Building an Agent-Based Laboratory Infrastructure for Higher Education

Hong LIN, Khoi NGUYEN, Muna SAQER
Department of Computer and Mathematical Sciences
University of Houston-Downtown
1 Main Street, Houston, Texas 77002, USA

ABSTRACT

We present an ongoing project at the University of Houston-Downtown (UHD) that aims to build a grid as a laboratory environment to support undergraduate education. We intend to use this PC clusters centered grid to allow students to perform laboratory exercises through web interfaces. In order to accommodate lab packages of a growing number of courses, we design the system as a modular system using multi-agent modeling. Students are recruited to implement the units of the system as senior student project topics or research activities sponsored by the Scholar's Academy of UHD. Through these projects, we geared our research toward higher education and provided students with opportunities to participate in building a computational infrastructure for curriculum improvement. This is very important for a minority-serving institution (MSI) with limited resources such as UHD.

Keywords: Undergraduate Education, Education Infrastructure, Laboratory, Grid, Multi-Agent Systems, Computer Cluster.

1. INTRODUCTION AND BACKGROUND

Agent-oriented design has become one of the most active areas in the field of software engineering. The agent concept provides a focal point for accountability and responsibility for coping with the complexity of software systems both during design and execution [1]. It is deemed that software engineering challenges in developing large scale distributed systems can be overcome by an agent-based approach [2]. In this approach, a distributed system can be modeled as a set of autonomous, cooperating agents that communicate intelligently with one another, automate or semi-automate functional operations, and interact with human users at the right time with the right information.

A distributed learning system typically involves many dynamically interacting educational components, each with its own goals and needs for resources while engaged in complex coordination. It is very difficult to develop a system that could meet all the requirements for every level of educational hierarchy since no single designer of such a complex system can have full knowledge and control of the system. In addition, these systems have to be scalable and accommodate networking, computing and software facilities that support many thousands of simultaneous users concurrently working and communicating with one another [3].

We have studied the implementation of Collaborative Agent System Architecture (CASA) [4] with Chemical Reaction Model (CRM) [5-6]. CASA is a model that can catch the interactive and dynamic nature of e-learning systems. Our research results are published in [7-8]. Following our existing work on the design methodology of multi-agent systems, we exploit this methodology in a project that aims at a grid system for laboratory use in undergraduate education. The new method will provide a solution to current problems in design of comprehensive environments to support lab activities in teaching courses on parallel/distributed systems and networks, to respond the increasing need for effective convey of the knowledge of current technology to students to equip them for a career in the modern fast-changing computer industry. One of the most important parts of this project is designing labs that can be performed through the Internets. Our first step is implementing lab packages for our parallel computing and computer networking courses in a grid that encompass lab facilities centered at a Beowulf cluster. We will then extend our lab environment to include other CS and Mathematical courses.

The challenge we are facing, however, is that we need to build an infrastructure that will accommodate multiple courses in different disciplines. The problem we are solving include: (1) an interface that is extensible to incorporate more lab modules and customizable to different course structures; and (2) an computational backbone that provides services for various lab activities, such as testing a parallel program, production of network phenomena, performance analysis. Performing these activities requires coordination among multiple nodes. Also, the architecture of the system requires extensibility and scalability to accommodate multiple course modules. To address the first problem, we follow the practice we had when we built the lab package for our CSI course. Outstanding features of this package include a lab explorer that allows students to browse through lab activities and the ability to invoke programs through the interface. We adopt the same structure in the lab package we designed for our parallel computing and networking courses. To address the second problem, we need to build an array of servers that run on a computational grid. A grid is a system of networked computing and storage sources (see Grid.org) that allows the sharing of information and computational powers. The grid is also a platform on which experiments of distributed data processing and computation can be exercised. Services are provided by different nodes of the grid system. The design of the grid must meet certain criteria so that the incorporation of any unit fits into our long term blueprint. For example, as aforementioned, the underlying infrastructure must support incremental and dynamic addition of lab exercises into the lab package. This is to support our ongoing construction of closed labs for our courses in parallel computing, computer networking, and other courses [9]. On the other hand, however, the complexity of the system makes the design of its infrastructure difficult. Our existing research results suggest that the agent model is a powerful tool to solve problems in a distributed system.

Therefore, we use agent technology to build the architecture of the grid system to manage the coordination and communication among the nodes and handle the load balancing issues. We envision that our practice will provide a solution to the problem of immersing current technologies into educational efforts which have been continuously made at UHD through the development of a comprehensive lab environment.

2. THE PROJECT

As described above, agent system provides an architectural model for distributed networking system. As an active research area, the study in agent technology strives to apply intelligent information processing technologies to complex software systems. Features of an agent system have been summarized in the literatures, for example, according to Griss and Pour [10], an agent shows a combination of a number of the following characteristics: autonomy, adaptability, knowledge, mobility, collaboration, and persistence. These features exist in different types of agent systems such as collaborative agents, interface agents, reactive agents, mobile agents, information agents, heterogeneous agents, and economic agents. Because of the Gamma language's higher-order operations and its closedness to specifications (no artificial sequentiality), these features can be described directly without being adapted to fit into proprietary frameworks. Since this paper focuses on the architectural design of the grid system, we omit some technical details about CRM. Interested readers can refer to our publications for explanations of our methods. In [7], a sequence of case studies shows that features of various agent systems can be grasped by the Gamma language succinctly. In [8], we give a comprehensive example of specifying a course material maintenance system using the Gamma language. In addition, part of our work in constructing the cluster is presented at the 16th IASTED International Conference on Modeling and Simulation (MS 2005) [9].

The Design

The project includes a sequence of major steps: grid construction, lab design, client/server model definition, definition of the interface of functional units, agent-based architecture construction, a module language for program refinement, and architecture specification in the Chemical Reaction Model. Our plan can be described as a pyramid-shaped model illustrated in Figure 1.

The system will be designed using a bottom-up strategy (the Design Theme). We construct the grid and design lab modules using existing toolkits, such as Globus Toolkit 3, Java, and Apache Server. The services provided by the system are implemented in client/server architecture. A Java based user interface delivers the services on the web. Servers run on the clusters. Multiple servers interact with one another in the agent based infrastructure. A formal definition of the interfaces of functional units of the system forms the basis for multi-agent system design. Each agent is then designed in the Module Language we have proposed for specifying multi-agent systems [8]. The overall system is specified in the Chemical Reaction Model. In Figure 1, we can see the multi-agent system is the conceptual model for implementing grid services, and the interfaces of functional units define the interaction among functional units and are the central part of the agent system. The interface also separates the architectural design from the design of individual functional units.

Adding/deleting services or features in the grid can be done in a top-down strategy (the Application Theme). If a service of a new type is to be added into the system, for example, it is added into the architectural specification. Through an automatic transformation procedure, the specification is re-written into a multi-agent system in the module language. The actual program that codes the services is then incorporated into the system through the standard interface. Therefore, updating services or lab exercises in the system will not cause any change in other parts of the system and correctness and reliability of the system can be ensured to the maximum extent.

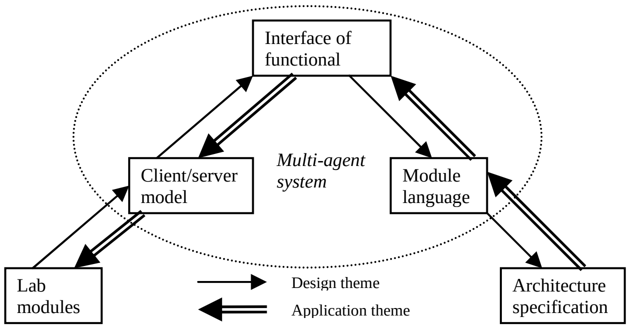


Figure 1 The pyramid model of the project

A Show Case

The following is a list of labs we are using in our parallel computing and networking courses. These labs are carefully designed based on the goals of the course set forth in its syllabus and pursuit in our teaching experience. Lab topics are either typical topics of the area or problems we tackle within the course projects. Our lab design emphasizes the operability and vividness as well as the manifestation of the basic concepts and typical technologies. We also address the role played by the cluster when we design the labs.

- *Topology*: Circuiting messages in a ring
- *Collective communications*: Matrix transpose
- *Group management*: Matrix multiplication with Fox's algorithm
- *Scientific computation*: Solving linear systems with Jacobi's algorithm
- *Combinatorial search*: Traveling salesman problem
- *Parallel I/O*: Vector processing - Summation
- *Performance analysis*: Visualization with Upshot – Trapezoidal rule problem
- *Parallel library*: Solving linear system with ScaLapack
- Scalability analysis: Bitonic sorting
- *LAN configuration*: The use of NICs and hubs
- *Network analysis*: Monitoring a chat room
- *Address resolution*: Experiment with ARP burst
- *IP masquerading*: Clustered web servers
- *WAN configuration*: The use of routers
- *Performance tuning*: Deal with congestion
- *Service configuration:* The configuration of a networked file system:

Here we show one example lab we have designed. This lab allows students to use standard metrics to analyze the performance of a parallel program. The students predict the performance of the parallel program they choose, load the program onto the cluster, compile and run the program, and then compare the predicted results to the experimental results. As illustrated in Figure 2, one lab session is organized in a series of tasks and each task a series of activities. In this lab, students study some standard measurement criteria, viz. speedup and efficiency, for performance analysis of parallel algorithms in Task Activity 1 and 2, and predict the speedup and efficiency of the chosen program given the size of the problem input and the number of nodes in Activity 3. Task 2 requires the students to load the chosen program onto the cluster and then compile the code. The students can click on the C++ Compiler button in the bottom of the page to compile the code once the loading is finished. Task 2 Activity 1 walks the students through the program loading process. Activity 2 asks the students to compile the code. The code is then checked in Activity 3 by a program to ensure its correctness. Erroneous code causes the students to be asked to correct the code till it is errorless. In Task 3, the students can analyze the experimental performance of the program by using MPICH JumpShot profiling software and compare the experimental results to the theoretical predicts, which have been done in Task 1. In Activity 1, the students are required to insert profiling commands into the program and obtain a profile of the program by running it. In Activity 2, the students start up the JumpShot program from the program menu to obtain a Gantt chart of the program. The students then calculate the actual performance data by using the logged timing data and compare the experimental results to the predicted. This is done in Activity 3. Figure 3 shows some snapshots of the lab activities. Figure 3(a) shows the window that takes the student's response to performance prediction questions. Figure 3(b) shows the moment when the student opens a program through a dialog window and monitor the execution of the program through a popup window. Figure 3(c) shows a text window in which the student adds profiling statements into the program.

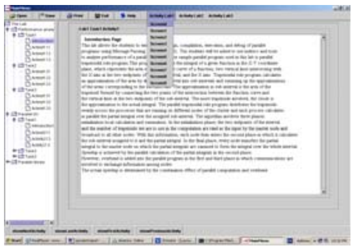
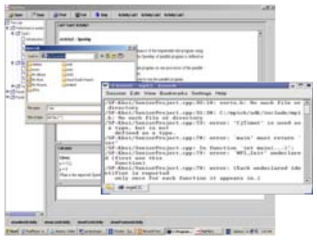
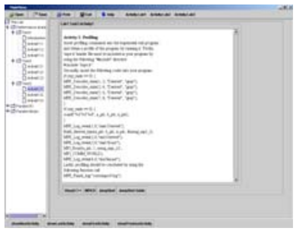


Figure 2 The Main Window of the Lab platform

a. Performance Prediction



b. Compilation and execution



c. Profiling

Figure 3 Snapshots of Lab – Performance Analysis

3. STUDENT PROJECTS

Research at UHD is tightly coupled with its educational programs. Student involvement is an indispensable part of our research. For years, UHD's Scholar's Academy (SA) has been pairing up faculty and students and hosting organized research. Outstanding students are invited to present their work at the annual Student Research Conference (SRC). The Department of Computer and Mathematical Sciences has also widely recruited students in building the Labs and developing lab software. Senior student projects have been carried out throughout the design of the laboratory. Also, volunteering student research assistants constantly work in the UHD Grid Computing Lab to configure the clusters and implement research modules. With the limited resources of an undergraduate institution such as UHD, it is very important to involve students in research programs, not only to create activities for students to obtain hands-on experiences, but to couple research and education seamlessly. In the following, we present three student projects that are directly related to the project of building an integrated lab environment.

A Pioneer Work: Cluster Configuration and Testing

Parallel Computing course is an important part of Computer Science curriculum. We teach students to use Message Passing Interface (MPI) to design and test parallel programs. Since the Parallel Computing course is a writing course, students are also given a writing project which requests the students to write a report about applications of parallel programming. We are building a lab environment which can give the students hand-on experience in solving real-world large scale applications so that the students can get an image of the real performance of the parallel programs. To this end, a Beowulf cluster is constructed, configured, and tested using 15 similar computers and 1 newer, faster server using the MPI. The operating platform is Fedora 2 (Linux Red-Hat).

The Beowulf cluster is a rather simple architecture that many could recognize. There are two different configurations for the Beowulf: Class I and Class II. The Class I Beowulf is built entirely of commodity hardware and software. This type of cluster is usually less expensive than the Class II clusters that use specialized hardware to achieve higher performance. Intuitively, this project is geared towards a Class I Beowulf cluster. As shown in Figure 4, it consists of 15 nodes (computers) and 1 server. The server operates as the *master* node, and the 15 nodes serve as computational *slaves*. They are connected via a high-speed Ethernet and switch. All day-to-day operations and coding are done on the server.

Figure 4 Class I Beowulf Cluster

This project focuses on the MPICH implementation on a Class I Beowulf cluster running on Fedora 2. Since MPICH is mirrored across the cluster, users can send MPI commands such as *mpirun* on the cluster. Packets of data are sent to a desired number of nodes on the cluster that are in turn sent back to the server to accomplish a given task.

The whole purpose of this Class I cluster is to achieve equivalent or greater processing compared to specialized computers and/or scalable parallel machines. Due to limited resources, a performance comparison to a Class II cluster or a scalable parallel computer (SPC) could not be made. Benchmarks on this cluster were made using MPI programs on a time-based analysis. Whatever program ran on the cluster, timings from start to finish of calculations were recorded.

Design of Lab Interface

The project goal is to design a layout interface for performing lab activities on a cluster of PCs. The main program is stored on the root node of the cluster. Students can upload a program onto the cluster, run it, and monitor the result. The lab allows students to use existing parallel computers, high-performance workstations, and vector computers to experiment using Linux operating system and java interface program. Simple parallel architecture ideas and basic analytical models of parallelism will be presented. The students will be able to run sample C++ program and see and analyze the result.

For this project, the first step was to carefully and manually design the lab layout and sketch the main menu layout. The second step was to add the lab’s tasks and lab’s activities to the main menu. The following step was to add activities such as print, close, open, save for the labs. The next step was to start thinking about how to automate the process. Java introduced the layout for the GUI (Graphical User Interfaces). It allows fields to automatically grow and shrink depending on how much screen is available. In this project some of the features of great quantity objects are combined with features from Java Layouts to create labs layouts.

The targeted versatility of the use of the lab package is ensured by the following criteria in our development plan:

- **Scalability**: We can add new nodes into the grid or delete nodes from the grid.
- **Extensibility**: The design of the lab environment makes it possible to incorporate other software packages to enhance the functionality.
- **Customizability**: Object-oriented design of the architecture and standardized interfaces of objects make the lab composition easy.
- **Accessibility**: The lab software creates an interface for the users to control node activities through the web browser.
- **Robustness**: The client-side component of the lab software ensures the correctness of the program before loading it onto the grid for execution.

The architecture of the lab package is an extension of the framework of the labs designed for our CSI course. The overall design for this GUI was that it would be simple, reliable, and portable. Although currently the software package developed is merely a prototype, it allows for further extension in accordance with our architectural design depicted as above. The GUI was developed with Java. Java is well known for its stability and portability. The GUI was developed with the Java 2 SDK with netBeans by Sun Microsystems. The main points for the flow of the GUI are:

- Card/Tab-Layout style Window
  - Menu bar with options
  - exit, help
- Tabs will contain:
  - Introduction (background information on clusters/MPI)
  - User window for loading, compiling/building, and running their MPI programs
- Demonstration Programs
  - Sorting Algorithms & sample distribution programs

Layout manager is an abstract class which handles constraints and simplifies implementation of new layouts. It's used as the super class for most of the other layouts. It provides a configurable horizontal and vertical margin around all components. In addition, it has an option to allow the layout to include invisible components in its layout policy. The main menu is designed in a way that it contains all other layouts screens. The main menu frame obtains the tree, the panel, and the upper and lower toolbars. Each toolbar has some action activity provide by the menus or buttons. The panel obtains the lab tasks and activities. The screens cards are controlled by the tree, the panel or the toolbars items (Figure 5).

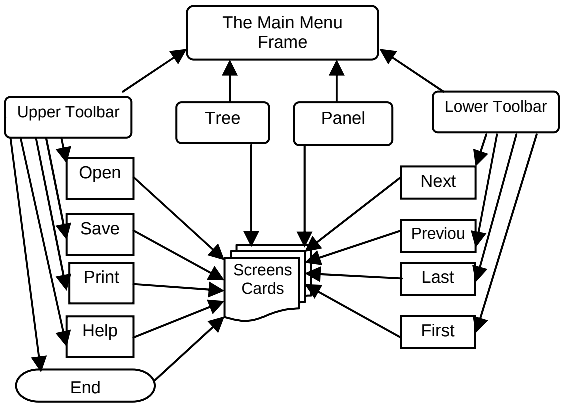
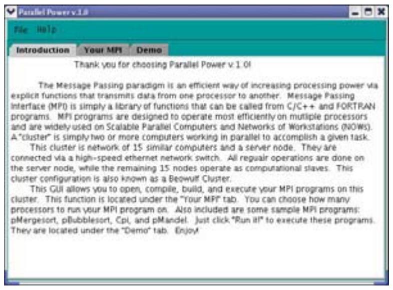


Figure 5 Interface Layout Design

The GUI was coded utilizing the JSwing packages and forms. All layouts were created as NULL layouts, for it offered more flexibility in placing items such as text windows and action buttons. The GUI was built upon a *Frame* form. Frames are typically used as stand-alone top-level windows as the main user interface to the application. Therefore, it was the optimal choice for the GUI.

The *Introduction* window/tab has general information on the cluster, MPI, and the GUI. A simple JTextpane was added to the base frame that contained the text. Figure 6 is the actual screen shot of the *Introduction* window.



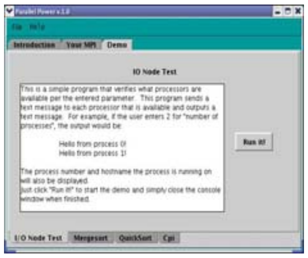
Thank you for choosing Parallel Power v 1.0!

The Message Passing paradigm is an efficient way of increasing processing power via explicit functions that transmits data from one processor to another. Message Passing Interface (MPI) is simply a library of functions that can be called from C/C++ and FORTRAN programs. MPI programs are designed to operate most efficiently on mutliple processors and are widely used on Scalable Parallel Computers and Networks of Workstations (NOWs). A "cluster" is simply two or more computers working in parallel to accomplish a given task.
This cluster is network of 15 similar computers and a server node. They are connected via a high-speed ethernet network switch. All regualr operations are done on the server node, while the remaining 15 nodes operate as computational slaves. This cluster configuration is also known as a Beowulf Cluster.
This GUI allows you to open, compile, build, and execute your MPI programs on this cluster. This function is located under the "Your MPI" tab. You can choose how many processors to run your MPI program on. Also included are some sample MPI programs: pMergesort, pBubblesort, Cpi, and pMandel. Just click "Run it!" to execute these programs. They are located under the "Demo" tab. Enjoy!

Figure 6 The Introduction

For a user to load, compile/build, and run MPI programs on the cluster, a separate window, called *Your MPI*, was created. Here users can open their MPI program source code, compile & build it, and execute it. This window was added as a JPanel. JPanels are used to place other objects such as buttons and text areas on. On this panel, *Open, Build,* and *Run it!* buttons were added. Actions were assigned to each button to execute the said tasks. Also, a JTextArea was also added to display important messages and instructions. Mouse event listeners were added as reminders for the action buttons. Figure 7(a) shows the window displaying instructions. Figure 7(b), (c), and (d) show actual screenshots of a "Open File", "Build", and "Run it!" action, respectively, in the *Your MPI* window:

The "build" action button builds the opened file. Hence, the "Run it!" button executes the program. Also added was the *Demo* window for users to run sample programs. Within this window are a set of more tabs – one for each sample program. The sorting programs, Mergesort and Quicksort are located here to sample. Also included are the Cpi and a basic I/O program. This GUI offers the bare minimum features for running MPI programs on the cluster. Further development can be made on the GUI in areas of graphics and content.

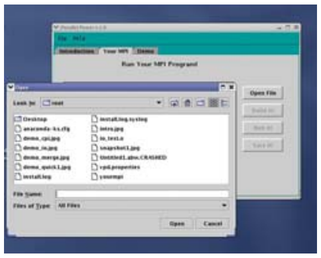


This is a simple program that verifies what processors are available per the entered parameter. This program sends a text message to each processor that is available and outputs a text message. For example, if the user enters 2 for "number of processes", the output would be:

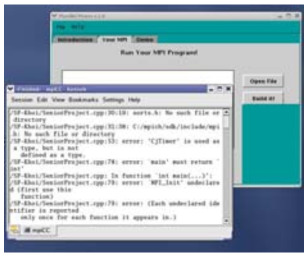
Hello from process 0!
Hello from process 1!

The process number and hostname the process is running on will also be displayed.
Just click "Run it!" to start the demo and simply close the console window when finished.

(a) The Instruction Window

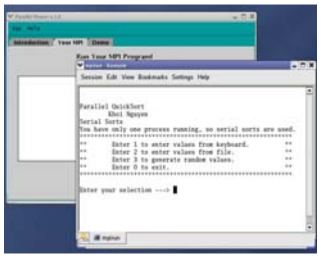


(b) The "Open file" Window



```
/SP-Khoi/SeniorProject.cpp:30:18: sorts.h: No such file or
 directory
/SP-Khoi/SeniorProject.cpp:31:38: C:/mpich/sdk/include/mpi
.h: No such file or directory
/SP-Khoi/SeniorProject.cpp:53: error: `CjTimer' is used as
 a type, but is not
   defined as a type.
/SP-Khoi/SeniorProject.cpp:78: error: `main' must return `
int'
/SP-Khoi/SeniorProject.cpp: In function `int main(...)':
/SP-Khoi/SeniorProject.cpp:79: error: `MPI_Init' undeclare
d (first use this
   function)
/SP-Khoi/SeniorProject.cpp:79: error: (Each undeclared ide
ntifier is reported
   only once for each function it appears in.)
```

(c) The "Build" Window



```
Parallel QuickSort
        Khoi Nguyen
Serial Sorts
You have only one process running, so serial sorts are used.
*********************************************************
**       Enter 1 to enter values from keyboard.        **
**       Enter 2 to enter values from file.            **
**       Enter 3 to generate random values.            **
**       Enter 0 to exit.                              **
*********************************************************

Enter your selection ---> 
```

(d) The "Run it" Window

Figure 7 Interface to Run a Program

Both the student projects have been presented at the Student Research Conference of the University of Houston-Downtown (Figure 8).

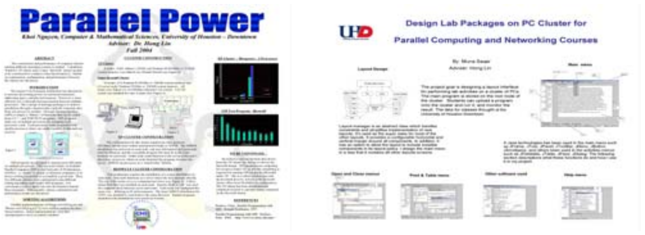

Parallel Power
Design Lab Packages on PC Cluster for Parallel Computing and Networking Courses

Figure 8 Student Posters

4. CONCLUSIONS

We present a method for designing a computational grid that supports online lab exercises, as part of our Information Technology track of curriculum design. A lab package is designed to support the learning process in courses of parallel computing and networking. The grid is centered at a Beowulf cluster, which provides a computational backbone of the grid, and services are deployed in distributed nodes of the computing networks and organized by a multi-agent system. To address high level architectural design issues, such as scalability, extensibility, and modularity, we use Chemical Reaction Model to formally specify the architecture and we facilitate a transformational method for implementing the system to the module interface level. We have developed the lab with an interface that accommodates different lab activities in different courses and demonstrated the design by show cases. Students have been involved in the implementation of the laboratory in forms of senior student projects and SA sponsored research projects. This makes our research coupled with education tightly.

5. ACKNOWLEDGEMENT

This research is partially supported by NSF grant “Acquisition of a Computational Cluster Grid for Research and Education in Science and Mathematics” (#0619312). Some of the student research projects are supported by U.S. Army Research Office Award #W911NF-04-1-0024 through Scholars Academy of UHD.

6. REFERENCES

[1] E. Yu, “Agent-oriented modelling: software versus the world”. **Agent-Oriented Software Engineering AOSE-2001 Workshop Proceedings**, Montreal, Canada, May 2001. LNCS 2222. Springer-Verlag, Berlin, Germany, 2001, pp. 206-225.

[2] G. Paquette, “Designing Virtual Learning Centers”. In H. Adelsberger, B. Collis, J. Pawlowski (Eds). **Handbook on Information Technologies for Education & Training**, Springer-Verlag, Berlin et al, 2001, pp. 249-272.

[3] M.A. Vouk,, D.L. Bitzer, and R.L. Klevans, “Workflow and end-user quality of service issues in web-based education”. **IEEE Trans. on Knowledge and Data Engineering**, 11(4), 1999, pp. 673-687.

[4] R.A. Flores, R.C. Kremer, and D.H. Norrie, “An architecture for modeling internet-based collaborative agent systems”. in T. Wagner & O.F. Rana (Eds.), **Infrastructure for Agents, Multi-Agent Systems, and Scalable Multi-Agent Systems**, LNCS1887, Springer-Verlag, London, UK, 2001, pp. 56-63.

[5] J.-P. Banâtre, and D. Le Metayer, “The Gamma model and its discipline of programming”. **Science of Computer Programming**, 15, 1990, pp. 55-77.

[6] J.-P. Banâtre, and D. Le Metayer, “Programming by multiset transformation”. **CACM**, 36(1), 1993, pp. 98-111.

[7] H. Lin, “A language for specifying agent systems in E-Learning environments”. in: F.O. Lin (ed.), **Designing Distributed Learning Environments with Intelligent Software Agents**, IGI Global, Hershey, PA, USA, 2004, pp. 242-272.

[8] H. Lin, and C. Yang, “Specifying Distributed Multi-Agent Systems in Chemical Reaction Metaphor”. **The International Journal of Artificial Intelligence, Neural Networks, and Complex Problem Solving Technologies**, Springer-Verlag, 24(2), 2006, pp. 155-168.

[9] H. Lin, and K. Nguyen, “Classroom simulation of massive parallel computing”. **Proc. The 16th IASTED International Conference on Modeling and Simulation** (MS 2005), Cancun, Mexico, May 18-20, ACTA Press, Calgary, AB, Canada, 2005, pp. 45-50.

[10] M. Griss, and G. Pour, “Accelerating development with agent components”. 34(5), **Computer**, *IEEE*, 2001, pp. 37-43.