Chapter VII

Integrating Agents and Web Services into Adaptive Distributed Learning Environments

Fuhua Lin
Athabasca University, Canada

Larbi Esmahi
Athabasca University, Canada

Lawrence Poon
Athabasca University, Canada

Abstract

This chapter discusses an integrated approach to designing and developing adaptive distributed learning environments. It presents a distributed learning environment based on agent technology and Web services technology. Agents are expected to be used as the core components in intelligent distributed learning environments because of their inherent natures: autonomous, intelligent, sociable, etc. However, to integrate agents into existing legacy learning environments or into heterogeneous

learning environments, one may encounter many difficulties. They may be technical issues, economical issues, social issues, or even political issues. Web services technology, on the other hand, characterized by its standardized communication protocol, interoperability, and easy integration and deployment, is an excellent complimentary partner with agents in distributed learning environments. The integration of Web services and agents simplifies the complexity of development, saves time, and, most important of all, makes distributed learning environments feasible and practical. To take advantage of the merits of agents and Web services, we advocate agent-supported Web services in designing and developing distributed learning environments.

Introduction

Over the last few years, universities and colleges have made substantial progress in using the Internet to deliver courses. This is referred to as "e-learning" or digital learning. This trend blurs the differences between information technology applications in education and distance education. While taking courses, students on campuses often extensively acquire distributed learning resources, communicate, and collaborate with other teachers and learners anywhere and at any time. Therefore, campus-based education and distance education, to some extent, tend to be integrated. As well, distance training is frequently used in enterprise training. Distance education and training developed rapidly over the past several years. The research on distance education and distance training has become one of the hottest fields in educational technologies.

The Internet and Web-based distributed learning can potentially deliver personalized course materials and services, and therefore, are potentially able to accommodate a larger variety of learners than what can be accommodated currently.

A distributed learning environment can be implemented practically by using a set of Web services. These Web services offer a set of software artifacts and technologies that service providers or users can modularize and encapsulate with well-defined standard interfaces, host on their platforms of choice, manage and run either locally or remotely, transport over the Internet or any intranet by using standard protocols over and above TCP/IP, locate from central regis-

tries, and exhibit near plug-and-play characteristics with an environment (Geng et al., 2003).

The dynamic and distributed nature of both resources and applications in distributed learning environments requires that software does not merely respond to requests for information but intelligently adapts and actively seeks ways to support learners and educators. Agent technologies have become some of the primary weapons in the arsenal aimed at addressing the emergent problems and managing the complexity of such environments. The agent-based approach is suitable for supporting distributed learning, because relationships among learners, courses, and instructors last for a considerable period of time. Software agents can be implemented in a granular fashion, achieving results quickly. Intelligent software agents contain a level of intelligence.

The purpose of this chapter is to discuss how to facilitate the creation of dynamic, intelligent, and autonomous services to achieve adaptive distributed learning using intelligent software agents and agent-supported Web services.

Background

Distributed Learning vs. Distance Learning

Traditionally, distance learning has emphasized delivering educational resources to learners in remote districts or working full-time, and providing the opportunities for open education. Distributed learning, however, enables learners to get and use educational resources distributed in different remote places. In distributed learning, the learners may be either on or off campuses. Distributed learning is based on networked learning environments. With both distance learning and distributed learning, the instructor and the learner do not have to be in the same physical location at the same time.

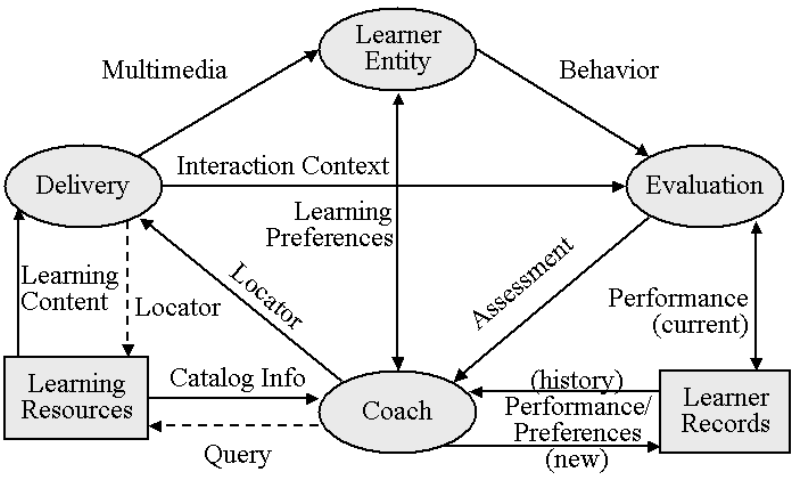
Distributed Learning Environments

A distributed learning environment in the real world can be examined in several ways. One way is to think about the main educational components and the interactions among them and their environments. Our analysis of distributed learning environments begins with an overview of the various entities compris-

ing the distributed learning environments and then the relations among the various entities.

IEEE's learning technology systems architecture (LTSA) provides a model for studying learning environments (http://ltsc.ieee.org/). LTSA is generic enough to represent a variety of different learning systems from different domains. Figure 1 shows the model used by IEEE's LTSA. In the model, the learner entity is an abstraction of a human learner. The learner entity receives the final multimedia presentation, while the learner's behavior is observed, and learning preferences are communicated with the coach. Then, the coach sends queries to the learning-resources component to search for learning content appropriate for the learner entity. The queries specify search criteria based, in part, on learning preferences, assessments, and performance information. The appropriate locators (e.g., learning plans) are sent to the delivery process. The learning-resources component stores "knowledge" as a resource for the learning experience, and the queries can be searched in this repository. The matching information is returned as catalog information, i.e., a set of content tags that can also be seen as "card catalog" entries. The locators are then extracted from the catalog information and used by the delivery process to retrieve learning content and deliver that content as a multimedia document to the learner. The components to the right of the learner entity correspond to performance control. Performance is measured by the evaluation component,

Figure 1. IEEE's LTSA

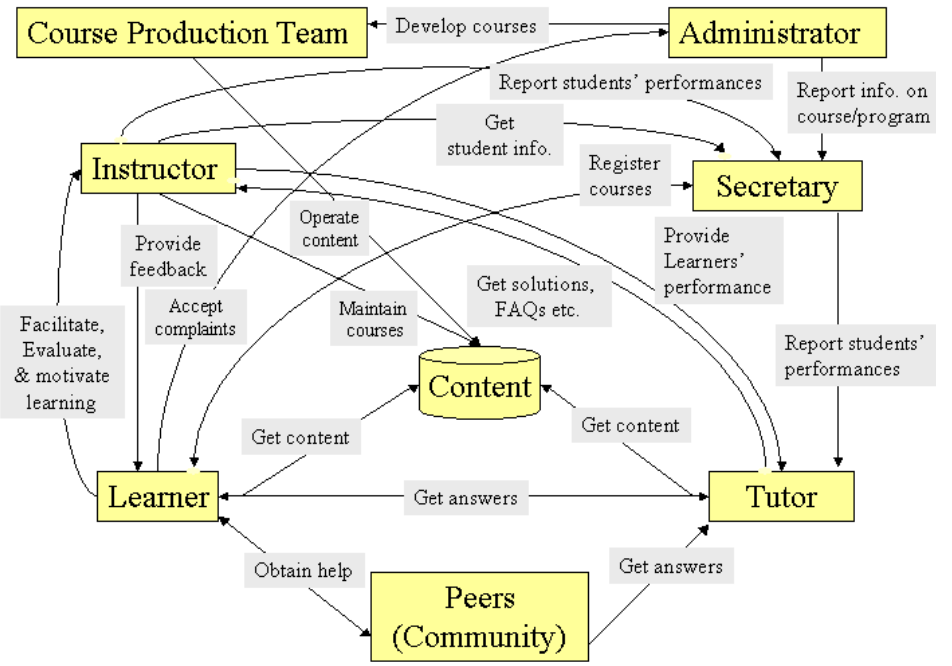


and the measurements are stored in the records database. The coach, when locating new content, can then use the data in the records database.

An Example

The learning environment of Athabasca University (AU) (http://www.athabascau.ca) can be viewed as an example of such an architecture. It has the following core components: the Learner, the Peer, the Instructor, the Content, the Tutor, the Administrator, the Course Production Team, and the Secretary. The "Learner" is an individual who receives a course from the system. The collection of individuals who receive the same course from the system within the same time frame and from the same instructor is referred to as the "Peers." The "Instructor" or "Coordinator" is an individual who facilitates the delivery of pedagogical resources by performing a number of administrative tasks, including course-content sequencing, tutor recruitment

Figure 2. Illustration of the main educational components and dependency model of Athabasca University, Canada



and coordination, learning-progress assessment, learner guidance, authorization of the learners, and advancement of higher modules. The "Content" is the knowledge resources shared by the participants in the learning system. The Content can be divided into three categories: *administrative* content (activities), *pedagogical* content (materials), and *reference* content (library). A "Course" is a subset of the content, possibly supported by distributed knowledge resources (e.g., learning objects repositories), facilitated by one instructor and delivered to a well-defined set of learners. The "Tutor" is an individual who answers the course-related questions of the Learner. The "Administrator" is an individual responsible for planning programs. The "Course Production Team" is responsible for course design, production, and management. The "Secretary" is responsible for student registration, student-information management, and staff-information management. Figure 2 illustrates the main educational components and dependency model of a generic distributed learning environment.

The dependency of the components is also illustrated in the figure. For example, the Learner depends on the Content to get course materials.

The life cycle of a distributed learning course begins with the course planning. The next stage involves course development, in which the course is designed and developed by a team of course developers consisting of professors, editors, and visual designers. Then the course package is delivered to students via the Web or seminars under the coordination and facilitation of an instructor. One or more tutors may be needed for tutoring the course, depending on the size of the class. The course continually goes through course evaluation and course revision until it is closed.

The main advantages of distributed learning over traditional classroom-based learning and traditional distance education are (a) flexibility in the time and location for learning, (b) interactivity among the learning elements due to the Web's multimedia capability, and (c) interactions among instructors, tutors, and learners in synchronous and asynchronous modes. Furthermore, distributed learning has the potential advantages of providing access to distributed educational resources for course authors and students, personalized course materials for individuals, and virtual learning communities for collaborative learning.

However, both distributed learning and the development of distributed learning environments are associated with some challenges.

Issues and Challenges

In most of the existing distributed learning systems, the instructors arrange the course materials in order to cover one or more topics. For example, in Web-based distributed learning environments, the course materials are placed online to make them downloadable or visible to the students, who can use them by following the path established by the instructors. Currently, Web-based distributed learning has the following problems.

First, in terms of system development, software systems for distributed learning are typically complex, because they involve many dynamically interacting educational components, each with its own need for resources, and involve engaging in complex coordination. Developing a monolithic system that could meet all requirements for every level of the educational hierarchy would be very difficult, because no single designer of such a complex system could have full knowledge and control of the system. The systems have to be scaleable and accommodate networking, computing, and software facilities that support many thousands of simultaneous distributed users using different operating systems that can concurrently work and communicate with each other and receive adequate quality of service support (Vouk et al., 1999). Such systems should be easy to extend. A small change in the domain knowledge should not require an intensive system-wide modification to alter the information and all the functions that initiate actions based on that changing information.

Second, the existing Web-based learning management systems are not concerned with individual learner differences and do not adjust to the profiles of individual students (regarding actual skills, preferences, etc.). Currently, we use a curriculum designed just for a specific segment of the potential student population. We build courses around textbooks and other materials designed for that curriculum and do not understand students' situations and requirements and do not utilize the possible contributions that students can make to the learning content and process. As the number of distributed learners increases, serious efficiency problems in course development and maintenance will occur when course developers try to generate personalized course materials.

Third, no appropriate support exists to help handle the constantly increasing demand for and extension of information. As a result, instructors may be spending more time teaching in distributed learning environments than they would spend teaching the same course in a classroom setting. The problem results mainly from the use of generic communication tools, such as e-mail and

computer conferences, which have imposed a significant workload on educators. Because students can participate at any time, this technological advantage also demands more effort from the instructor, the tutor, and the supporting staff. They need to be more responsive at nearly any time.

Fourth, in such a distributed learning environment, the instructors and tutors are not always available online, so the interactions between instructors and tutors and students are asynchronous. As a result, the need for assistance for students is particularly salient.

Last, the features of Web processing (caching and client-side information hiding), in most cases, obstruct the collection of student-performance data.

Solutions and Recommendations

Adaptive learning Environments

An ***adaptive*** learning environment is a learning environment in which an automatic modification is performed at usage time, i.e., during the educational session, and is based on the learner's characteristics. These characteristics, such as the learner's familiarity with the educational subject and the learner's goals and interests, are assumed to be continuously modified during the same educational session. These characteristics are not known prior to each educational session and are automatically detected by the system, through monitoring the learner's actions. According to Jones and Winne (1992), adaptive learning environments can be viewed as the intersection of two traditionally distinct areas of research: instructional science and computational science.

We call an adaptive learning environment for distributed learning a "distributed adaptive learning environment." The main features of such environments are adaptivity and distribution.

Adaptivity

Adaptivity includes adaptive curriculum planning, adaptive sequencing, adaptive course generation and delivery, and adaptive testing. Adaptivity needs intelligence. We need some intelligent tools that are "smart" enough for the

tasks at hand. These tasks need highly deliberative tools for mining databases of learner information and educational resources and taking the collective task knowledge contained in the heads of the educators and codifying it into software systems.

Distribution

From IEEE LTSA, we can see the dynamic and distributed nature of data and applications in distributed learning environments. Therefore, a common understanding seems to be that a distributed learning system must consider a decentralized approach in which overall management is performed "centrally," but course materials (hypertext documents, multimedia documents, technical manuals, scripts, and other applications) are served up locally by using various pieces of software that run on the students' machines (Wang & Holt, 2002). Interactivity and intelligent tutoring capabilities (i.e., various help facilities) must be provided by client-side software, as well.

Unfortunately, none of the currently available distributed learning systems delivers these advanced functionalities, mainly because of the complexity and heterogeneousness of the systems and the lack of methodology for systems modeling, in particular, knowledge modeling.

Web Services Based Approach

A "Web service" is an accessible application that other applications and humans can automatically discover and invoke. An application is a Web service if it is (a) independent as much as possible from specific platforms and computing paradigms; (b) developed mainly for interorganizational situations rather than for intraorganizational situations; and (c) easily able to be integrated (i.e., combining it with other Web services does not require the development of complex adapters) (Dale, 2003).

Web services fundamentally offer new ways of doing business through a set of standardized tools, and they support a service-oriented view of distinct and independent software components interacting to provide valuable functionality. In learning systems, a learning services architecture and learning services stack have been proposed by the Learning Systems Architecture Lab at Carnegie Mellon University. These provide a framework for developing service-based learning technology systems (LSAL, 2003). In this approach, rather than

building large, closed systems, the focus is on flexible architectures that provide interoperability of components and learning content, and that rely on open standards for information exchange and component integration (Blackmon et al., 2003).

However, Web services technologies have several limitations (Huhns, 2003): a Web service knows only about itself, not about its users, clients, or customers; Web services are not designed to use and reconcile ontologies among each other or with their clients; Web services are passive until invoked and cannot provide alerts or updates when new information becomes available; and Web services do not cooperate with each other or self-organize, although they can be composed by external systems.

Due to these limitations, Web services cannot completely overcome the information overload issue in distributed learning environments. Overcoming these limitations appears to require agent-like capabilities.

Intelligent Software Agent-Based Approach

We believe that the software engineering challenges involved in developing large-scale distributed learning environments can be overcome by using an agent-based approach. We can design some processes in a distributed learning system as autonomous, cooperating components that communicate intelligently with one another, automate or semi-automate educational processes, and interact with human users at the right times with the right information.

Agent-oriented software engineering (AOSE) has become one of the most active areas in the field of software engineering. The agent concept provides a focal point for providing accountability and responsibility for coping with the complexity of software systems during design and execution (Eric, 2001). The agent-based approach to developing complex distributed systems has been successfully applied and documented in many domains, including air-traffic control, manufacturing, information retrieval, network management, and entertainment (Wooldridge & Jennings, 1995).

What are "Agents"?

Agents are software programs that operate autonomously when triggered and perform tasks of repetitive nature. Research on agent-based computing has

evoked much interest in multiagent systems (MAS). For intelligent agents to solve problems efficiently, they must cooperate and communicate with each other. A MAS is described, according to O'Hare and Jennings (1996), as "a loosely-coupled network of problem solvers that work together to solve problems that are beyond their individual capabilities." Similarly, the various MASs should be able to interact with each other in the form of virtual communities. The logical extension of MASs is "agent societies"—groupings of agents that come together to collaborate to meet certain common goals (Dellarocas, 2000).

Why Agents?

Agents probably have two main advantages in this context. First, distributing tasks to numerous specialized, fine-grained agents promotes the modularity, flexibility, and incrementality of learning systems and lets new services come and go without disturbing the overall system. The agents have their local knowledge about specific tasks and their autonomy. Limiting the complexity of an individual agent simplifies *control*, promotes *reusability*, and provides a *framework for tackling interoperability*. Second, agents' autonomous nature makes their use a "fire-and-forget" approach, as they are able to react by themselves if they have access to the right data. This central feature of software agents, the ability to independently carry out tasks delegated to them by people or other software, reduces the workload of users.

For example, the vast educational resources available today or tomorrow simply could not function without being able to delegate to software the multitude of tasks that would otherwise be left to armies of people to handle.

Some Related Work

Greer et al. (2001) elaborated the lessons learned from several large-scale real-world deployments in the I-Help (Greer et al., 1998) agent-based peer-help learning support system. The software engineering lessons learned are useful for us in deploying a complex system in the real world for a large number of users. Gavrilova et al. (1999) described a project involving an intelligent MAS for distance learning using the Learner model approach. Conceptualized in the literature (Jafari, 2001) are three types of intelligent agents to assist teachers and learners. Thaiupathump et al. (1999) investigated the effects of

applying intelligent-agent techniques to an online learning environment. These researchers created the "knowbots" that automated the repetitive tasks of human facilitators in a series of online workshops. The findings indicated that the use of knowbots was positively associated with higher learner completion rates in the workshops. Lin and Holt (2001) identified the roles of agents in distributed educational activities. Baylor (1999) defined three major potential educational uses for agents as cognitive tools: (a) managing information overload, (b) serving as pedagogical experts, and (c) creating programming environments for the learners (Baylor, 1999).

The application of agent-based systems to commercial complex distributed systems has generated tremendous interest. However, researchers have been very slow in developing this technology for commercial applications, mainly because of the lack of an accepted industry-standard method for the development and implementation of agent-based systems (Sturm et al., 2003).

Agents

In a distributed adaptive learning environment, agents are seen as software entities that pursue their objectives and perform their tasks while taking into account the resources and skills available to them. The resources include human users, other agents, information, and data. Therefore, we can divide all agents in distributed learning environments into three categories: (a) personal agents, (b) task agents, and (c) regulatory agents.

Personal Agents

A "personal agent" is a virtual representation of a human user operating in a distributed adaptive learning environment and is comprised of several subcomponents that act together to perform the functions required of the agent.

First, a personal agent has a unique identity, which allows it to be recognized by other users as well as gives credibility to the transactions that the virtual self may engage in subsequently.

Second, a personal agent has a memory function that stores the user's preferences, which may be explicitly indicated by the user or learned from past experiences. Similarly, the personal agent also learns from interaction with other agents. For efficiency purposes, the memory module is able to "forget" information that is not used, avoiding storage overload.

Third, the personal agent possesses a processing capability that serves the dual purpose of resource allocation and coordination of functions, e.g., prioritization of various tasks set by the user. To do so effectively, the processing capability interacts with the memory component, as well as with the external sensors, to generate a response. Last, the personal agent contains an internal audit function with the role of monitoring the status of the agent for maintenance purposes as well as restricting the agent's behavior to a generally accepted code of agent practices. To become effective, the personal agent must be able to sense new developments in the environment, filter the variety, as well as effectively respond to changes in the environment.

Task Agents

A task agent is required to perform certain specific tasks, such as providing services, knowledge, and information resources, and also providing intermediary functions such as coordinating and communicating with the other agents. Therefore, the task agent's memory contains specific task-related information in greater depth than the personal agent's memory possesses. The task agent also has a monitoring and learning function that allows it to update its own information through new updates when necessary. Because the task agent is deemed a "common resource" shared by many users, its processing capability comprises a spooling function, in which requests are queued in accordance to their priority. In performing multiple tasks, the resource allocation function determines how many resources should be provided to each uncompleted task.

An Example of Task Agents—User Profile Agent

User profiles represent the users' information needs and preferences. Such profiles can take a variety of forms, ranging from sparse-vectors of document ratings to rich, highly structured representations based upon XML. A profile may be located entirely within the locus of the user's control, e.g., on his or her own personal computer (PC) or personal digital assistant (PDA), or may be retained as one of many profiles on a server controlled by a Web service. Recently, some research has begun to focus on a more distributed approach, in which the Web service needs to be able to deal with the security or privacy of user profiles, and how users can find relevant information even if they do not

want to reveal too much of their profiles. A user profile agent can be developed to fulfill these requirements.

Regulatory Agents

The regulatory agents primarily perform the following two functions, setting standards and auditing. The regulatory agents are separated at various levels and are mainly localized regulatory agents and international regulatory agents. In their standard-setting role, the international regulatory agents set standards mainly for the baseline requirements to which each agent must comply. In turn, the personal agents and the task agents, which constantly monitor the external environment, update their respective baseline requirements to include the latest standards. As an illustration of the rules constituting these baseline requirements, universal rules such as the following are to be included: (a) agents will not harm their masters, (b) agents will not harm other agents, and (c) agents will have to protect themselves from other agents.

In their auditing roles, the international regulatory agents will audit the localized regulatory agents, which in turn, audit the personal agents and the task agents. The scope of the audit services includes the accuracy of an agent's baseline requirements, the program's integrity (the audit will look for any virus infection), and so on. As mentioned earlier, the personal agents and the task agents have limited life spans. At regular intervals, individual personal agents and task agents must report to their localized regulatory agents for a thorough audit screening in order to extend their life spans.

Agent Interactions

The interaction among various agents and, subsequently, among various MASs is based on accepted standards and principles and forms the basis of the interactions among the intelligent agents' societies.

User—Personal Agent Interactions

The major purpose of personal agents is to help (rather than replace) users in handling complexity in the environment. We will explain how a human user and

the personal agent of the user interact through the three stages: *configuration*, *processing*, and *presentation*.

Configuration

Here, the human user configures his or her personal agent's personality, goal, and other characteristics based on the human user's beliefs, values, desires, and intentions. This configuration forms the "DNA" of the personal agent. The instructions must be precise, not open to interpretation, and coherent with the baseline requirements. Also, another source of input comes from the external environment, which is constantly changing, churning out new information exogenously. The agent is constantly learning from interactions with the external environment, adding to its wealth of experience.

Processing

Based on the human user's predetermined preferences and from its learning of the user's unspecified and hidden preferences through observation over time, the personal agent is able to filter the information from the data smog in the environment and extract the relevant information of interest to the user. Subsequently, the personal agent classifies, summarizes, and presents the extracted information in the desired form to the user.

Presentation

The content of the output report is that of highly personalized and topic-based information according to the user's preferences and desires. Moreover, this content is highly relevant, as it is based on real-time events. The content can be presented by multimedia. The determination of the type of presentation format is in accordance with the users' preference and the inherent limitation of the delivery devices. For example, a message in a text format has to be carried as text in the form of a short message system (SMS) if the user is to access the message by using a mobile phone. In a situation in which the user is accessing his or her PDA, then the report can be translated into video form to provide a richer medium to the user. Users can specify how their personal agents can contact them when the users are away from their desks, for example, by desktop computer, mobile phone, or PDA.

At times, the users will want to initiate a conversation with their personal agents when the users are away from their desks. If so, the users can log on to any

terminal, stipulating a password to authenticate them in order to access their personal agents.

Agent–Agent Interactions

The interactions among agents can be divided into three major categories: Personal Agent–Personal Agent, Personal Agent–Task Agent, and Task Agent–Task Agent.

Personal Agent (PA)–Personal Agent (PA)

Communication between two personal agents is important, because they are expected to learn, not only from their past errors, but also from the experiences of other personal agents. Similarly, two or more PAs can collaborate on similar tasks. Just as humans can cooperate in the real world, groups of agents can cooperate with one another to achieve a common purpose. For effective collaboration to take place, personal agents, each with separate competencies, need to coordinate their actions with various task agents. Individual PAs in the environment may take up different roles in a group, e.g., notification, monitoring, and coordination. Moreover, PAs may be hostile to one another. However, based on the baseline requirements, PAs are not allowed to harm one another.

Personal Agent (PA)–Task Agent (TA)

At times, a personal agent needs to interact with one or more task agents in carrying out its assignment. The communication of the personal agent and the task agent(s) varies according to two situations. When the personal agents know the task agents, i.e., when the PAs know explicitly which task agent they wish to contact, this knowledge facilitates direct communication. On the other hand, when the personal agent does not know the identity of the task agents in question, then the use of Web services comes into play.

Task Agent (PA)–Task Agent (TA)

The interaction between two task agents is similar to the Personal Agent–Task Agent interaction.

Regulatory Agents–Other Agents

As mentioned earlier, the role of the regulatory agents is to set new standards. Hence, the international regulatory agents set new standards and broadcast them to the international service providers, which then disseminate the standards down to the local service providers. The other agents, e.g., the personal agents and the task agents, which are constantly monitoring for the latest updates on their respective service providers, in turn, update the latest standards into their baseline requirements. At intervals, the local regulatory agents then ensure that the personal agents and the task agents have accurately and promptly updated their baseline requirements with the latest standards. If the local regulatory agents discover any noncompliance, they correct the mistake themselves and, at the same time, file a report to the owner of the personal agent.

Examples

Agents and Web Services for Courses

The development of agents for courses or course agents involves *pedagogy*, *learning design*, and *learner modeling*. The pedagogical basis of course agents can be built on two underlying educational philosophies: *objectivist* and *constructivist*. The objectivist assumes the learner is an empty vessel that can be filled with knowledge. It leads directly to an instructivist or transmissionist pedagogical approach, where the teacher fills an empty vessel, which is the student (Phillips, 1997).

The other philosophy, the constructivist epistemology, assumes that the learner can build on his or her own knowledge based on an existing set of experiences, so the student is viewed as a "researcher." A major goal of the constructivist approach is to ensure that the learning environment is as rich and interactive as possible.

A course agent can be based on a constructivist learning environment, in which the student and the student's agent can explore at will. However, such discovery learning makes the often-unfounded assumption that the student has research skills. Therefore, a well-designed distributed learning course should incorporate the most appropriate aspects of each learning theory (Gillespie,

1995). We believe that the course agent will be more effective for guided discovery learning based on a constructivist learning theory. The agent can give the student control in discovering knowledge, but the discovery process is supported by additional guidance functions to provide support and feedback.

Personal Agents for Courses

Two types of personal course agents are used in distributed learning: *instructor course agents* and *learner course agents* that assume roles in the participating protocols of learning activities.

An instructor course agent is an assistant of the instructor, helping the instructor generate, deliver, and maintain a course. This kind of agent interacts with other task agents to fulfill the tasks delegated by the instructor, such as broken-link checking, learning-objects recommendation, notification, monitoring, and information gathering.

A learner course agent is a simulated instructor that can provide adaptive course material and instruction appropriate according to the learning process of the individual learner. This kind of agent can be viewed as an authoritative representative of the course author and the instructor. It can also interact with other task agents to carry out the tasks requested by the learner, e.g., answering frequently asked questions (FAQs). Because a learner's profile includes the learning activities he or she has participated in, and the corresponding performance can be easily kept in the environment and made available to agents, an instructor course agent can help the instructor understand learners and make suggestions. A learner's profile contains detailed information, such as a learner's errors and misconceptions, so in the absence of the instructor, the agent is able to give advice to learners when they are learning, in the absence of the instructor (Chan, 1995).

Task Agents for Courses

The task agents include course-planning agents, course assembly agents, course maintenance agents, and evaluation agents.

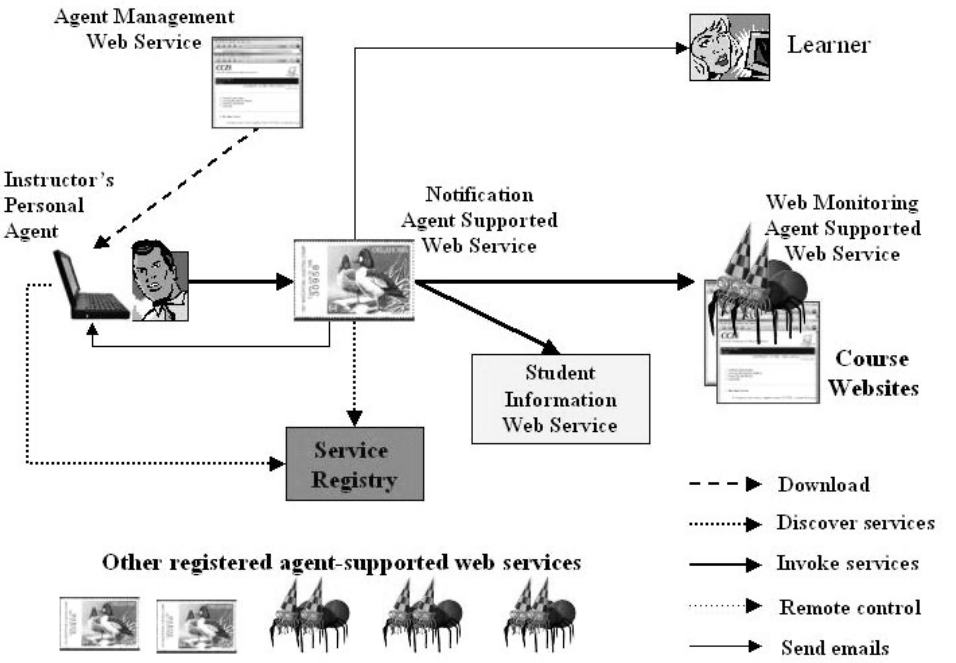
A *course-planning agent* uses information about the learner, resources, and curriculum-planning knowledge, to construct an optimized course plan. When a consistent and complete plan is found, it will be presented to both the student

and the instructor for approval. The course-planning agent uses well-known planning techniques. Each course plan is a curriculum graph. The course-planning agent creates curriculum plans by finding a chain of learning objectives that connect the intended final learning goal with the student's prior knowledge.

After the course-planning agent generates a course plan that partially determines the learning activities and path, a specific "best" solution for learning materials is extracted by using some problem-solving mechanism, such as TAEAS, an heuristic scheduling system (Wagner et al., 1999). TAMES can produce a comprehensive linear instantiation of one possible solution to the problem, based on the constraints, such as the preferred time or the quality or cost of the learning materials. TAEAS generates an initial solution and then enables the student to alter the parameters to retrieve a second solution. Given the constraints supplied by the student, TAMES then offers a variety of solutions.

Once a specific course plan has been approved and scheduled, the student is given access to the selected resources one at a time, interacting via a browser

Figure 3. The system architecture for course maintenance



or other interface. As the student finishes using each resource, the student model is modified to track the student's learning progress, until the student either reaches or abandons the intended learning goal. Difficulties encountered along the way are handled by replanning.

Course Maintenance Agent

One of the merits of Web-based e-learning courses is that it can provide up-to-date information. In order to provide *current*, *correct*, and *complete* materials to students, course instructors need to update e-learning course materials from time to time. The reasons for Web-based course maintenance are threefold.

First, materials in courses in ever-changing fields, such as "Computing and Information Systems," need to be updated more often than in other courses. Working in such a dynamic distributed learning environment, course instructors or instructors often need to review and revise course materials in a short time frame.

Second, because of the complexity of the materials, and the short development cycles within which the materials are produced, our best human efforts are sometimes not adequate to prevent occasional errors from slipping through. Therefore, students should be prepared to encounter the odd minor "glitch" in online courses. However, course instructors should make the necessary adjustments for the benefit of students. Whenever there is a significant change in the content of designated Web pages, students who are interested in the topic and all students who are taking the course will be notified by the course instructor via e-mail.

Third, Web-based course materials have many hyperlinks. These hyperlinks need to be maintained regularly to ensure their availability. However, it is common for an online course to have a few hundreds of hyperlinks. These hyperlinks can be broken for many reasons: Web servers may be down because of hardware failure, Web pages may be relocated to another server, or power may be cut off in another part of the world. To maintain these hyperlinks solely by human efforts is becoming more difficult, if not impossible. The degree of difficulty is hard to comprehend if we consider the fact that hyperlinks can be "dead" and "alive" at different points of time. The need for an automated system to help course instructors maintain hyperlinks is pressing.

The system consists of a couple of Web services located in different places. The Web services include notification Web service, Web monitoring Web service, student information Web service, and instructor information Web service.

There are two types of task agents supporting the Web services: Web Monitoring Agent and Notification Agent. As Web services, they have a dual nature that combines characteristics of Web services technologies and agent technologies: the abilities to be published, found, and called as a service, and the ability to move from platform to platform and make autonomous decisions.

The Web Monitoring Agent is to monitor targeted Web pages and determine whether or not the content in those pages has been significantly changed. The meaning of "significantly changed" is based on a couple of predefined criteria. For examples, the number of hyperlinks or photos increased or decreased, or the content lengths of the Web page increased or decreased by examining its MIME header. If it discovers such changes, it will trigger a Notification Agent to send a message to those students who are interested in receiving the message. Figure 3 shows the system architecture.

Agent Management Web Service

The Agent Management Web service serves as a front end for agent management and deployment through Web technology. A registered user can log-in to download an agent platform and his or her favorite personal agent. Downloading can be through FTP or HTTP protocol. Once logged in, a user can update his or her account information or supply necessary information for agents to run. For example, a course instructor can provide his or her course information, such as a course name and its base hyperlink, so that later, a Notification Agent and a Web Monitoring Agent can make use of this information in order to process the course instructor's request.

The Agent Management Web service also acts as a proxy to an UDDI registry. It assigns unique agent identifications to agents and records agent information, such as Agent Type, and relays these data to an UDDI Registry. The Personal Agent can search from the registry and invoke services provided by the Agent Management Web service. For example, the Personal Agent can ask for the location of a Web Monitoring Agent that is free to work for the user.

The Agent Management Web service is also a Web services provider; it can serve SOAP-compliant clients by exchanging SOAP messages so that users can embed the results returned by the agents into their applications. For example, a course instructor can embed the broken hyperlinks, found by the

Web Monitoring Agent, into his or her Web e-mail application, and send them to students or build his or her course Web pages with the logics taking care of the results returned by the Agent Management Web service. Without coupling with Web Services, agents can notify the course instructor, even though there is a time gap between the broken links found and remedial actions taken. Coupling agents with Web Services, a course instructor has no urgent need to care about the remedial actions if contingent actions had already been taken.

A Personal Agent, as a client of a service, can perform searches of different entries stored in a UDDI. It can then make message and RPC style calls to a Web Service. A Personal Agent is also an interface between the user and the multiagent platform. Through the Personal Agent, a user can manipulate the options provided by other agents. For example, a course instructor can choose how often to receive e-mail from the Notification Agent if broken links are found from his or her course materials. The Personal Agent abides on the platform of the user's computer. Different groups of users are assigned different types of Personal Agents. The assignment is based on their roles in the system. For example, the Personal Agents for course instructors are different from those of the students. Course instructors can choose under what conditions they should be notified if the contents of external links are changed. A Personal Agent is GUI driven and can be used to control all the agents with identification registered under the user name.

Web Monitoring Agent-Supported Web Service

A Web Monitoring Agent has two functions: one is to detect broken links, and the other is to detect Web content changes. It scans the given pages periodically. When the agent detects a significant change (e.g., the link is broken), it sends a message to the *Notification Agent*.

Most of the work is done by agentized and multithreaded class *Spider* (www.JeffHeaton.com). A queue named *Workload* holds the base URL to be processed.

Step 1: A spider opens a connection to the base URL by openConnection().

Step 2: If the connection is failed, the whole process stops.

Step 3: If the connection is successful, the spider parses the Web page to find all the URLs and put them into Workload.

Step 4: The spider checks other information from the MIME header, such as Last Modified, Content Length, etc., and stores this information in the database for comparison purposes.

Step 5: Then the spider opens a connection to the next URL in work.

Step 6: If connection is failed, the spider will report this URL as broken links, store the URL in the database, and open another URL in Workload.

Step 7: If connection is successful, the spider repeats Steps 3 to 6.

Step 8: The spider stops to work until there are no more URLs in Workload.

Notification Agent-Supported Web Service

Incorporating Notification Agents into the system is one of the ways for the MAS to give responses to users. Notification Agent is responsible for sending e-mail on behalf of other agents in the MAS. It is the postman of the whole community. Whenever an agent needs to send e-mail, it asks a Notification Agent to do so. The agent packages an agent message with the necessary details, such as the message, the sender, and the recipient e-mail address, and forwards the message to the Notification Agent. Once a message is received, the agent checks the validity of the information and sends the e-mail accordingly. The Notification Agent makes use of JavaMail class to perform the actual sending. The Notification Agent has no access to sender and recipient information, this information is provided by other Web services through XML request messages.

Student Information Web Service

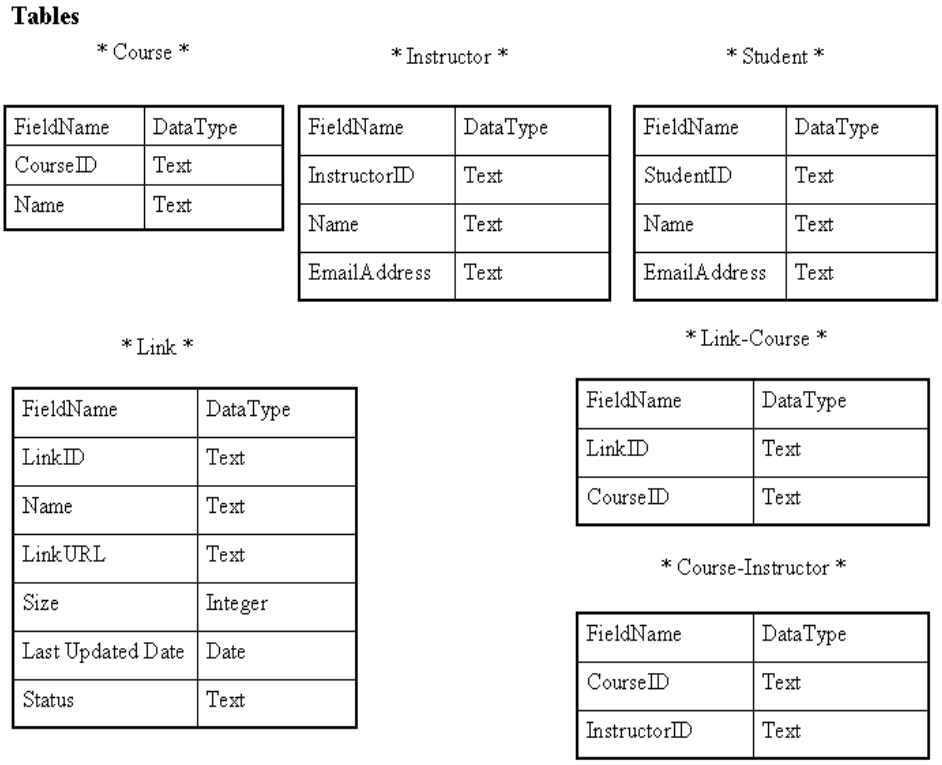
The Student Information Web Service is designed to provide student information. For example, it maintains an e-mail list of those students taking courses in open and distance-learning environments at Athabasca University.

Databases

The database resource includes a student information database, an instructor information database, and a course link database. The simplified data model of the databases is shown in Figure 4.

We implemented the agent system for the online course link maintenance using the architecture shown in Figure 4. The agents and Web services ran on five different servers for testing purposes. The agents and the agent platforms are

Figure 4. Tables for the database



Tables

* Course *

FieldName	DataType
CourseID	Text
Name	Text

* Instructor *

FieldName	DataType
InstructorID	Text
Name	Text
EmailAddress	Text

* Student *

FieldName	DataType
StudentID	Text
Name	Text
EmailAddress	Text

* Link *

FieldName	DataType
LinkID	Text
Name	Text
LinkURL	Text
Size	Integer
Last Updated Date	Date
Status	Text

* Link-Course *

FieldName	DataType
LinkID	Text
CourseID	Text

* Course-Instructor *

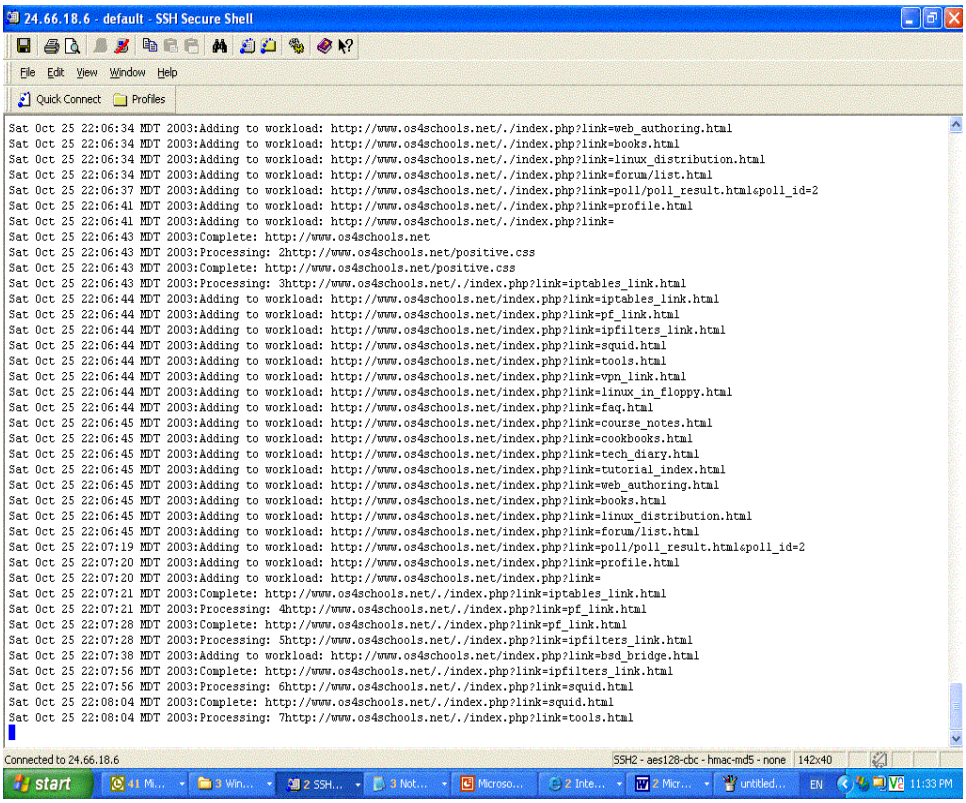
FieldName	DataType
CourseID	Text
InstructorID	Text

written in Java. We deployed the computers at different locations. The computers, Intel-based Pentium III class machines with 512MB RAM, are loaded with the following software:

- Red Hat Linux 8.0
- J2SE v 1.4.2
- Apache Web Server 2.047 w/Axis 1.1
- PHP 4.3.3
- MySQL 4.0.14

From preliminary experimental results, the approach proposed is feasible. The Web Services are provided by Apache Axis. We used JDBC to connect to

Figure 5. A screen shot of the Web monitoring agent



MySQL databases. Shown in Figure 5 is a screen shot of the Web monitoring agent.

We are doing experiments to test the scalability and usability of the system. The experiments will focus on the perceptions of the users regarding the helpfulness and overall usefulness of the agent system. Perceived satisfaction will be measured by a questionnaire asking about the students' perceptions of the quality improvement of course materials in using the agent. The questionnaire will also be sent to course instructors and administrators to allow us to compare the work efficiency, i.e., how many broken links the agent detected, how much the time between when the course materials were changed and the students were notified of the change was shortened by using the agent system, and how much of the course instructors' time was saved in maintaining course materials and notifying students and answering students' questions regarding course

material updates of the agent-supported content management system compared to nonagent course content management systems.

Agents for Learning Objects

The notion of "learning object" (LO) is a new way of thinking about learning content (Rory et al., 2002). Traditionally, course content comes in a several-hour chunk called a lesson. Learning objects are much smaller units of learning, ranging, for example from 2 to 30 minutes. Small, independent chunks of knowledge or interactions stored in a database can be presented as units of instruction or information. A learning object is based on a clear instructional strategy, intended to cause learning through internal processing and action.

Agents for learning objects include locating agents, monitoring agents, notifying agents, personal agents, and learning objects agents.

A locating agent (LA) is able to accept user interests as keywords and is able to offer approximate matches, if interests are expressed in terms of subject taxonomies. Taxonomies are {definition here}. These subject taxonomies can be large, especially when elaborated by cross-references. They form the basis for the agent's ability to make sense of user interests and their relative relationships to the subject matter of LOs. Our strategy is to provide a learning object repository (LOR) interface and agents that support students' learning through the search process. For instance, the LOR search interface will provide tools like spell-checking and content-specific thesauri to help sharpen query formulation.

The Monitoring Agent

A monitoring agent (MA) provides a time-saving way for LOR administrators to monitor the status of LORs. The MA checks content changes and detects broken links in LOs, saving administrators from the tedious and time-consuming task of doing this manually. The MA can do these automatically.

The Notifying Agent (NA)

Users specify events of interest and receive notifications by e-mail when these events occur. These events include identifying new items appearing in a

repository, new versions of LOs entering a repository, and some broken links. The NAs attach themselves to a LOR Broker Web service.

Both MAs and NAs accept messages in ACL (agent communication language) that specify events of interest and the actions they trigger. For example, one message might ask for e-mail notification whenever a repository adds a new LO, for example, a Java tutorial. Another message might define filters to extract articles matching current curricular items from a Web page. Students will also be able to use these kinds of agents to find relevant information in a timely manner.

Personal Agent

A personal agent (PA) runs on the user's machine. It manages the interface that connects human users to an LOR or network of LORs by

- Expressing user queries in a form that search agents can interpret
- Maintaining user profiles based on specified, default, and inferred user characteristics
- Customizing the presentation of query results

A PA is able to transfer the anonymous information transacted between itself and the broker to a specific user. And, the PA applies some more personal filtering, such as selecting between materials with similar topics but different human languages. It also ignores offers from the broker that the user has already seen. The actual learning delivery is primarily under the control of the learning management system (LMS). The PAs can assist instructors in developing and managing curriculum materials. For instance, the instructors' customized query-planning agent, with its specialized knowledge of pedagogical relevance, helps instructors quickly search and retrieve material useful in their courses.

Learning Object Agent (LOA)

An LOA is a representative of the author of a LO, able to *answer questions* that a learner would love to ask, for example:

- Which sections should I read first?
- Could you show me the most important sections?
- My particular interest is Java Network Computing, and I have half an hour. What should I read?
- I am lost—help me get my bearings.

The agent contains the reference engine, which applies the rules contained with the LOs. The agent must also provide a good user interface, possibly graphical or natural language driven, that can solicit requests from the learner and deliver responses. It is also responsible for dealing with the LOR agent.

Learning Object Repository Agent (LORA)

While an LOA works on behalf of learners using LOs, The LORAs work for the dynamic learning object repository. As learning objects are added, LORAs attend to linking the information, driven by the concept hierarchy, and then inform LOAs of the new state of the distributed LOR.

Because LORAs have access to the big picture, they can see how the LOR is being used.

LORA agents watch, listen, and learn how people are using the LOR. When the LORAs notice certain patterns recurring, they can help a LOA, and hence, a learner to find relevant information. This would be an application of a neural network, a program that learns from patterns. The LOAs may temporarily become users of other LOAs. When a learner poses a request for information that the LOA cannot satisfy, the agent may contact the LORA and ask for help. The LORA knows where to go for this help and can call up one or more different LOAs to supply the agent with the needed information. These LOAs may be running different inference mechanisms. They are certainly using different rules and LOs. We are using distributed Blackboard technology as a method to get these systems to work in a cooperative fashion.

Future Trends

Agent-Supported Web Services

Service-oriented computing is becoming the prominent paradigm for distributed computing and is creating opportunities for service providers and application developers to develop value-added services by combining Web services. Web services technology is currently being touted as the ideal solution to meet the previously mentioned requirements for the dynamic composition of Enterprise Information Systems (Yang & Papazoglou, 2000).

Agents have the potential to harmonize Web services' behaviors. The design of many software agents is based on the assumption that the user needs to specify a high-level goal instead of issue explicit instructions, leaving the how and when decisions to the agent. A software agent exhibits a number of features that make it different from other traditional components (Jennings et al., 1998), including autonomy, goal orientation, collaboration, flexibility, self-starting ability, temporal continuity, character, communication, adaptation, and mobility.

Software agents can play both roles of a Web service client as well as the role of a Web service. As a client of service, an agent can perform searches of different entries stored in a UDDI. It then can make message- and RPC-style calls to a Web service. As a Web service, an agent has a dual nature that combines the characteristics of the two technologies: the ability to be *published, found, and called* as a Web service and the ability to *make autonomous decisions*.

An agent-supported Web service can include a local information space and a set of agents supporting the Web service.

A service registry is the medium and services provider for service discovery.

The PA, upon receiving a task from a user, requests a task agent (TA), initiates a message, and posts it onto the nearest service registry. The message contains pertinent information, such as the job task, the requirements and criteria for the job, the expiry date, and ways of contacting the PA.

At the same time, the first respondent (a TA) interested in accepting the offer indicates that the case is closed to prevent other service providers from responding to the same message. Subsequently, the first responding TA then

keeps in touch with the concerned PA for finer negotiation. When all the qualified TAs are preoccupied, the message remains outstanding on the service registry until it attracts an interested TA that is free. For certain tasks that require a number of service providers bidding for a specific job, the interested TAs contact the concerned PA, which filters and selects the vendor based on certain criteria, such as the quoted price and the qualifications.

Web services for distributed learning include knowledge management and information-resource management. Knowledge-management Web services manage domain knowledge (ontologies, concepts, etc.) and knowledge about curriculum planning.

Information resource management Web services include learning object-repositories management, instructor-information management, tutor-information management, and student-information management. These information-resource management Web services are responsible for getting information about the resources needed.

To provide efficient and effective Web services, some agents can be deployed to support the Web services, taking advantage of the autonomy and distribution of agents. For example, a "spider-like" broken-link-checking agent can be used to maintain a LOR, supporting the LORs' management Web service.

Another example is the Ontology Web service. Ontology is a taxonomy database used for a target language for (a) the terms in the prerequisite and postconditions of learning objects, and (b) the terms in the learner profiles. An agent-supported ontology Web service can *maintain* the ontology knowledge autonomously when needed.

Knowledge Management

Ontology-Based Domain Modeling

There are two aspects to the obstacles of agent technology in distributed learning. One is the difficulty of understanding and interacting with data. The other is agent knowledge modeling. Here, knowledge modeling can be characterized as a set of techniques that focus on the specification of static and dynamic knowledge resources.

Modeling Curriculum Design Patterns

The emergence of design patterns for dealing with chaotic systems has applications in many fields. The real issue that needs to be addressed is when utilizing the framework to do so is appropriate. The course-development team and, in particular, the course designer should be able to utilize design patterns to assist in the modeling of the courses.

Design patterns can be used as a powerful tool in the creation of a curriculum's plans. When used together with the application of Learning Objects Oriented Course Design, design patterns are another tool that can be applied in order to achieve more robust, flexible, and adaptive curriculum plans, to organize LOs into cohesive yet independent course structures.

The ability of design patterns to do more than document the curriculum plan and course design decisions that were made in its creation offers a degree of protection for the adaptability built into the system. Design patterns also offer another way in which the curriculum plans can be conceived and created. The addition of this layer of abstraction to the Learning Object-Oriented paradigm clearly allows for a more robust and adaptable curriculum.

Conclusions

We have discussed an integrated approach to designing and developing adaptive distributed learning environments. The main objectives are to reduce the complexity of the development of distributed learning environments and to reduce the workloads of users (educators and learners) with personalized assistance in the environments where various resources are widely distributed, heterogonous, and ever-changing.

Web Services technology provides a new way to integrate existing systems or applications, and the ability to access data in a heterogeneous environment. In Web Services technology, rather than building large, closed systems, the focus is on flexible architectures that provide interoperability of components and learning content, and that rely on open standards for information exchange and component integration.

To reduce the information workload of educators and provide assistance to learners, distributed learning environments require that software not merely

respond to requests for information but intelligently adapt and actively seek ways to support learners and educators. To take advantages of these two technologies, we advocate integrating agents and Web services into distributed learning systems.

For future work, we will address the semantics integration and knowledge management when using the approach proposed in this chapter.

References

Baylor, A. (1999). Intelligent agents as cognitive tools for education. *Educational Technology, XXXIX*(2), 36–41.

Blackmon, W. H., & Rehak, D. R. (2003). Customized learning: A Web services approach, *Proceedings: Ed-Media 2003*.

Chan, T. –W. (1995). Artificial agents in distance learning. *International Journal of Educational Telecommunications, 1*(2/3), 263–282.

Dale, J., Ceccaroni, L., Zou, Y., & Agam, A. (2003). Implementing agent-based Web services, challenges in open agent systems. 2003 Workshop, July 15, *Autonomous Agents and Multi-Agent Systems Conference* in Melbourne, Australia, July 14–17.

Dellarocas, C. (2000). Contractual agent societies: Negotiated shared context and social control in open multi-agent systems. *Workshop on Norms and Institutions in Multi-Agent Systems, Fourth International Conference on Multi-Agent Systems (Agents-2000)*, Barcelona, Spain, June.

Gavrilova, T., Voinov, A. V., & Lescheva, I. (1999). Learner-model approach to multi-agent intelligent distance learning system for program testing. *IEA/AIE 1999* (pp. 440–449).

Geng, X., Gopal, R. D., Ramesh, R., & Whinston, A. B. (2003). Scaling Web services with capacity provision networks. *IEEE Computer*, November, 64–72.

Greer, J., McCalla, G., Cooke, J., Collins, J., Kumar, V., Bishop, A., & Vassileva, J. (1998). The intelligent helpdesk: Supporting peer-help in a university course. In B. P. Goettl, H. M. Halff, C. L. Redfield, & V. J. Shute (Eds.), *Intelligent tutoring systems. LNCS1452* (pp. 494–503). Heidelberg: Springer-Verlag

Greer, J., McCalla, G., Vassileva, J., Deters, R., Bull, S., & Kettel, L. (2001). Lessons learned in deploying a multi-agent learning support system: The I-Help experience. *Proceedings of AIED'2001* (pp. 410–421). San Antonio, TX.

Huhns, M. N. (2002). Agents as Web Services. *IEEE Internet Computing*, July/August, 93–95.

Jafari, A. (2002). Conceptualizing intelligent agents for teaching and learning. *Educause Quarterly*, *3*, 28–34.

Jennings, N., Sycara, K., & Wooldridge, M. (1998). A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems*, *1*(1), 7–38.

Jones, M., & Winne, P. H. (1992). *Adaptive learning environments*: *Foundations and frontiers*, NATO ASI Series F: Computer and Systems Sciences, Vol. 85.

Lin, F., Poon, L., & Leung, S. (2004). Integrating Web services and agent technology for e-learning course content maintenance. The *17th International Conference on Industry and Engineering Application of Artificial Intelligence and Expert Systems* (IEA/AIE), Special Session on IT for e-Learning, Ottawa, Canada, May 17–20.

Lin, F., & Holt, P. (2001). Towards agent-based online learning. *IASTED Int. Conf. Computer and Advanced Technology in Education (CATE)* (pp. 124–129). June 27–29, Banff, Canada.

Luck, M., McBurney, P., & Preist, C. (2003). *Agent technology: Enabling next generation computing: A roadmap for agent based computing*, AgentLink II.

O'Hare, G. M. P., & Jennings, N. R. (1996). *Foundations of distributed artificial intelligence*. New York: John Wiley & Sons.

Phillips, R. (1997). *The developer's handbook to interactive multimedia*. Kogan Page.

McGreal, R., Anderson, T., Friesen, N., Sosteric, M., Hewitt, K., Ring, J., Richards, G., Hatala, M., Calvert, T., Chiasson, M., Roberts, T., Carey, T., Harrigan, K., Paquette, G., & Downes, S. (2002). eduSource: A pan-Canadian learning object repository. In *Proceedings of the E-Learn 2002 Conference. Montreal: Association for the Advancement of Computing in Education*.

Sturm, A., Dovi, D., & Shehory, O. (2003). Single-model method for specifying multi-agent systems. *Proceedings of AAMAS, 2003, Melbourne* (pp. 121–128).

Thaiupathump, C., Bourne, J., & Campbell, J. O. (1999). Intelligent agents for online learning. *JALN, 3*(2), November, 1–19.

Vouk, M. A., Bitzer, D. L., & Klevans, R. L. (1999). Workflow and end-user quality of service issues in Web-based education. *IEEE Trans. on Knowledge and Data Engineering, 11*(4), 673–687.

Wagner, T., Garvey, A., & Lesser, V. (1998). Criteria-directed heuristic task scheduling. *International Journal of Approximate Reasoning,* Special Issue on Scheduling, *19*(12), 91–118.

Wang, H., & Holt, P. (2002). The design of an integrated course delivery system for Web-based distance education. In *Proceedings of the IASTED International Conference on Computers and Advanced Technology in Education (CATE 2002)* (pp. 122–126).

Wooldridge, M., & Jennings, N. (1995). Intelligent agents: Theory and practice. *The Knowledge Engineering Review, 10*(2), 115–152.

Yang, J., & Papazoglou, M. P. (2000). Interoperation support for electronic business. *Communication ACM, 43*(6), 39–47.

Yu, E. (2001). Agent-oriented modeling: Software versus the world. *Agent-Oriented Software Engineering AOSE-2001 Workshop Proceedings. LNCS 2222* (pp. 206–225). Heidelberg: Springer-Verlag.