Exploiting Agent Technology

Hong Lin
University of Houston-Downtown, USA

INTRODUCTION

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 (Yu, 2001). It is deemed that software engineering challenges in developing large-scale distributed systems can be overcome by an agent-based approach (Paquette, 2001). 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 (Vouk, Bitzer, & Klevans, 1999).

We have studied the implementation of collaborative agent system architecture (CASA) (Flores, Kremer, & Norrie, 2001) with a chemical reaction model (CRM) (Banatre & Le Metayer, 1990, 1993). CASA is a model that can catch the interactive and dynamic nature of e-learning systems. Our research results are published in Lin (2004) and Lin and Yang (2006). 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. The unified model in chemistry-inspired languages will enable formal specification of an evolving system and provide a framework for top-down design of the entire system.

BACKGROUND

With the fast innovation of computer and communication technologies, computer curriculum is being adapted to accommodate teaching modules that enhance teaching effectiveness by utilizing frontier technologies. For example, the Department of Computer Science, University of Houston-Downtown (UHD), is building an information technology (IT) option, which consists of courses in modern computer technologies defined by the current industrial desires, in its Computer Science degree program, to respond to 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 Internet. 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 problems we are solving include: (1) an interface that is extensible to incorporate more lab modules and customizable to different course structures; and (2) a computational backbone that provides services for various lab activities, such as testing a parallel program, production of network phenomena, and 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 (Lin & Nguyen, 2005). 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.

THE PROJECT

Goals and Objectives

The barrier in front of us is the integration of various networking technologies into one client/server model to provide a uniform lab environment for different lab activities. Given the targeted use of this solution, we need to define and implement the infrastructure that balances functionality and reliability. Based on our existing research experience, we desire a formal system to define the architecture of the grid system so that the development of the services and lab modules will no longer be pursued on case-by-case basis. The formal system must provide a language for the architecture specification, and a derivative method for system refinement. Architectural design should focus on system topology, interactions among system units, and dynamic features of the system, without involving proprietary platform information such as the operating systems on individual nodes, programming languages for program units, and vendor specific machine features. With the formal definition of the architecture on hand, interfaces among system units will be formally specified and design and deployment of each functional unit, such as a lab module, will not affect other units or cause any revision on the overall system.

Unfortunately, traditional formal methods in computer sciences are usually oriented to typical statically defined problems and not suitable for large-scale dynamic systems. Although there are attempts for developing formal methods in parallel and concurrent programming, no formal methods have ever been systematically used on evolving areas such as grid computing. We need a new model that can address the dynamic nature of a complex system without any presumption on the computation model.

As described earlier, an agent system provides an architectural model for a 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 (2001), 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 (Weiss, 2003). 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 Lin (2004), a sequence of case studies shows that features of various agent systems can be grasped by the Gamma language succinctly. In Lin