Design and Implementation of Virtual Laboratories Using Wrapping and Agent Techniques

Y. H. Li, C. R. Dow, F. W. Hsu, [+]C. M. Lin, and [‡]T. C. Huang
Department of Information Engineering and Computer Science
Feng Chia University, Taichung, Taiwan 407
[+]Department of Computer and Communication Engineering, Nan Kai College, Nan Tou, Taiwan 542
[‡]Department of Electrical Engineering, National Sun Yat-Sen University, Taiwan 804
{ sbmk, crdow, sam }@pluto.iecs.fcu.edu.tw
[+]lcm@nkc.edu.tw, [‡]tch@mail.nsysu.edu.tw

Abstract

This work designs and implements a virtual laboratory that breaks the limits of time and location in traditional education under the innovation of network and information technology. Since the development of a virtual laboratory from scratch is time consuming and costly, our system can be used to wrap an existing CAI tool without knowing its source code. The key concept is to use mobile agents as middleware for wrapping. The framework consists of three parts: mobile agent execution environment, mobile agent, and learning platform. Experimental results demonstrate that the performance of our system outperforms a client- server-based virtual laboratory in terms of dispatching time.

Keywords: Mobile agent, virtual laboratory, learning technology, wrapper, and design pattern.

1 Introduction

Virtual laboratories, virtual course-rooms, virtual collaboration rooms, virtual libraries, and virtual private offices are categorized into the distance learning framework [7]. In general, a traditional laboratory is composed of a number of physical instruments and a set of software applications. To construct a laboratory with these physical instruments is usually very expensive. Therefore, virtual instruments or simulation tools can be used to construct a virtual laboratory. Currently, various simulation tools such as CAI/CAD tools are available in the market, but most of them are stand-alone tools. To include these stand-alone applications in a virtual laboratory and use them via networks, the source code of these systems have to be modified to enable the network capability. Thus, it is a challenge to reuse the stand-alone application for constructing a virtual laboratory without even knowing its source code.

Mobile agent [12] is an emerging technology and has the potential for being a convenient structuring technique in distributed and Internet applications. The mobile agent technique has been used in the design and implementation of a virtual laboratory of digital circuit [5]. However, the building process of the virtual laboratory was from scratch and most functions of the system are implemented in an on-demand approach. Thus, it is necessary and interesting to provide a framework to help designers in developing mobile agent-based virtual laboratory applications.

This work proposes a framework to solve above problems and provides various design patterns as well as features for virtual laboratory developers and end-users. In our framework, we use wrappers as middleware and design various agents to provide interaction and collaboration between instructor and learner or learner and learner in a virtual laboratory. Various existing CAI tools or on-line web sites can be incorporated into a virtual laboratory. The framework that reuses the existing CAI tools and uses design patterns to develop virtual laboratories will save both time and resources.

The rest of the paper is structured as follows. Section 2 briefly describes the background materials and related work. In Section 3, the wrapping concepts and models are described. The virtual laboratory agents and design patterns are presented in Sections 4 and 5. Implementation of virtual laboratories and experimental results are presented in Sections 6 and 7. Finally, the conclusions are described in Section 8.

2 Related Work

There are many research areas related to our work, such as e-learning, virtual laboratory, mobile agent, wrapper, and software patterns. Based on the equipment and user access in each experiment, laboratories can be classified into four types [5], including traditional laboratory, remote

0-7803-8819-4/04/$20.00 ©2004 IEEE.

lab [6], micro lab, and macro lab [17]. This section introduces some related work of these topics.

The virtual laboratory proposed by Muzak *et al.* [11] is a macro lab, which focuses on the digital computers and is based on the client-server computing paradigm. The virtual laboratory changes the traditional lecturing method into a network enabled lecturing environment. Moreover, it focuses on the lecturing only and lacks of some important activities such as an assessment system. Another virtual laboratory for teaching electric machinery proposed by Tzeng and Tien [17] has the realistic, interactive, and flexible characteristics and is based on client-server computing paradigm. When a virtual laboratory becomes larger, traffic jam will be caused in the environment. Thus, the problems of adaptability and scalability of virtual laboratories need to be solved.

The mobile agent techniques [8, 10, 12, 14-16, 18] have been widely used in many Internet applications. Mobile agents, characterized by the unique ability to transport their state and code with them, migrate among the components of a network infrastructure where they resume execution. The merits [14] of mobile agents include the reduction of the network traffic and latency in client/server network computing paradigm, autonomy, dynamic adaptation, protocol encapsulation, heterogeneity, robust performance, and fault tolerance. The mobile agent has the unique ability to transport from one system in a network to another. When large volumes of multimedia data are stored on remote hosts, these data should be processed in the locality of the data rather than transferred over the network.

One important problem we face when building a virtual laboratory is where to place an extra function for a stand-alone learning tool without knowing its source code. The wrapper [5, 15] is a technique that provides a convenient way to expand upon existing functions of an application program without modifying its source code. Wrappers provide a way to compose applications from different parts. The fact that a mobile agent is wrapped should be transparent to other mobile agents in the system, and potentially to the agent itself. Therefore, the union of a mobile agent and a wrapper looks just like another stationary agent.

Software patterns [9] have their roots in Christopher Alexander's work in the field of building architecture. Patterns facilitate reuse of well-established solutions when known problems are encountered. The patterns that are adapted in the stage of object-oriented design (OOD) are called design patterns. Design patterns [13] are reoccurring patterns of programming code or components software architecture and which have succeeded as solutions to past design problems. The purpose is to increase the reusability and quality of code and at the same time reduces the effort of software development. They are useful in achieving flexible and extensible design and make future changes and modifications easier. There are different pattern formats [3], the minimal format contains the following headings or ones dealing with similar subject matters, including Name, Problem, Context, Forces, and Solution.

3 Wrapping Concepts and Models

To integrate existing learning tools for constructing a virtual laboratory, wrapping is the key component. From the user's perspective, the wrappers are hidden. As shown in Fig. 1, a wrapped CAI tool looks like any other software applications. However, from the system perspective, the wrapper is an agent itself. As shown in Fig. 2, a wrapper agent consists of two function parts, including I/O interception and Application Programming Interface (API).

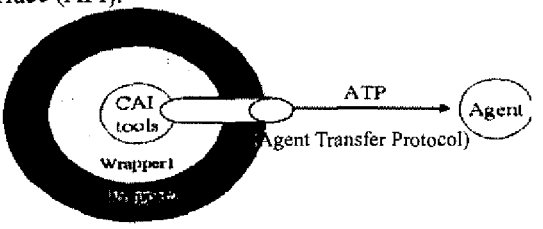

Fig. 1 Wrapping Concept for the User Perspective

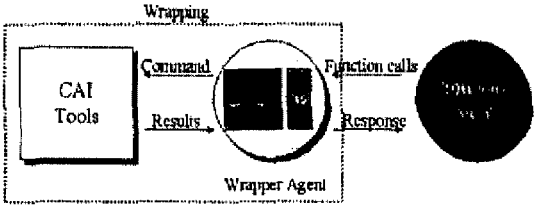

Fig. 2 Wrapping Concept for the System Perspective

I/O interception is in charge of exposing the functions of a CAI tool as a set of methods of an object by intercepting its I/O and commands. Each CAI tool has a fixed operation instruction set, such as hot keys, run-time parameters, etc. This component gives exact operation instructions to the CAI tool and triggers this tool to execute the corresponding functions. API is the interface for controlling the virtual laboratory.

Because of the wrapping concept, both adaptability and cost-effectiveness are provided in our virtual laboratory framework. From the system perspective, the framework supports object-oriented programming (OOP) in the development of a virtual laboratory and various standalone CAI tools can be easily reused in virtual laboratory by wrapping. Thus, the cost of time and system and educational resources can be reduced effectively.

The software model, as shown in Fig. 3, consists of four layers. The top layer is the application software, such as CAI tools. The second layer consists of services and user-defined mobile agents which interact with java native interface, Windows API, and agent API. The third layer is agent runtime layer which consists of two parts: a core

framework and a set of extensible system management components. The primary functions of the core framework are initialization and serialization/deserialization of agents, class loading and transfer, agent references, and garbage collection. Moreover, it provides core services for executing agent including creation, clone, dispatch, retraction, deactivation, and activation. The management components are cache manager, security manager, and persistence manager. At the bottom are the communication API and its implementation.

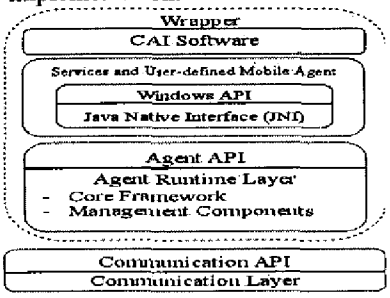

Fig. 3 Software Model

4 Virtual Laboratory Agents

In general, the virtual laboratory world is composed of various objects, including roles, hardware/software, network service, and activities. The roles include instructors and learners. In the hardware/software aspect, the hardware is the physical experimental equipment, and the software includes teaching function, learning function, and other functions. Because the virtual laboratory evolves from distance learning, it should provide the network capabilities and services for different networking paradigm. Thus, the instructors and learners can communicate with each other through the Internet.

Various teaching/learning activities should be contained in a virtual laboratory. The relationships among activities, software agent, and participant are shown in Fig. 4. Learners can also use the demo agent to demonstrate the learning procession to the instructor. An instructor can guide learners in different levels by using a guide agent. The tutoring agent dispatched by the instructor will execute a learning tool for learners to cognize the learning materials. Instructor and learners can use a criticizing agent to discuss subjects with each other. The assessment agent can provide different levels of assessment materials, and it can also provide homework for learners to practice. The QA agent has some predefined FAQ rules and it will reply with the appropriate answers when learners ask questions. The monitoring agent can act as the instructor to monitor the learner's actions and learning status.

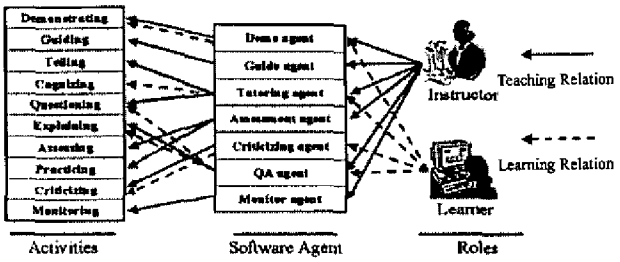

Fig. 4 Relationships in a Virtual Laboratory

In our proposed virtual laboratories, the mobile agent techniques are used to construct the virtual laboratories. There are three components in our virtual laboratory framework, including the mobile agent execution environments (MAEE), agents, and learning platforms. The details of these system components are described in our previous work [4-5].

5 Design Patterns

Our virtual laboratory components can be classified into seven design patterns, including system environment, nomadic management, delegation, behavior management, agent communication, agent coordination, and auxiliary. This classification scheme makes it easier to understand the virtual laboratory framework.

(1) System Environment Patterns: The system environment patterns include instructor and learner patterns that provide a mobile agent execution environment. The class diagram of system environment patterns is shown in Fig. 5. The execution environment has two types: instructor and learner environments, which inherit the environment pattern and provide two abstractions, initializeJob and routineJob.

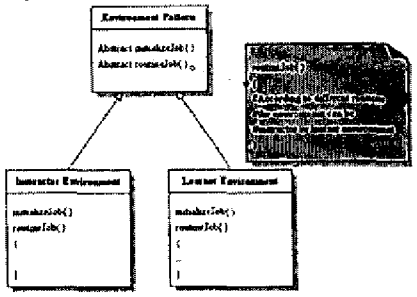

Fig. 5 System Environment Patterns

(2) Nomadic Management Patterns: The nomadic management patterns can be divided into two types, including one-way and round-trip types. The nomadic management pattern inherits the Aglets patterns and provides three nomadic schemes including sequential, parallel, and other schemes, which decide mobile agents' itineraries and their navigation among multiple destinations. These patterns maintain a list of destinations and routing schemes.

(3) Delegation Patterns: The delegating patterns contain sequential delegating and parallel delegating patterns, which define schemes for mobile agents to complete their tasks in sequential or parallel strategies. Moreover, the instructor can design a work or experiment that has several procedures to deliver to the learners, and let learners to complete this work or experiment in a sequential order according to the work's or experiment's content. In other words, these patterns provide collaboration models between instructors and learners or learners and learners.
(4) Behavior Management Patterns: The behavior patterns include cloning, disposing, deactivating, and activating operations for mobile agents. We use these behavior management patterns when the software agents accomplish their tasks.
(5) Agent Communication Patterns: The agent communication patterns include broadcast, multicast, and unicast patterns, which define three ways for a mobile agent to carry a message to other mobile agents. Software agents can establish remote communications by using agent communication patterns, which objectify messages in the form of agents that carry and deliver messages between software agents.
(6) Agent Coordination Patterns: The agent coordination patterns contain direct, blackboard, meeting-oriented, and Linda-like pattern, which define coordination schemes whereby a master agent can coordinate or exchange information with slave agents. The coordination patterns are applicable in the following situations: (a) when software agents need to interact with other software agents with reliable and high-bandwidth network connections; (b) when software agents need to interact with other software agents with unreliable and low-bandwidth network connections; and (c) when software agents need to interact with other software agents asynchronously.
(7) Auxiliary Patterns: The auxiliary patterns include GUI, visual appearance, agent management, and resource management patterns, which define some auxiliary functions for mobile agents. The auxiliary patterns are applicable in the following situations: (a) when software agents need GUI between users and software agents; (b) when software agents need some visual objects to represent themselves; (c) when users need to manage software agents' information; and (d) when software agents need to have interactions between CAI tools and software agents.

6 Implementation of Virtual Laboratories

Three different virtual laboratories have been designed and implemented based on our framework. We have implemented our virtual laboratories to several applications, including digital circuit, language learning, and digital signal processing. In the development of a virtual laboratory, the environments and mobile agents are constructed based on the design patterns which described in previous section. These three different virtual laboratories, including Virtual Digital System Laboratory (VDSL), Virtual Language Learning Laboratory (V3L), and Virtual Digital Signal Processing Laboratory (VDSPL) are described below.
(1) VDSL: The virtual digital system laboratory with network capability is created from an existing standalone virtual digital system laboratory, which includes a digital circuit board and a digital analyzer. By using mobile techniques, we enable the network capability of the standalone VDSL. In VDSL, the software contains wires, instruments, layout tools, and 44 kinds of ICs, and learners can practice the design, simulation, debugging, and layout. Instructors can dispatch various software agents providing by virtual laboratory to monitor, broadcast, demo, and cooperate with learners through network capability.
(2) V3L: The virtual language learning laboratory is implemented by integrating the mobile agent techniques with language learning tool. By using mobile agent techniques, we enable the network capability of standalone language learning tool. The learning system's main function includes pronunciation practice of noun and sentence and conversation practice. In V3L, learners are practiced pronouncing by speech recognition system. Because of network capability, various mobile agents with different tasks, and speech recognition system components, V3L improves interaction between instructors and learners in traditional class and distance learning environments.

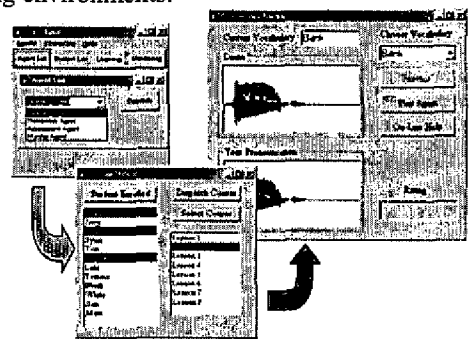

Fig. 6 V3L

(3) VDSPL: The virtual digital signal processing laboratory is created by using DMATEKs' TMS30C542 Evaluation Module (EVM) based on the Texas Instruments' (TI) Code Composer Studio (CCStudio) development software. Further we create the network capability for the virtual laboratory by integrating mobile agent techniques without modifying existing system's source code. Our system is an educational environment in learning digital signal processing (DSP). Learners may use this virtual laboratory to learn DSP by a real-time multimedia demonstration and discuss with instructors or learners through various agents functions. Furthermore, an

instructor can also monitor and evaluate the results, and send it back to a specified learner.

7 Experimental Results

In this section, we describe the conducted experiments and analyze the performances of mobile agent-based and client-server. First, we describe the experimental environment and design. Then, we present the experimental results and analysis.

7.1 Experimental Design and Environment

In our experiments, we used a dedicated cluster of seventeen PCs connected through 100Mb/sec switched Ethernet. These network computing paradigms are a mobile agent paradigm and a client-server paradigm. We consider that the performance is measured by the total time required to complete the requests. In mobile agent-based scenario, the experiments are designed to dispatch mobile agent from the instructor node to learner nodes, and the mobile agent's mission is to carry a demo script file to learner nodes from an instructor node via a network through a closed itinerary. In the client-server scenario, each request comes with a destination server. The learner nodes specify an instructor node and submit its request to the instructor node. We evaluated the effect of different network computing paradigms of our system. In our experiments, we compare the mobile agent paradigm with the traditional client-server paradigm. The number of agencies is from 4 to 16. The size of data that was carried by the mobile agent has been set to none, 10Kb, 100Kb, 500Kb and 1Mb.

In our mobile agent dispatching model, the master agent dispatches a cluster of mobile agents one by one. It is in fact a serial dispatching model decomposed the agent dispatching into several phases, including initialization, agent cloning, and agent transferring. Thus, there are three time factors for mobile agent-based network paradigm, and we define these time factors as follows:

- T_i: Initialization time of a mobile agent system.
- T_c: Cloning time of a mobile agent.
- Tt: Transmission time of a mobile agent from source to destination.

The total time of mobile agent migrating is $T = T_i + T_c + T_t$ if the length of dispatch path is one. If the length of dispatch path is not one, the total time is $T = (T_i + T_c + T_t) * L$, where L denotes the length of dispatch path.

In the client-sever model, the client and server programs are treated as fixed agents. The following are two time factors used in the client-server model.

- T_a: The thread time of the server socket accepts a client socket.
- T_m: The message communication time.

The total time cost of client-server transferring is $T = T_a + T_m$.

7.2 Performance Results

In the experiments, we evaluated the dispatching time vs. data size under different network computing paradigms. There are two figures for variant number of nodes, as shown in Figs. 6 and 7, with the increase of the number and size of mobile agents, the differences are rising. However, when the data size is smaller than 500k, the performance differences are smaller. When the data size is bigger than 500k, the performance difference becomes larger.

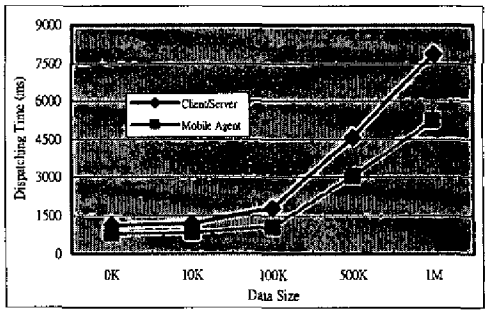

Fig. 7 Results for Dispatching 4 Nodes

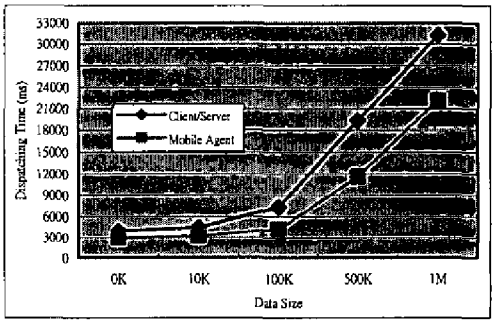

Fig. 8 Results for Dispatching 16 Nodes

Furthermore, with the increasing of number of nodes and data size, the performance of the mobile agent network computing paradigm is better than that of the client-server network computing paradigm. In the client-server model, there are typically several flows between the client and server. In the mobile agent model, these flows can be reduced to a single mobile agent. Based on the results, the mobile agent network paradigm is a better choice.

In distance learning, educational resources may be requested at the same time and instructors may need to dispatch various agents to do some tasks simultaneously. Thus, a parallel processing environment may be needed and the number of clients may be larger. With the improvements of network technology, the response time for an individual agent can become shorter and the network traffic can be reduced. In comparison, the dispatch process may cause the bottleneck and the

performance of the dispatching model becomes a critical factor. In our framework, the mobile agent paradigm provides a better solution than the client-server network computing paradigm for distance learning.

8 Conclusions

In this paper, we design and implement virtual laboratories which have the merits of adaptability and cost-effectiveness. Our virtual laboratory integrates the mobile agent technique and wrapper concept to perform software re-engineering. Further, various design patterns are formulated for the virtual laboratory development. The framework can be used to assist the designers in developing virtual laboratories in a formal approach and the existing CAI tools can be reused without the source code of the original programs. In the meantime, various software agents have been designed and our virtual laboratory has been targeted on various applications, including digital circuit, language learning, and digital signal processing. This work focuses on the development of virtual laboratories in the Macro University. Plans for the future are to integrate our framework with other components of the Macro University and to implement more related design patterns. Thus, future work is adding the knowledge management in these software agents to increase their intelligence. Then the teaching/learning environment based on our framework will get more intelligence through various software agents with knowledge management functions.

Acknowledgement

The authors would like to thank the National Science Council of the Republic of China for financially supporting this research under Contract No. NSC 92-2213-E-035-016.

References

[1] S. K. Chang, T. Arndt, S. Levialdi, A. C. Liu, J. Ma, T. Shih, and G. Tortora, "Macro University A Framework for a Federation of Virtual Universities," International Journal of Computer Processing of Oriental Languages, Vol. 13, No. 3, pp. 205-221, 2000.

[2] A. I. Concepcion, J. Ruan, and R. R. Samson, "SPIDER: A Multi-agent Architecture for Internet Distributed Computing System," Proceedings of the ISCA 15th International Conference on Parallel and Distributed Computing Systems, pp. 147-152, September 2002.

[3] D. Deugo and M.Weiss, "A Case for Mobile Agent Patterns," Proceedings of the Mobile Agents in the Context of Competition and Cooperation (MAC3) Workshop Notes, pp. 19-22, May 1999.

[4] C. R. Dow, F. W. Hsu, T. K. Yang and J. Y. Bai, "A Virtual Laboratory for Digital Signal Processing," Proceedings of International Conference on Information Technology: Research and Education (ITRE '2003), pp. 166-170, August 2003.

[5] C. R. Dow, C. Y. Lin, C. C. Shen, J. H. Lin, and S. C. Chen, "A Virtual Laboratory for Macro Universities Using Mobile Agent Techniques," The International Journal of Computer Processing of Oriental Languages, Vol. 15, No. 1, pp. 1-18, 2002.

[6] F. J. Gomez, M. Cervera, and J. Martinez, "A World Wide Web Based Architecture for the Implementation of a Virtual Laboratory," Proceedings of the 26th Euromicro Conference, Vol. 2, pp. 56-61, September 2000.

[7] G. Jiang, J. Lan, and X. Zhuang, "Distance Learning Technologies and an Interactive Multimedia Educational System," Proceedings of the IEEE International Conference on Advanced Learning Technologies, pp. 405-408, August 2001.

[8] D. Johansen and K. Lauvset, "An Extensible Software Architecture for Mobile Components," Proceedings of the 9th Annual IEEE International Conference and Workshop on the Engineering of Computer-based Systems, pp. 231-237, April 2002.

[9] J. Kienzle and A. Romanovsky, "A Framework Based on Design Patterns for Providing Persistence in Object-oriented Programming Languages," IEE Software Engineering Journal, Vol. 149, pp. 77-85, June 2002.

[10] T. Komiya, H. Ohsida, and M. Takizawa, "Mobile Agent Model for Distributed Objects Systems," Proceedings of the 5th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, pp. 62-69, April 2002.

[11] G. Muzak, I. Cavrak, and M. Zagar, "The Virtual Laboratory Project," Proceedings of the 22nd Internal Conference on Information Technology Interfaces, pp. 241-246, June 2000.

[12] M. K. Perdikeas, F. G. Chatzipapadopoulos, I. S. Venieris, and G. Marino, "Mobile Agent Standards and Available Platforms," Computer Networks Journal, Vol. 31, 1999.

[13] K. Schelfthout, T. Coninx, A. Helleboogh, T. Holvoet, E. Steegmans, and D. Weyns, "Agent Implementation Patterns," Proceedings of the OOPSLA 2002 Workshop on Agent-Oriented Methodologies, pp. 119-130, November 2002.

[14] L. M. Silva, G. Soares, P. Martins, V. Batista, and L. Santos, "Comparing the Performance of Mobile Agent System: A Study of Benchmarking," Computer Communications, Vol. 23, April 2000.

[15] N. P. Sudmann and D. Johansen, "Supporting Mobile Agent Applications Using Wrappers," Proceedings of the 12nd International Workshop on Database and Expert Systems Applications, pp. 689-695, September 2001.

[16] W. Suwu and A. Das, "An Agent System Architecture for E-commerce," Proceedings of the 12nd International Workshop on Database and Expert Systems Applications, pp. 715-726, September 2001.

[17] H. W. Tzeng and C. M. Tien, "Design of a Virtual Laboratory for Teaching Electric Machinery," Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, Vol. 2, pp. 971-976, October 2000.

[18] Y. Wang, "Dispatching Multiple Mobile Agents in Parallel for Visiting E-shops," Proceedings of the 3th International Conference on Mobile Data Management, pp. 61-68, January 2002.