DOCUMENT RESUME

ED 466 177	IR 021 248

AUTHOR	Jin, Qun; Ma, Jianhua; Huang, Runhe; Shih, Timothy K.
TITLE	Design Principles of an Open Agent Architecture for Web-Based Learning Community.
PUB DATE	2001-00-00
NOTE	7p.; In: ED-Media 2001 World Conference on Educational Multimedia, Hypermedia & Telecommunications. Proceedings (13th, Tampere, Finland, June 25-30, 2001); see IR 021 194.
AVAILABLE FROM	Association for the Advancement of Computing in Education (AACE), P.O. Box 2966, Charlottesville, VA 22902 ($40, AACE members; $50, nonmembers). Tel: 804-973-3987; Fax: 804-978-7449; Web site: http://www.aace.org.
PUB TYPE	Reports - Research (143) -- Speeches/Meeting Papers (150)
EDRS PRICE	MF01/PC01 Plus Postage.
DESCRIPTORS	Computer Simulation; Computer System Design; Distance Education; Educational Development; Foreign Countries; Higher Education; Instructional Materials; *Intelligent Tutoring Systems; Man Machine Systems; World Wide Web
IDENTIFIERS	Japan; *Learning Communities; *Virtual Communities

ABSTRACT

A Web-based learning community involves much more than putting learning materials into a Web site. It can be seen as a complex virtual organization involved with people, facilities, and cyber-environment. Tremendous work and manpower for maintaining, upgrading, and managing facilities and the cyber-environment are required. There is presented an inevitable choice to use various agents to assist or replace people to a certain extent for the work in a virtual learning community. There is a strong need for the development of the designated architecture for integrating and using various agents. This research focuses on developing an open agent architecture that can easily integrate developed agents into a learning system and flexibly modify the agents with permission. This paper describes the design ideas of such desired architecture, demonstrates how the architecture system works, and explains three agents--the scheduling agent, the friend-making agent, and the Web-course management agent--in a Web-based learning system called University21. (Contains 10 references.) (MES)

Design Principles of an Open Agent Architecture for a Web-based Learning Community

Qun Jin
School of Computer Science, The University of Aizu
Aizu-wakamatsu, Fukushima 965-8580 Japan
jinqun@u-aizu.ac.jp

Jianhua Ma and Runhe Huang
Faculty of Computer and Information Science, Hosei University
3-7-2, Kajino-cho, Koganei-shi, Tokyo 184-8584, Japan
{jianhua, rhuang}@k.hosei.ac.jp

Timothy K. Shih
Dept. of Computer Science and Information Engineering
Tamkang University, Tamsui, Taiwan 251, R.O.C.
tshih@cs.tku.edu.tw

Abstract: A Web-based learning community is far beyond putting learning materials into a Web site. It can be seen as a complex virtual organization with enormous work involved with people, cyber-environment, and facilities. There is an inevitable choice to use various agents to assist or replace human to a certain extent for the work in a virtual learning community. There is a strong need for the development of the designated architecture for integrating and using various agents. This research is focused on developing an open agent architecture that can easily integrate developed agents to a learning system and flexibly modify the agents with permission. This paper describes the design ideas of such desired architecture, demonstrates how the architecture system works, and explains three agents in a Web-based learning system, University21.

Introduction

With advances in the Internet and multimedia technologies, development of Web-based learning communities for K12, high education, lifelong learning of citizen has become a hot topic. However, establishing a Web-based learning community is far beyond putting learning materials into a Web site (Donath, 1997, Jones, 1997, Maynatt, 1997). A learning community is in fact a complex virtual organization (Ma, Huang, and Shin, 1999, Ma and Huang, 1999, Ma, Huang, and Kunii, 2000) involved with people, facilities, and cyber-environment. Tremendous work and manpower for maintaining, upgrading, and managing facilities and cyber-environment are required. Teaching/learning activities also require lots of assistance to release human from difficult and/or tedious work so as to achieve teaching/learning efficiency. With rapid developments of software agent technologies, it becomes a good choice to use various software agents (Johnson, 1998, Mengelle, etal, 1998) to assist or replace human for work in a virtual learning community. To efficiently conduct teaching, learning and management activities, many agents are required. In a virtual learning community, agents are mainly classified into three categories, personal agents, facility agents, and cyber-environment agents. Personal agents assist people for personal activities. Facility agents automatically provide monitoring, maintenance and upgrading of tools, systems, and resources. Cyber-environment agents are resided in virtual office, virtual private room, virtual courseroom, etc. for supporting teaching/learning related activities. Tab. 1 gives a list of only partial agents on demand in a virtual learning community system. Facing so many agents needed, it is natural to encounter a problem, that is, how to integrate agents to the virtual learning community system. It seems an architecture that allows easily and flexibly integrate developed agents is preferred. This motivates us to design and develop a so-called open agent architecture. Agents developed by different parties can be easily and flexibly integrated to a virtual learning community system via the open agent architecture. This paper is focused on describing the design principles of such desired architecture, demonstrates how the architecture system works, and describes some agents in University21 (Ma, Huang, and Kunii, 2000). Finally, conclusions are drawn and future work is addressed.

Personal Agents	Facility Agents	Cyber-environment Agents
- Personal scheduling agent - Friend-making agent - Mail/news filtering agent - Personality agent - Announcement agent - Meeting scheduling agent - Group forming agent	- System upgrading agent - Tools upgrading agent - Resource monitoring agent - Resource management agent - System maintenance agent - Tools maintenance agent - Security agent	- User administration agent - Curriculum administration agent - Web course maintenance agent - People awareness agent - Teaching/learning assistance agent * Authoring agent * Tutoring/pedagogical agent * Student learning monitoring agent * Assessment agent ......

Table 1: A list of partial agents on demand in a virtual learning system

The Design Principles

The open agent architecture is designed under the following emphases. (1) It should make system developers easy to integrate agents developed by other parties; (2) It should make agent developers flexible to upgrade or modify agents; (3) It should make agents transparent to users for use. It is necessary to point out that to be platform independent, the agent programs are written in Java language. The open agent architecture has the following three characteristics and can work in two modes: the centralized or the distributed

Ease

It means that a virtual leaning system can easily integrate an agent. In a centralized system, the system only needs to download the agent programs and unpack the agent programs, adds the agent name to the agent list, adds the agent IP address to the agent routing table, allocates a certain resource for the agent, and starts the agent server when the agent is requested by a user. In a distributed system, the system only needs a privilege to start/stop the agent server (Schmidt, 1998) and redirect users to a site where the agent programs reside and the agent server runs since agent programs are distributed in the local machine and resources are managed locally too.

Flexibility

It means that an agent can be flexibly added, removed, and modified with permission. In a centralized system, an agent can be flexibly registered or deregistered. Agent programs are uploaded or upgraded by the agent developer with permission. In a distributed system, an agent is only registered or unregistered with the agent name and address where the agent programs are resided. Uploading programs is not necessary. The agent developer can modify or upgrade the agent programs in his/her local site. The open architecture system is only given a privilege to start/stop the agent server in the remote site. When an agent is to be deregistered or upgraded, the system stops the agent server, deletes the agent name from the agent list, and removes the agent IP address from the agent routing table.

Transparency

It means that all registered agents are dynamically shown in the agent list with attached information like agent name, agent ID, server status, and program type. Whether an agent is available or not, whether an agent server is running or not, and whether an agent is being used or not, all those information is available to users. The server starts to run when a user makes a request of using the agent if the server is not running. The client program is automatically downloaded to the user site. If it is a Java applet, it automatically runs on the Web browser in the user side. Otherwise, the user runs Java application program with a simple command like *make run*.

The open agent architecture system is designed under the above emphases with features of ease, flexibility, and transparency. Fig. 1 gives a system architecture that consists of four basic layers: client, agent, platform and external layers.

- In the client layer, TCP protocol is used for client-to-client communications via the agent server. The agent server is running and then waiting for connections from its clients. After the connections are established, communications among the clients are conducted through the TCP protocol.

- In the agent layer, socket connection is used in a centralized system while IIOP (Internet Inter-ORB Protocol) (Schmidt, 1998) which runs atop the TCP transport protocol is used in a distributed system for communications among agents via the messenger.
- In the platform layer, the function modules of the open agent architecture are provided. This layer is further divided into two parts: system components and external system connection components. The system components consist of agent manager, agent data holder, and messenger. The external system connection components provide various proxy servers corresponding protocols that are ready for connection to external devices or systems. The request of using an external device or system is analyzed by the connection adaptor and the inquiry agent is connected to the corresponding proxy server.
- In the external layer, it places devices and/or systems that agents may require to connect, such as HTTP Server, Mail Server, FTP server, the Internet, database, search engine, etc.

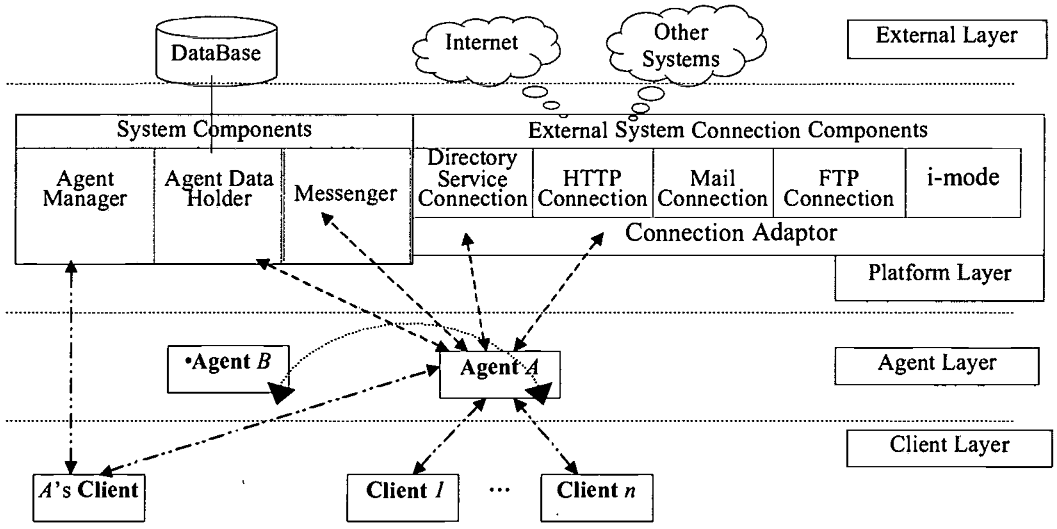

Figure 1: A basic composition of the open agent architecture

The Function Modules

How an agent is functioning in the client layer is relied on the agent (agent server and client programs) rather than the open architecture system. The most core functions of the open architecture system are provided in the platform layer. In this section, we will discuss each function module below.

Agent Manager

Agent management is a function module of operating an agent life cycle on the open agent architecture platform. It mainly provides functions like agent registration, upgrading, deregistration, running an agent server, connection of agent clients to the agent server, and front-end functions to the agent data holder.

To register a developed agent to the open agent architecture platform, an agent developer accesses a web page and clicks *register* button, and then an agent registration page is popped up. The developer inputs necessary agent related information, such as agent name, agent IP address, agent program package name, agent working mode, and server execution command. For working in the centralized mode, the agent program package (*.jar file) is uploaded, unpacked and placed to the host where the agent manager is resided. At the same time, agent related information is sent to the agent data holder by calling method *addAgent()*. For example, an agent name is added to the agent list and agent IP address is added to the agent routing table in the agent data holder. A space allocation request is also sent to the database via the agent data holder. The process of agent registration is shown in Fig. 2.

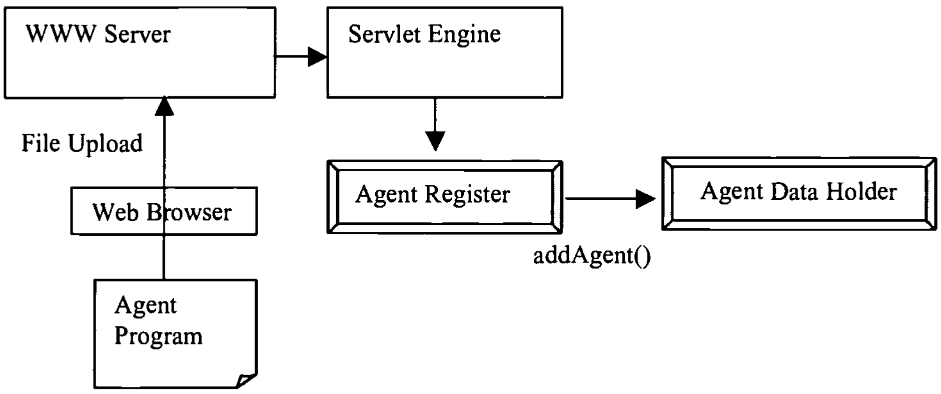

Figure 2: The process of agent registration

For working in the distributed mode, uploading of the agent programs is not necessary since all agent programs are distributed and run on the agent developer's local JVM machine. Only information about the agent, such as agent name, agent IP address, server execution command, etc., is registered. After an agent is registered, it is ready for use. When a user would like to use an agent, the user goes to a specified web page and clicks an agent icon (or an agent name from a list of agents), after that, the agent client program is automatically downloaded to the user's local machine and run on the local JVM (Java virtual machine). The client program requests a connection to the agent server and the agent server receives the request and accepts the connection. The connection between the server and the client is thus established.

Similarly, to deregister or modify an agent from the open agent architecture platform, an agent developer accesses a web page and clicks *deregister* or *modify* button. The developer has to input his/her user ID and password. If the permission is granted, he/she can enter information related to the agent to be deregistered or modified, the agent manager will perform the process of removing the agent, such as, disconnecting all clients, removing agent programs, deleting the agent related information from the agent list and the routing table, and releasing allocated space by calling *deleteAgent()* or *modifyAgent()* from the agent manager.

Agent Data Holder

The agent data holder mainly manages information and data of agents like agent name, agent description, agent type, agent's access path, and etc. The following methods are public to the external modules.

- *Void addAgent()*
- *Void deleteAgent()*
- *Void modifyAgent()*
- *String getAgentName()*
- *String getAgentPath()*
- *Void getAgentInfo()*

Calling those methods is either from the agent register program in the agent manager or from the messenger when an agent requests to communicate with another agent. There is an interface between the agent data holder and the database. Data related to an agent is stored in the database. The agent list table and the agent routing table as shown in Fig. 3 are updated during the agent life cycle operations and client connections.

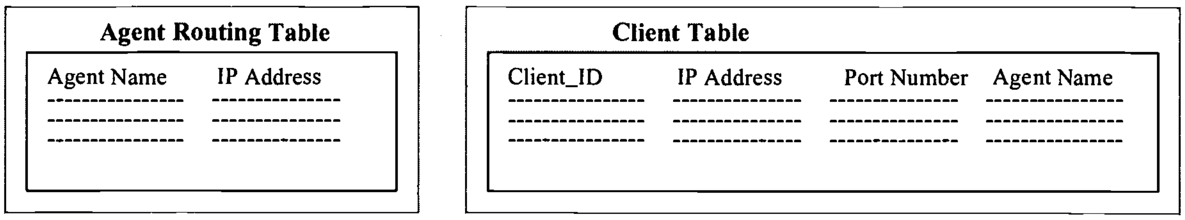

Figure 3: The agent routing table and the client table

Messenger

Messenger is a mediator among agents. It enables the communications between two agents. When receiving a request of connecting to another agent from an agent, the messenger analyzes the request with attached information and retrieves the requested agent information from the agent routing table via the agent data holder by calling *getAgent()* method as shown in Fig. 4. When receiving the retrieving result from the agent data holder, the messenger connects the inquiry agent to the requested agent by calling *connectToAgent()* method if the request agent exists, otherwise it returns a "*not found*" message to the inquiring agent if the requested agent does not exist.

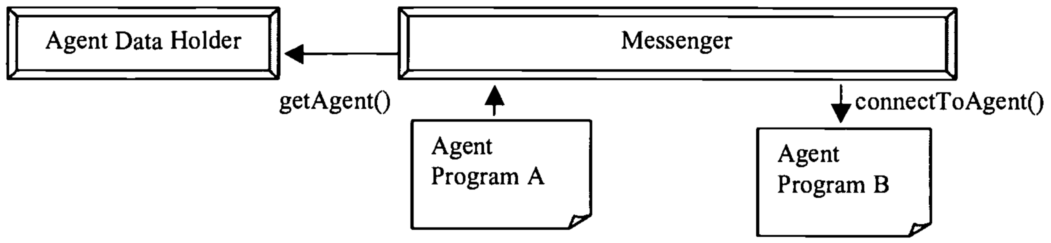

Figure 4: The role of the messenger

Connection Adaptor

In fact, the connection adaptor behaves like a protocol handler that dispatches the request from an agent and decides which type of communication protocol to be used between the inquiry agent and the requested external device or system. After that, the connection adaptor connects the agent to the requested device or system via a proxy server corresponding to the communication protocol to be used.

Agents in University21

University21 (Ma, Huang, and Kunii, 2000) is a virtual university system that is aimed at supporting all teaching, learning and administrating activities via the Internet. It requires many agents to assist teachers, students, and administration staff for the teaching, learning and administration work involved in University21. After developing many agents written in Java by different people or parties, we face a problem of integrating developed agents into University21 system. The open agent architecture as the agent system platform in University21 solves our problem. Currently, three most often used agents, scheduling agent (Ito and Shintani, 1997), friend-making agent, and web-course maintenance agent, are integrated into University21, respectively. The scheduling agent is used for personal scheduling, meeting scheduling, team work scheduling, and tele-lecturing scheduling. A user can request to use the agent from his/her private office, a virtual courseroom, or a virtual collaboration room. Friend making agent plays an important role in a virtual university system since users feel lonely when they are teaching/studying in such virtual environment and the friend making agent can help them find friends to chat, discuss, and even collaborate with. To achieve effective and efficient teaching/learning, friends are indispensable. A user can use the friend making agent from his/her private office. Web-course maintenance agent helps teachers to periodically check and update their teaching materials. With more and more Web-based teaching materials, checking and updating them become difficult and tedious. However, it is an important work. The web-course maintenance agent relieves teachers from such difficult and tedious work. A teacher can use the agent from a virtual couseroom. Both scheduling agent and friend making agent are multi-agent systems and web-course maintenance agent is a single agent system. For the multi-agent systems, communications among agents are conducted via the messenger on the platform layer of the open agent architecture. It is necessary to point out that University21 is currently a centralized system. Integrating three agents to University21 follows a centralized fashion. The programs of three agents are uploaded, unpacked, and resided in University21's server machine as we discussed above. The agent server programs are run in University21 host and the client program is automatically downloaded to and runs on the user's local Java virtual machine when a user requests to use the associated agent by clicking a hyperlink in a web page. The integrated agents are transparent to all users in University21. Users are not aware where agent programs are. What they do just interact with GUI of each agent. They can input their preference, make requests, be consulted with, and get reporting results as well as communicate with an agent through the GUI of the agent.

Conclusions and Future Work

To be platform independent, the open agent architecture is implemented in Java language. For those agents developed in Java language are expected to be easily and flexibly integrated into a virtual learning community system via the open agent architecture platform. As a virtual learning system, University21 has the open agent architecture embedded. Three agents are integrated into University21 via the open agent architecture platform. It proves the our design efforts of the open agent architecture: making the system easy to integrate developed agents, making agent developers flexible to modify their agents, and making agent related information transparent to users. The agent architecture is just like a door of a virtual learning community system that is open to developed agents and allows them to easily and flexibly join and work for human in the teaching/learning community.

It is necessary to emphasize that the open agent architecture is applicable not only in Web-based learning community system but also any Web-based virtual systems such as virtual trading/shopping center system, home agent system, a virtual company, and a virtual city where various agents are used. However, this research is just at its infant stage, there is lot of remain work to be done. We will mainly focus on implementing a distributed open agent architecture, handling agent synchronization problem, and making agent manager intelligent.

Acknowledgments

The author thanks Mr. Masao Nonaka and Ms. Sanae Wada for their partial implementation work.

References

Donath, J.S. (1997). *Inhabiting the Virtual City: the Design of Social Environment for Electronic Communities*. PhD Thesis, MIT media Arts and Sciences. http://judith.www.media.mit.edu/Thesis/

Ito, T. & Shintani T. (1997). *Persuasion among Agents: An Approach to Implementing a Group Decision Support System Basedon Multi-Agent Negotiation* (pp592-597). In the Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97). Nagoya, Japan.

Johnson, L. (1998). *Pedagogical Agents* (pp13-22). In the Proceedings of ICCE'98, Vol.1.

Jones, Q. (1997). *Virtual-Communities, Virtual Settlements and Cyber-Archaeology: A Theoretic Outline*, JCMC (3). http://jcmc.huji.ac.il/vol3/issue3/jones.html

Ma, J., Huang, R., & Shih, T. K. (1999). *The Concept, Framework and Architecture of an Integrated Educational System for Virtual Universities* (pp619-626). In the Edited Book: Advanced Research in Computers and Communications in Educations, Vol.2, IOS Press, ISBN 1-58603-027-2.

Ma, J., & Huang, R. (1999). *Towards an Integrated Educational System for Global Teaching and Learning* (pp256-263). In the Proceedings of the International Conference on Distributed Multimedia System (DMS'99), Aizuwakamatsu, Japan.

Ma, J., Huang, R. & Kunii, T. L. (2000). *University21: An Integrated Educational System* (pp109-139). Chapter 7 in the Book: International Perspective on Tele-education and Tele-learning, Edited by G. Orange and D. Hobbs, Ashgate publishing Limited, ISBN 0-7546-1202-3.

Maynatt, E. D., Alder, A. Ito, M., & O'Day, V.L. (1997). *Design For Network Communities*. CHI97 Electronic Publications. http://www.acm.org/sigchi/chi97/proceesings/paper/edm.htm

Mengelle, T., etal., (1998). *Teaching and Learning with Intelligent Agents: Actors, Intelligent Tutoring Systems*. Lecture Notes in Computer Science 1452.

Schmidt, D. C. (1998). *Evaluating Architectures for Multithreaded Object Request Brokers* (pp54-60). Journal of Communications of the ACM, No. 10, Vol. 41.