A Multi-Agent Architecture for Peer-Help in a University Course

Article · July 1998

Source: CiteSeer

CITATIONS
4

READS
85

6 authors, including:

Julita Vassileva
University of Saskatchewan
367 PUBLICATIONS **10,204** CITATIONS
SEE PROFILE

Ralph Deters
University of Saskatchewan
235 PUBLICATIONS **4,600** CITATIONS
SEE PROFILE

Jim E Greer
University of Saskatchewan
162 PUBLICATIONS **3,408** CITATIONS
SEE PROFILE

Gordon I McCalla
University of Saskatchewan
195 PUBLICATIONS **4,529** CITATIONS
SEE PROFILE

A Multi-Agent Architecture for Peer-Help in a University Course*

Julita Vassileva, Ralph Deters, Jim Greer, Gordon McCalla, Vive Kumar, Chhaya Mudgal
Department of Computer Science, University of Saskatchewan, Saskatoon S7N 2T5, Canada

Email: [jiv,ralph,greer,mccalla,vsk719,chm906]@cs.usask.ca

Abstract

I-Help is an integration of previously developed ARIES Lab tools for peer help to university teaching. In this paper we discuss an approach to distributing the centralised monolithic architecture of I-Help, by using a multi agent-architecture.

The I-Help project

I-Help is an integration of previously developed ARIES Lab tools for peer help to university teaching. One of its components, CPR provides a subject-oriented discussion forum and moderated FAQ-list supporting students with electronic help. Another component, PHelpS selects an appropriate peer helper who will support the student with direct peer help via an elaborate chat-environment. The selection of appropriate discussion forum, FAQ-article or human peer helper is based on modelling learner knowledge in the context of the concept / topic structure of the subject material. A full paper describing the I-Help Project will be presented at ITS'98 (Greer et al., 1998).

In this paper we discuss an approach to distributing the centralised monolithic architecture of I-Help, by using a multi agent-architecture, presented in (Vassileva, 1997). This presentation shows how this architecture can be applied in I-Help and gives some technical data about the current state of implementation.

The agent-based approach has been motivated by the computational challenges we faced during the recent testing of I-Help in a university class with about 400 students. It turned out that a centralised approach imposes an immense load on the database (ORACLE) connection with the Web-based interface (via CGI scripts and PERL). Since all student data and domain knowledge structures were stored in the database, each help request generated numerous database queries and it sometimes took an inordinate amount of time to generate a response. This "unsuccessful" experiment taught us a good lesson: that it is nearly impossible to scale up a centralised solution using the web with an underlying database and that we have to strive to develop distributed solutions. A Java – based agent architecture offers an attractive alternative for several reasons: it naturally provides a threading mechanism, which is very appropriate for multi-agent applications, it is integrated in all current web-browsers and provides a more elegant and flexible programming environment than CGI. In addition, JAVA has good database connectivity with all leading databases (through JDBC).

Apart from this "technical" motivation for distributing the architecture of I-Help, we had another, deeper, AI-oriented motivation. We have been looking for some time into multi-agent systems and we have found a number of extremely interesting and potentially useful research issues concerning inter-agent negotiation, persuasion and competition in agent societies. I-Help provides an excellent environment for studying these issues and the emerging issues related to agent-based user modelling and agent based-teaching.

A distributed, agent-based architecture reflects naturally the distributed web-based environment in I-Help. There is no virtual difference between humans and software agents. It is no longer advantageous to have a monolithic ITS, which is like an almighty teacher knowing the answer to any question or learning need that may arise in the learner. Networking provides a possibility to find some narrow-focussed ITS or other learning resource, suitable for the domain of the question or the learning need. This might take the form of some courseware, or a human-partner who can help the learner with the problem and explain things. In this way a human enters the teaching loop to compensate for limitations that are by necessity related to educational software or ITS.

It is no longer a necessary condition that there is a powerful diagnostic component to perform individualised teaching. A human can "enter this loop" too and help in diagnosing the problem that the learner is facing. This diagnosis can be later used by an ITS or by another human, taking the role of a helper to provide the learner with advice. In this way an agent-architecture provides for a natural synergy between humans and software agents (ITSs, diagnostic components, pedagogical expert systems, on-line help systems, web-based pools of on-line materials, etc.)

The multi-agent architecture

A relatively detailed description of the multi-agent architecture we propose is presented in a full paper at ITS'98 (Vassileva, 1998). In this workshop we'll sketch only the main idea and will provide more technical details of the implementation of the architecture. Users, learners, applications and learning environments are represented by

Published in Proceedings of ITS'98 Workshop on Pedagogical Agents, San Antonio, TX, pp.64-68.

autonomous goal-based social agents who communicate, co-operate, and compete in a multi-system and multi-user distributed environment (see Figure 1). Each agent maintains three distinct models of the user / application which it represents:

- a model of the user's/application's goals (in the case of human users – the goals and the preferences of the user),
- a model of the resources available to the user / application, which in the case of a user, can include his / her cognitive resources (i.e. there are several model domain specific models of the user's knowledge, experience, skills etc.),
- a model of the relationships of the user or application (like personal friends, acquaintances, agents which have proven to be useful, or who have offered help and the user/application is "indebted" to).

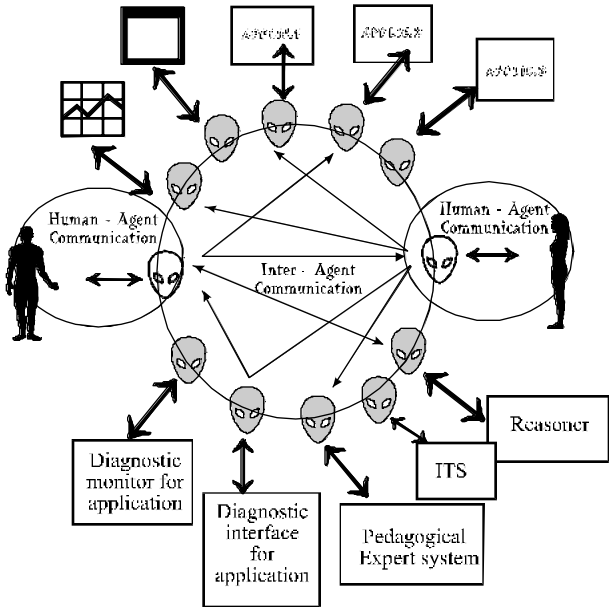

Figure 1: Agent-based architecture

These models allow the agent to reason about user goals, to plan about the user resources and to search for agents of other users or applications who have the same goal and/or possess the resources that the user doesn't have.

A personal agent knows about the current goal (or preferences) and state of resources (knowledge) of the user either directly (when the user poses a help-request as is done currently by I-Help) or through some application-specific diagnostic component or task-based interface like those described in (Collins, et al., 1997), (Vassileva, 1996).

Such diagnostic application is represented by its own application-agent and the personal agent has to negotiate with this agent in order to use the resources/services offered by the diagnostic application. If no diagnostic application or interface is available, the personal agent may request diagnosis from another human user by contacting the user's personal agent. In case this agent agrees (it may ask the user for agreement), it will find and contact the agent of a domain-specific interface for human "diagnosers" to perform the service of providing for a description of the problem and accepting the diagnoser's input such that can be entered automatically in the user model for this domain.

Once the user's goal and state of resources is diagnosed and entered into the user model, the personal agent has to decide how to assist best the user in achieving his/her goal. For example, let's assume that the personal agent discovers that a user is pursuing a goal (e.g. solving a programming problem) in a non-optimal way (that means the user lacks some resource, in this case – knowledge). The personal agent may decide to teach the learner (instruct, explain, provide help, etc.) by requesting a teaching service offered by the agent of some educational software (an ITS or intelligent help-component, web-based materials, a FAQ-directory, or discussion forum) or by contacting the personal agent of another knowledgeable user to request human help. The personal agent may also decide to request pedagogical advice about which of these options is optimal. For this reason it will contact the application agent of a pedagogical expert system and negotiate this service.

So in summary, in this architecture users are represented by personal agents who pursue goals which can be assigned explicitly by the user, or inferred / diagnosed and represented in a user model.

There are two major aspects in the realising such a multi-agent system:

- Implementing the Learner / User – Agent Interaction Sphere (Figure 2), and
- Implementing the Agent – Agent Interaction Sphere (Figure 3)

Issues in Implementing the Learner / User – Agent Interaction Sphere

The personal agent can have its own goals that differ from the user's goals, for example, to be responsive and co-operative to other agent's requests. Sources of "motivation" for the agent to pursue such goals can be realised by introducing notions of cost and utility of service, and an embedded goal in all agents to minimise the costs and maximise utility, i.e. to maximise the "profit", (i.e. the difference between utility and cost), by offering services for other agents.

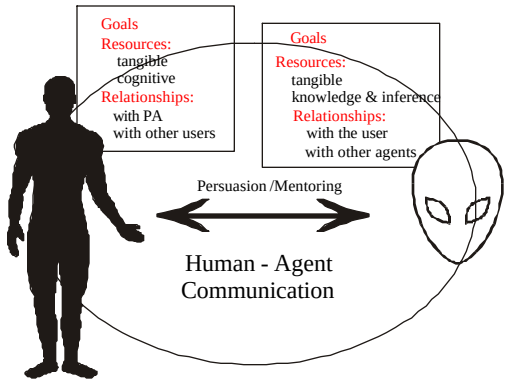

Figure 2: The Learner / User – Agent Interaction Sphere

A global measure of utility can be a fictitious "currency" introduced in the agent society. The services can be resources belonging to the user represented by the agent. In this way the personal agents become not only reactive (i.e. responding to user's requests), but proactive and autonomous. Such an agent plans and acts to achieve goals that may not be explicit user goals, but are goals of the personal agent itself. Since the users / learners are also an autonomous agents pursuing their own goals, in conflict situations, the decision of which and whose goals will be pursued (the personal agent's or the learner') is made interactively, in a process of negotiation and persuasion between the user and the personal agent.

ITS and other teaching applications are represented with autonomous application-agents are pursuing specific embedded goals. There are two types of goals pursued by pedagogical agents. The first type of goals is common to all agents; they concern the agent's interaction with the agent society. For example, to maximise the agent's profit by offering specific type of services, in this case – teaching services. The second type of goals is specific for agents of teaching applications; they concern the interaction between the human and the application; they are teaching goals. These goals can be cognitive (subject- and problem-specific), motivational, and affective (learner- and subject-specific). These goals have to be adopted by the human user who is not necessarily motivated by profit maximisation like the agents in the agent society, so the user may choose not to adopt the goal, even though it would be beneficial for him/her. For example, if the personal agent (with the help of a diagnostic agent) has found out that the user is performing some task in a certain domain not optimally, it may contact the agent of appropriate teaching software to provide a short tour or explanation to the user. However, the user may not feel as doing anything wrong and may decide not to spend time for learning, but to continue working on his/her main task. Of course ultimately the user has the final word, but we feel that a personal agent and a teaching agent have to be able to "defend" their suggestion, i.e. to be able to try to persuade the user in the benefits of adopting the new goal.

We believe that persuasion is a key issue in the interaction between a pedagogical agent and a human user. There have been numerous persuasion strategies developed in social sciences and AI (Shank & Abelson, 1977), and some of them have a striking similarity to strategies employed for increasing learner's motivation in teaching. In order to implement persuasion capabilities in agents we envisage to use again the service mechanism in the agent society. There will be an application that offers persuasion services (an expert system acting as an attorney office) for specific types of goals. A personal or a teaching agent can borrow such service by negotiating with the application agent of the "attorney".

Issues in Implementing the Agent – Agent Interaction Sphere

In addition to personal agents, our architecture envisages application agents, attached to every application or learning environment, which have an explicit representation of the application's purposes and resources. These agents communicate and negotiate among themselves and with personal agents for achieving their goals. This means that we need an appropriate communication language about goals and resources, which would allow these agents to share information. Currently the agents communicate in a proprietary language and protocol, but we'll probably switch to an extended KQML. We plan to extend it with some additional dialogue constructs reflecting the specifics of communicating about goals. All agents will have to obey a kernel-set of the language, while domain specific teaching and diagnostic agents will be able to communicate in extended versions of the language, specific for the domain.

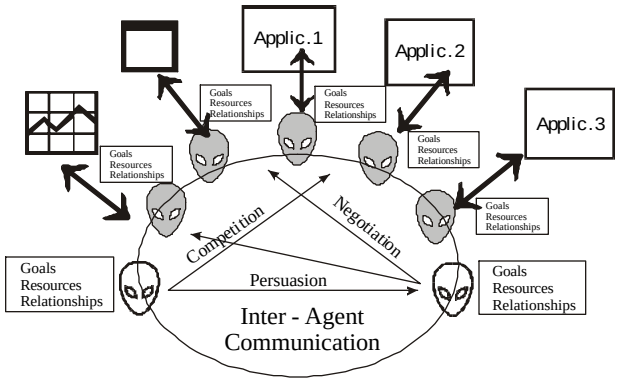

Figure 3: Inter-Agent Interaction Sphere.

In addition to the basic communication, there will be a higher strategic level of communication, on which the decisions about the persuasion / negotiation strategy will take place. The agents have to be able to form teams when this is profitable, to co-operate or to compete or even take adverse steps of blocking another agent if needed. For example, if the personal agent A of a user notices that help is often asked for by another personal agent B, but B never returns the favour, then A may decide to be uncooperative the next time B asks for help. A may even decide to be

vicious and spread a rumour around in the agent society that agent B is uncooperative.

It is obvious that to support such social reasoning, various knowledge bases are necessary, containing knowledge about basic rules of negotiation and social behaviour. However, the agent-based architecture frees us from the need to build this basic social intelligence in every single agent. It allows representing one such knowledge base and reasoning component with another agent from which the normal agents can borrow expertise or advice. This makes the framework flexible, extendible and modifiable.

Current Implementation State

We are currently in an initial stage of implementation of this ambitious architecture. We chose JAVA 1.1.5 as a platform. We have designed and implemented all basic parts of the architecture: an agent environment (including a resource manager), a basic "kernel" agent, from which all the agents are instantiated, and a communication mechanism among the agents.

Agent Implementation

The agents are implemented as threads. Each agent consists of 2 or more threads:

- *a message processing thread*: an important aspect in the design of a multi-agent system concerns the awareness of the agent to messages or signals; therefore it is necessary to guarantee the processing of messages by reserving at least one thread for this task,
- *a kernel thread*: since the agent has to be aware of its state all the time, it is necessary to reserve a thread for monitoring the thread's state, computational resources state and performance,
- *work threads*: these are threads which enable the agent to pursue goals. By enabling the agent to generate work threads we offer the possibility that the agent works on several goals simultaneously. An agent can assign several goals to a single work thread or generate more work threads in order to have a work thread for every goal.

Since the creation of a work thread demands usage of a commonly shared resource processor, every new work thread requires an endorsement from the resource manager. The resource manager can increase or decrease the amount of permitted working threads at any time. By limiting the number of work threads we decrease the share of processing time for every thread, by increasing the number, we offer more processing time for every thread.

The Communication among Agents

Agents communicate by sending messages (objects) to each other. Three different communication forms have been implemented:

- *Peer communication*: the agent uses an address book to generate a channel to the desired agent. The channel takes care to send the message to the target agent. The use of a channel enables migrations of agents during communication, which may be forced by computational resource limitations. That is, agents may reside on any registered machine in the intranet, thus supporting a fully distributed system. A channel offers a bi-directional communication line, which is useful for private communication among agents.
- *Bus communication*: it offers communication among several agents. An agent can create, join, leave, or send a message to a bus. Every message is broadcast to all members. A bus can be itself a member of a bus. It is useful for sending requests among agents, but it may lead to a flood of answers that the requesting agent will have to process.
- *Hub communication*: Unlike the bus communication hub communication differs in that the message is not sent to all members at once. It is sent to the first member of the hub, who is given some time to react. In case that member rejects the request or doesn't reply, the message is sent to the next member of the hub. This type of communication is appropriate when an agent is requesting a service from the agents on its list of relationships, ordered according to the degree of closeness or importance of the relationship.

The Agents Environment

The environment in which all personal and application agents "live" guarantees a safe existence and fair chance of getting computational resources for the agents. It provides resource management, detects and limits or eliminates faulty and malicious agents, i.e. agents that consume too many resources. It also cares for appropriate agent migration for the purpose of optimal use of resources. The agent environment consists of the several self-explainable modules: monitor, resource management, address book, transporter (for agent migration) and decider (in case of resource conflicts).

We are currently working on the user models: the uniform representation of user data in terms of goals, resources and relationships. The next steps will be the creation of a set of simple reasoning rules about goals and resources and its incorporation into an existing constraint-based problem solver, which will provide the reasoning service for the agents. The creation of similar services for negotiation and persuasion will follow.

Acknowledgments.. This research was carried out under the auspices of the Canadian Telelearning Network of Centres of excellence, project 6.2.4.

References

Collins, J., Greer, J., Kumar, V., McCalla, G., Meagher, P., Tkach, R. 1997. Inspectable User Models for Just in Time Workplace Training. In A. Jameson, C. Paris, C. Tasso, (Eds.) User Modeling, Proceedings of the UM97 Conference, Springer Wien New York, 327-337.

Greer, J., McCalla, G., Cooke, J., Collins, J., Kumar, V., Bishop, A., Vassileva, J. 1998. The Intelligent Helpdesk: Supporting Peer-Help in a University Course, In Proceedings ITS'98, San Antonio.

Schank R.C. and Abelson R.P. 1977. Scripts, Plans, Goals, and Understanding, Lawrence Erlbaum.

Vassileva, J. 1996. A task-centered approach for user modeling in a hypermedia office documentation system. User Modeling and User-Adapted Interaction 6 (2-3): 185-223.

Vassileva, J. 1997. Goal-Based Autonomous Social Agents: Supporting Adaptation and Teaching in a Distributed Environment, In Proceedings ABIS'97, Saarbruecken.