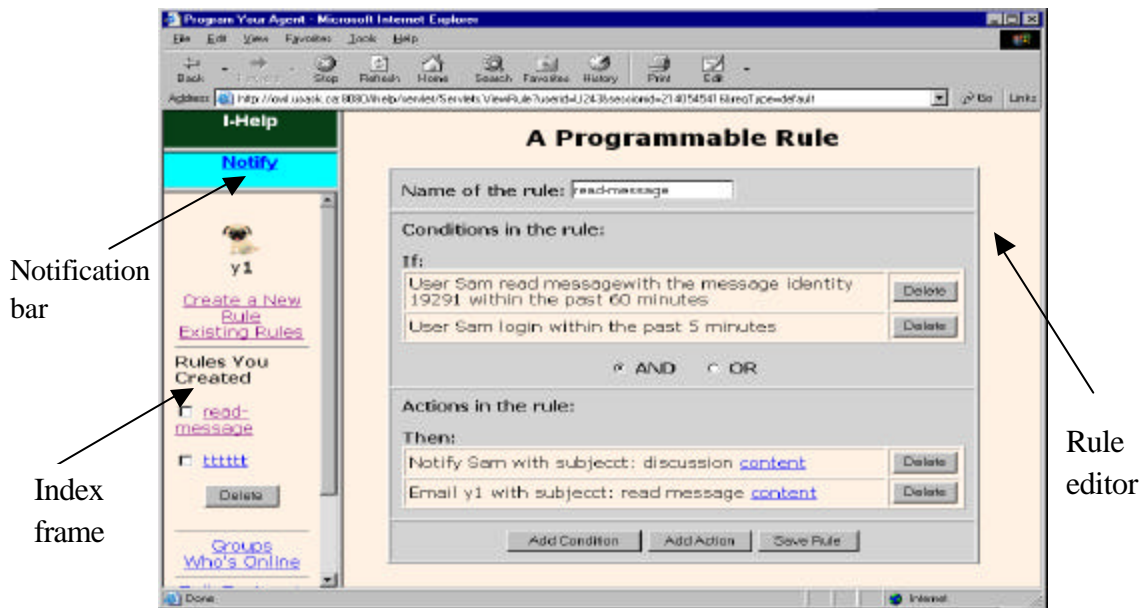


Figure 1.Rule Management Interface

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 1 shows that there is a new notification message for the user. The index frame (left part in Figure1) 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 rule are performed in the rule editor. There are two different approaches for users to edit a rule by using two sets of interfaces.

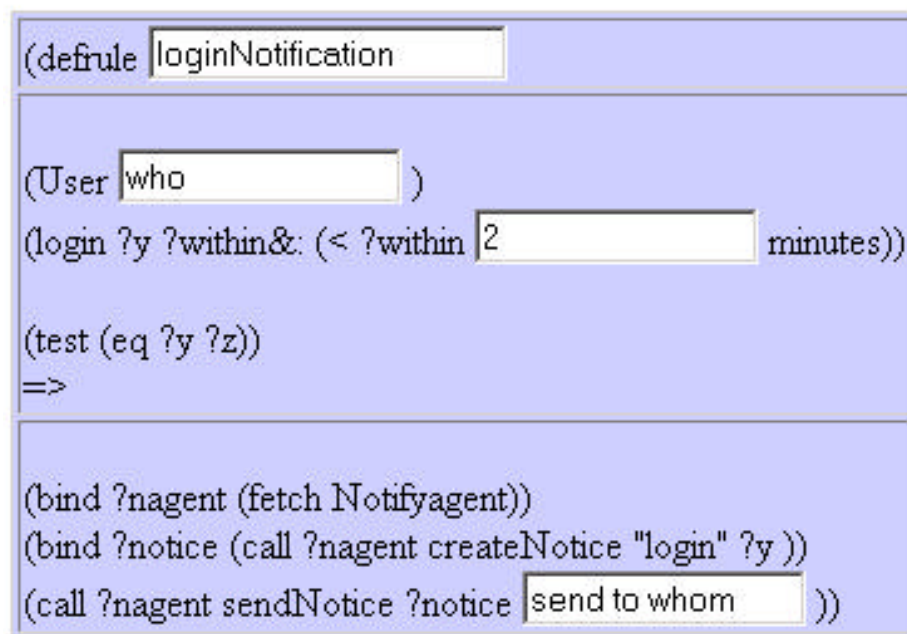
4.1 ARMS interfaces

[text and figures presented identical as show in Appendix A]

4.2 CLIPS interfaces

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 2 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.



The image shows a screenshot of a CLIPS rule template interface. The rule is named 'loginNotification'. The condition part consists of three lines: '(User who)', '(login ?y ?within&: (< ?within 2 minutes))', and '(test (eq ?y ?z))'. The action part starts with '=>' followed by three lines: '(bind ?nagent (fetch Notifyagent))', '(bind ?notice (call ?nagent createNotice "login" ?y))', and '(call ?nagent sendNotice ?notice send to whom)'. The text is displayed in a monospaced font on a light blue background.

```
(defrule loginNotification

(User who )
(login ?y ?within&: (< ?within 2 minutes))

(test (eq ?y ?z))
=>

(bind ?nagent (fetch Notifyagent))
(bind ?notice (call ?nagent createNotice "login" ?y ))
(call ?nagent sendNotice ?notice send to whom ))
```

Figure 2. An Interface for Login Notification Template

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 2). In addition to selecting and filling the value in a template, the users are able to make complex rules by

combining several events/actions. The system provides a simple Java API for users' references on some predefined functions so that users can write a rule like the example below. Of course the users have the option to define their own rules without using any functions provided by the system.

```
(defrule loginNotification
```

```
  (receiver ?x)
```

```
  (login ?y ?date ?time ? within&: (< ?within 2 minutes))
```

```
  (name ?z)
```

```
  (test (eq ?y ?z))
```

```
=>
```

```
  (bind ?nagent (fetch Notifyagent))
```

```
  (bind ?notice (call ?nagent createNotice "login" ?y ?date ?time))
```

```
  (call ?nagent sendNotice ?notice ?x))
```

Appendix F: An Interview Form for Comparative Study

This document consists of a set of questions that were asked in the interview for the comparative study, which includes views on surveillance, privacy concerns, comparison on interfaces of two approaches, and general questions on using the I-Help rule management system.

Questionnaire for Comparative User Study on Two Approaches

Surveillance :

[text presented identical as shown in Appendix C]

Comparison on interfaces of two approaches:

1. How easy to program your agent (configure a rule)?

1) ARMS approach:

- a. Easy to understand /operate without help
- b. It's easy with a little help
- c. It's confusing to understand/operate without help
- d. It's very hard to understand/operate even with help

2) CLIPS approach:

- a. Easy to understand /operate without help
- b. It's easy with a little help
- c. It's confusing to understand/operate without help
- d. It's very hard to understand/operate even with help

2. For the tasks that are available in both approaches, you prefer to use which approach:

- a. ARMS approach

- b. CLIPS approach

3. Which approach you think have more power:

- a. ARMS approach
- b. CLIPS approach

4. Have you ever had an intention to watch more events than the system provided currently?

- a. yes
- b. no

5. Will you like to generate some rules that are not in the rule template? And how?

- a. Yes, try to find a way using ARMS approach
- b. Yes, find a way using predefined functions in CLIPS approach
- c. Yes, try to write my own rules
- d. No

6. If you have the power, will you try to write some rules that will damage the system or other users?

- a. no
- b. maybe have a try

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

- a. ARMS is less secure
- b. CLIPS is less secure
- c. ARMS is dangerous
- d. CLIPS is dangerous

General questions on using the rule management system

1. Which way do you prefer when you want to know some events happening in the system?

- a. Check by myself for some events such as whether a person log in to the system.
- b. Send email directly for some events such as whether a person read a question or not.
- c. Program my agent by generating a set of simple rules.
- d. Instruct my agent by complex rules.

2. Do you feel it is useful in e-learning environment?

- a. Very useful
- b. It is helpful in some extent
- c. Little helpful
- d. Useless

3. What other activities will be useful?

4. How will you use the system when you have ability to program your agent? Why? (despite privacy concern)

- a. More than before
- b. Same as before
- c. Less than before

Because:

5. Will privacy concern prevent you from using the system?

- a. Yes
- b. in some extent
- c. no at all

Appendix G: The Case Study Consent Form

A comparative study on two approaches of I-Help end user programming environments

The I-Help one-to-one discussion system is a "peer help" system where the students share their knowledge with each other. The aim of this research is to provide an end-user programming environment in I-Help that facilitates the information exchange among the users, and enhances the communication between users within the virtual learning environment. This user study is to investigate how users behave when given the ability of programming their agents, what are the users' concerns about the system security and their privacy 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 customized and restricted rule system.

You will be asked to evaluate the feasibility of end user programmable agents and compare two approaches of I-Help end user programming environments. The experiment included attending an introduction session, comparing two approaches, and completing an exit questionnaire on feelings about the system and security and privacy concerns. The amount of time required for the entire experiment will not exceed an hour. Your test results will be used for evaluation of the I-Help end user programming environment, and for no other purpose. Test results will be kept confidential and your name will not be used for any purpose during the study: alias will be used instead.

Subject Consent:

I agree to be a subject for the I-Help end user programming environments comparative study, according to the conditions described above.

Name:

Address:

Phone:

Email:

Signature:

Date:

Ethic approval: BSC #2001-198