An Intelligent Distributed Environment for Active Learning*

Yi Shang, Hongchi Shi, and Su-Shing Chen
Department of Computer Engineering and Computer Science
University of Missouri-Columbia
Columbia, MO 65211, USA

ABSTRACT

Active learning is an effective learning approach. In this paper, we present an intelligent agent assisted environment for active learning. The system is to better support student-centered, self-paced, and highly interactive learning approach. Students' learning-related profiles, such as learning styles and background knowledge, are used in selecting, organizing, and presenting learning materials. A new approach to course content organization and delivery is being developed based on smart instructional components, which can be integrated into a wide range of courses. The system is being implemented using the prevalent Internet, Web, digital library, and multi-agent technologies.

Keywords: active learning, Web-based education, multi-agent system, XML

1. INTRODUCTION

A major challenge in computer science education is to improve both instructional productivity and learning quality for a large and diverse population of students under real-world constraints such as limited financial resources and insufficient qualified instructors. The literature in education suggests that students who are actively engaged in the learning process will be more likely to achieve success [12]. The approach of active learning emphasizes on engaging students in the learning process [6, 22, 23], where learning activities involve some kind of experience or some kind of dialogue. The two main kinds of dialogue are *dialogue with self* (think reflectively) and *dialogue with others*. The two main kinds of experience are *observing* and *doing*. There are several ways to incorporate more active learning into teaching:

(1) Expand learning experiences. The most traditional teaching consists of little more than having students read a text and listen to a lecture, providing a very limited form of dialogue with others. Examples of more dynamic forms are creating small groups of students and having them make a decision or answer a focused question periodically, or finding ways for students to engage in authentic dialogue with people other than fellow classmates who know something about the subject (on the Web, by email, or live).

(2) Take advantage of the "power of interaction." There are four modes of learning when the two kinds of dialogue are coupled with the two kinds of experience. Each of the four modes has its own value, and just using more of them should add variety and thereby be more interesting for the learners. However, when properly connected, the various learning activities can have an impact that is more than additive or cumulative; they can be interactive and multiply the educational impact. For example, if students write their own thoughts on a topic (*dialogue with self*) before they engage in small group discussion (*dialogue with others*), the group discussion should be richer and more engaging. If they can do both of these and then observe the phenomena or action (*observation*), the observation should be richer and again more engaging. Then, if this is followed by having the students engage in the action itself (*doing*), they will have a better sense of what they need to do and what they need to learn during *doing*. Finally if, after *doing*, the learners process this experience by writing about it (*dialogue with self*) and/or discussing it with others (*dialogue with others*), this will add further insight. Such a sequence of learning activities will give the teacher and learners the advantage of the Power of Interaction. Alternatively, advocates of problem-based learning would suggest that a teacher start with *doing* by posing a real problem for students to work on, and then having students consult with each other (*dialogue with others*) on how best to proceed in order to find a solution to the problem. The learners will likely use a variety of learning options, including observing and dialogue with self.

(3) Create a dialectic between experience and dialogue. New experiences (whether of *doing* or *observing*) have the potential to give learners a new perspective on what is true (beliefs) and/or what is good (values) in the world. Dialogue (whether with self or with others) has the potential to help learners construct the many possible meanings of experience and the insights that come from them. People learn faster when new concepts are useful in their present as well as future lives. The roles of an educator is to assess the audience's interest, current skills, and aims. This information then guides the structuring of a learning atmosphere and selection of methods most satisfying and effective for the learners.

*Research supported in part by the National Science Foundation under grants DUE-9980375 and EIA-0086230 and the University of Missouri System Research Board.

WWW10, May 1-5, 2001, Hong Kong.
ACM 1-58113-348-0/01/0005.

The growing emphasis on student-centered active learning has yielded a veritable revolution in educational theory and practice. A number of current theories of learning and pedagogy revolve around constructivism, which emphasizes the student's knowledge construction process. In student-centered education, students should be engaged in active exploration, be intrinsically motivated, and develop an understanding of a domain through challenging and enjoyable problem-solving activities. Systems and procedures that support active learning have been intensively investigated and developed [7, 8, 9, 11, 18].

In this paper, we present an intelligent agent assisted system to support student-centered, self-paced, and highly interactive learning, a first step in building an effective active learning environment. The system provides a rich set of online contents and around the clock information access, maximizes the interactivity between the intelligent learning system and the students, and customizes the learning process to the needs of individual students. In the system, students' learning-related profiles, such as learning styles and background knowledge, are used for selecting, organizing, and presenting the learning materials to individual students and in supporting active learning. It supports personalized and more pleasant interaction between the users and the learning systems, enables adaptive delivery of IT education content, facilitates automatic evaluation of learning outcomes, and provides easy-to-use authoring tools. The system also incorporates a new approach to course content organization and delivery, which is developed based on smart instructional components, called *lecturelets*. Lecturelets are designed for customized interactive presentation of subjects. They are self-contained, autonomous, and can be easily integrated into a wide range of courses. The intelligent distributed environment for active learning (IDEAL) is implemented using the prevalent Internet, Web, digital library, and multi-agent technologies. IDEAL adopts an open system architecture supporting open standards in information technology and can scale to large-scale, distributed operations.

2. MULTI-AGENT SOFTWARE SYSTEM

IDEAL is a Web-based, distributed, multi-agent learning system with a three-tier architecture as shown in Figure 1. The system ties the Web clients (for students) and the underlying information servers (for courseware and student profiles) together with the multi-agent resource management. The information and agents are supported by a distributed system consisting of workstations and storage devices connected via high-bandwidth networks. IDEAL is implemented using the prevalent technologies of the Internet, WWW, software agents, and digital libraries [17, 31, 32].

Several characteristics specific to asynchronous learning make multi-agent systems attractive. First, the students of a virtual class on the Internet are widely distributed, and the number of potential participants is large. This renders static and centralized systems inadequate. A distributed multi-agent system with personalized agents for each student is very attractive. Secondly, the classes are dynamic in nature. The background, knowledge, and skill of active students will change over time. The learning materials and teaching methodologies of the courses will change too. Thirdly, students have different background and personality. Teaching methodology should be tailored toward each student's interest and knowledge to make teaching and learning more effective. Furthermore, students often enroll in several courses at the same time. Coordination of learning on different topics for each student will enrich the learning experience. Finally, students tend to get together to discuss study topics and share common interests. Smooth communications, including visualizing and sharing common contexts, need to be supported. Hence, multi-agent systems have become a promising paradigm in education [3, 31].

IDEAL consists of a number of specialized agents with different expertise. In IDEAL, each student is assigned a unique personal agent that manages the student's personal profile including knowledge background, learning styles, interests, courses enrolled in, etc. The personal agent talks to other agents in the system through various communication channels. An online course is supported by a collection of teaching and course agents. The course agents manage course materials and course-specific teaching techniques for a course. Multiple course agents exist on distributed sites to provide better efficiency, flexibility, and availability. The teaching agents can talk to any course agent of a course and often choose one nearby for better performance. The course agents also act as mediators for communication among students.

A teaching agent interacts with a student and serves as an intelligent tutor of a course. Each teaching agent obtains course materials and course-specific teaching techniques from a course agent and then tries to teach the materials in the most appropriate form and pace based on the background and learning style of the student. The teaching agents may adopt various cognitive skills such as natural language understanding, conversation, natural language generation, learning, and social aspects. These skills make it easier for students to interact with the teaching agents through natural forms of conversation and expression. Multimedia presentations such as graphics and animation make difficult concepts and operations easier to understand.

The basic components of a teaching agent are a domain expert module, a pedagogical module, and a student modeler. The domain expert module creates exercises and questions according to the student's background and learning status, provides solutions, and explains the concepts and solutions to remedy student's misconceptions. It contains a problem generator, a problem solver, an explanation generator, and a domain knowledge base. The pedagogical module determines the timing, style, and content of the teaching agent's interventions. It is a rule-based production system that uses the student model and pedagogical knowledge to determine the appropriate actions. The student modeler provides a model of a student based on her learning style, knowledge background, and interests. It may also incorporate the information gathered through dialogues with the student and the student's learning profile such as the actions the student performed and the explanations she asked for.

3. COMMUNITY INTERACTION

In a learning community, instructors and students work together systematically toward shared academic goals. Collaboration is stressed, and competition is deemphasized. The instructor's primary role shifts from delivering content to setting up learning environments and serving as coach, expert guide, and role model for learners. The student's role changes as well, from relatively passive observer of teach-

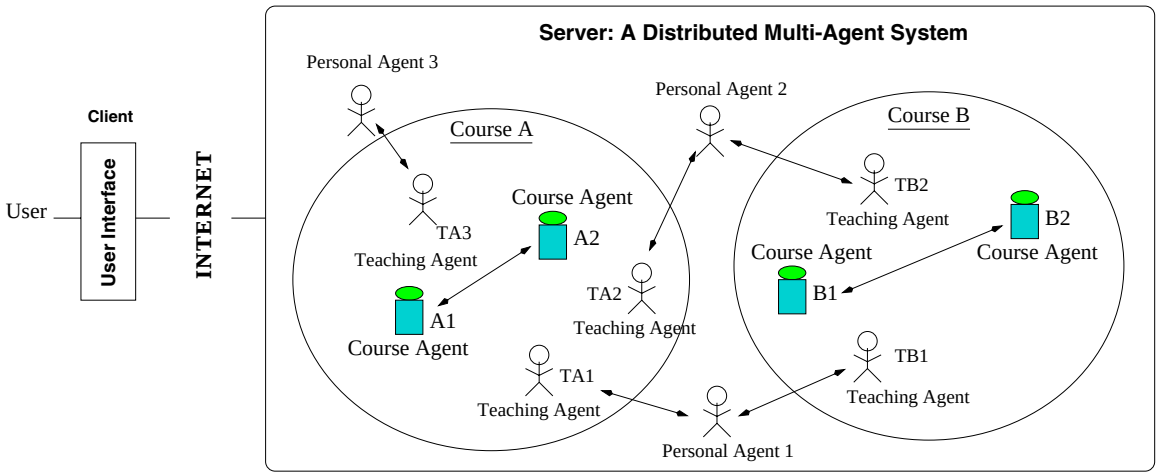

Figure 1: The framework of IDEAL, a Web-based interactive learning environment.

ing and consumer of information to active co-constructor of knowledge and understanding [2, 10, 28].

There are three major issues in the online support of a learning community: how to support various communication channels including one-to-one, one-to-many, and many-to-many, how to find other people that share similar interests, and how to visualize and share common contexts. Active support using agent technology based on interaction between software agents and between humans and software agents is necessary in addressing these issues [13].

The learning community enabled by IDEAL contains a collection of personal units and course agents. A personal unit consists of a user and his or her personal agents. Each personal agent can acquire the user's profile and help the user by gathering, exchanging, and viewing information. The course agents provide shared information, knowledge, or contexts within the course community and act as mediators for informal communication among people. They can collect the user profiles and maintain information on the course community.

4. STUDENT MODELING

Student modeling is crucial for an intelligent learning environment to be able to adapt to the needs and knowledge of individual students [5, 20]. There are many techniques for generating student models. Most of them are computationally complex and expensive, for example, the Bayesian networks [20, 21, 29], the Dempster-Shafer theory of evidence [4], and the fuzzy logic approach [14]. Other techniques although computationally cheap, such as the model tracing approach [1], can only record what a student knows but not the student's behavior and characteristics.

The difficulties in applying Bayesian modeling are the high cost in knowledge acquisition and in the time to update the student model. The inference in Bayesian belief networks is NP-hard, and the model requires prior probabilities. In developing practical and efficient Bayesian methods, we trade complexity of knowledge representation and depth of modeling for linear-time belief updating and a small number of model parameters.

In IDEAL, a student model is inferred from the performance data using a Bayesian belief network. The measure of how well a skill is learned is represented as a probability distribution over skill levels, such as novice, beginning, intermediate, advanced, and expert. Under the assumption that the performance on questions is independently distributed, to model one skill with n skill levels and q questions for each in a Bayesian network, we need nq probabilities plus the n prior probabilities of the skill levels to calculate the probability distribution of skill levels given all the question scores. To model k skills with the same skill levels for each, we need knq probabilities, which is too large for non-trivial real-world applications.

To reduce the number of probabilities required and improve the efficiency of the algorithm in IDEAL, questions of similar difficulty are grouped into categories associated with the conditional probabilities of answering each set of questions correctly to the possible skill levels. Now, only knc probabilities are required, where c is the number of categories.

The probabilities are further reduced by matching the question categories to the skill levels. For $n+1$ skill levels, only n question categories are required. If a student has reached a certain skill level, then he should be able to answer all questions at that skill level and all easier questions. Considering that students sometimes miss questions that they should know or may guess the right answer, the probability of a slip s, e.g., 0.1, and a probability of a lucky guess g, e.g., 0.2, are used in the conditional probabilities for correct answers to questions of increasing difficulty. By using these two probabilities, a simple way to set the conditional probabilities for 5 skill levels is as follows:

Question Categories	Skill Levels				
	Novice	Beginning	Intermediate	Advanced	Expert
Beginning	g	$1-s$	$1-s$	$1-s$	$1-s$
Intermediate	g	g	$1-s$	$1-s$	$1-s$
Advanced	g	g	g	$1-s$	$1-s$
Expert	g	g	g	g	$1-s$

Now, the total number of probabilities required is reduced to the prior probabilities for the skill levels plus the probabilities s and g.

Based on this model, the probability distribution of the skill levels given performance data can be determined in lin-

ear time. Based on the Bayesian theory and the assumption that the performance data are independent, the conditional probability of skill levels is as follows:

$$\begin{aligned} p(X = x_j|\vec{e}) &= \frac{1}{p(\vec{e})} * p(X = x_j) * p(\vec{e}|X = x_j) \\ &= \frac{1}{p(\vec{e})} * p(X = x_j) * \prod_{i=1}^{n} p(e_i|X = x_j) \\ &= \frac{1}{p(\vec{e})} * p(X = x_j) * \\ &\quad (1-s)^{\sum_{i=1}^{j} e_{i+}} * s^{\sum_{i=1}^{j} e_{i-}} * \\ &\quad g^{\sum_{i=j+1}^{n} e_{i+}} * (1-g)^{\sum_{i=j+1}^{n} e_{i-}} \end{aligned} \quad (1)$$

where X represents the skill levels; $\vec{e}$ is the evidence vector of n elements, in which each element e_i contains two numbers, e_{i+} and e_{i-}, corresponding to the number of correct and incorrect answers to questions at difficulty level i, respectively.

The advantages of this model are (1) questions can be added, dropped, or moved between categories with minimal overhead; (2) the model incorporates uncertainty and allows for both slips and guesses in student performance; (3) the time complexity is linear in the number of data items, whereas updating belief networks is in general NP-hard; and (4) only a small number of parameters are required. The restrictions of the model are that only binary-valued evidence is modeled and only one skill can be modeled at a time.

5. CURRICULUM SEQUENCING

Curriculum sequencing is one of the key components in an intelligent tutoring system [19, 27, 30]. In our approach, the topics are represented in a dependency graph, with links representing the relationship between topics, which include prerequisite, co-requisite, related, and remedial. Remedial topics such as special topics are not required to be learned by all the students. Each topic is divided into subtopics corresponding to smaller grained units that allow the intelligent tutor to reason at a finer level. When a subtopic is displayed to the student, the actual content is dynamically generated based on the student model.

Curriculum sequencing can be seen as a two-step process: finding relevant topics and selecting the best one. A student is ready to learn a topic only if he has performed sufficiently well on its prerequisites. How well a topic is learned is judged by the student's performance on and his access patterns to the course materials. The access patterns include how much time he has spent studying a topic, whether he used corresponding multimedia materials such as audio and video, and if the topics were reviewed multiple times. Specifically, the performance on a topic is determined based on the following three factors:

1. Quiz performance. Quizzes give a tutor the most direct information about the student's knowledge. Quizzes can be dynamically constructed based on the student model. Questions are provided to cover the topics most recently completed, as well as topics that should be reviewed. Each question has a level of difficulty, which is also used in updating the student model. Correctly answering a harder question demonstrates a higher ability than correctly answering an easier question. The quiz scores are calculated using the following formula:

$$Score^{k+1} = \begin{cases} \alpha * Score^{k} + (1-\alpha) * \frac{d}{L} & \text{if answered correctly} \\ \max(0, \alpha * Score^{k} - (1-\alpha) * \frac{L+1-d}{L}) & \text{if answered incorrectly} \end{cases} \quad (2)$$

where $0 < \alpha < 1$ is a constant corresponding to the updating rate, k is the index of the updating iteration, $1 \leq d \leq L$ is the level of difficulty, and L is the total number of levels. The $Score$ is bounded between 0 and 1. Each topic has an initial score of 0 or some heuristic value derived from prior knowledge.

2. Study performance. The main interaction that students have with the learning environment is through viewing or listening to the course materials in multimedia forms. The study score is used to judge how much comprehension the student has gained through these activities. A topic is usually presented in multiple pages and each page is assigned a weight corresponding to its importance. Then, the study score in the range from 0 to 1 is calculated based on the pages visited and the amount of time spent on each page. An optimal time for each page is used as the baseline. If the student spends this optimal amount, then the score for the page is 1. As he moves away from this point, his score decreases. The study score of a student on a topic is the weighted sum of the scores on the pages calculated as follows:

$$Study\_Score = \sum_{i=1}^{N} w_i S_i \quad (3)$$

where N is the number of pages, w_i is the weight on the ith pages with $\sum_{i=1}^{N} w_i = 1$, and S_i is the score on the ith page. If page i is studied more than once, the total time on the page is used for S_i.

3. Reviewed topics. The review score on a topic records how much the student has returned to review the topic again. It is based on how many times the topic is reviewed and how much of the materials is viewed each time. If a student is reviewing frequently, then he has not learned the material. The review score is in the range from 0 to 1 and starts at 1 for each topic. Each time the student reviews the topic, the review score is updated by multiplying the value calculated using Equation (3).

These three scores, quiz performance, study performance, and reviewed topics, are then combined into a single value, *learned score*, indicating how well the topic is learned. The quiz score is the most important among the three. When a student has a reasonably high quiz score, such as over 0.8, then the other scores do not matter much and the final value is the quiz score. However, if his quiz score is less than 0.8, the other factors become important, and the final value is a weighted sum of the three scores with weights such as 0.7, 0.2, and 0.1, respectively.

The *ready score* of a topic indicates whether the student is ready to learn the topic or not. It is calculated based on the topic's learned score and its pretopics' learned scores. If a given topic's learned score is too low, it should be presented again, perhaps being taught differently than it was the first

time. In order to start a new topic, a student should show sufficient scores in its pretopics. One formula of the ready score is the weighted-sum of the topic's learned score and its pretopics' learned scores with predetermined weights.

In IDEAL, the student has the option of letting the teaching agent choose the next topic or choosing it himself. In both cases, the student must achieve sufficiently high ready score for the topic. If the teaching agent is asked to choose the next topic, it will choose one with the highest ready score. If the student decides to choose the next topic, he is presented the topic dependency graph annotated with suggestions on which topics to repeat and/or which new topics to study.

Once a topic is chosen, how to teach, such as how to dynamically construct page contents, can be made based on the three individual topic scores. For example, a student who has poor quiz scores on a topic and who has not studied the topic for very long should be treated differently from a student with the same quiz score but who spent much more time studying. The second student should be presented with more background materials to improve his comprehension. Furthermore, each time a student reviews a topic, it should be taught in a different way than it was the last time.

6. COURSE MATERIAL ORGANIZATION AND DELIVERY

In IDEAL, it is essential to have an effective electronic means for managing, delivering, processing, and presenting educational materials. Achieving these goals requires an approach that is not only extensible into the future but also adaptable to incorporate new technologies and requirements. To ensure broad adoption, the technology selected needs to be widely and freely available as an open standard. By selecting a paradigm that by its very nature is dynamically defined, extensible, and simple, these goals can be intrinsically met.

We develop an innovative approach based on active XML (eXtensible Markup Language) documents for organizing and delivering course materials. Courses materials are decomposed into small components, corresponding to lecturelets, around the subjects to be learned. *Lecturelets* are "smart" XML documents, namely XML documents that carry not only contents but also Java code. Lecturelets can be dynamically assembled to cover course topics according to individual student's progress. By using standards for accessing XML documents with style information, lecturelets can be cataloged, searched, exchanged, and viewed. In contrast to most of the existing learning materials that are static, our approach provides an exciting dynamic process that can be infinitely extended. It has the potential to change the way universities manage and transfer their educational materials.

Our approach is enabled by the four complementary and powerful technologies: XML, template, agent, and repository. Each component adds unique tools that leverage the other pieces.

(a) XML provides the foundation. XML brings with it all the rich capabilities and transport layers of the Web and the Internet in general. The logical structure of an XML document can be specified in a Document Type Definition or DTD. Representing the course materials as structured XML documents makes searching, archiving, reading, and navigating the documents simpler.

(b) Templates are the rules providing the glue that holds the whole dynamic interactive learning process together. Templates are referenced or travel along inside the XML as a special section, and can be easily read and interpreted. They are supplemented by DTDs. DTDs enable task interoperability, while templates enable processing, including presentation, of tasks. DTDs let two participants understand each other's XML documents, while templates define what happens to the documents. The leading browsers supporting XML allow for the lecturelets to be viewed exactly the way the user wants it.

(c) Agents interpret the templates to perform the task needed and may interact with the user to create a new template for each new specific task, or look up and attach the right template for an existing job. They also can reference DTD's to determine display characteristics for documents. This is where Java and ActiveX fit in. Lecturelet agents on the Web browser can obtain updated information and instructions from the server agent and can also provide feedback to the server agent for statistical analysis and data mining. The benefit of using intelligent agents is to make the system much easier to use, more intelligent, and more fault tolerant. In many cases agents will resolve problems without the user being aware there was a problem.

(d) The repositories provide the storage for lecturelets, student models, and other components involved in the learning process. New lecturelets can be dynamically added to the repositories with little interruption to the ongoing learning activities. The repositories also allow for indexing, automatic lookup, and sharing of lecturelets among different learning systems.

Lecturelets contain both the XML documents and the instructions (templates/agents) on how the documents should be processed or displayed. The internal elements of lecturelets and the framework for intelligent delivery of lecturelets on the Internet are shown in Figure 2. The lecturelet framework makes the transfer of educational materials between different software systems transparent to the user and as easy as possible. It allows software agents to reach out to the Internet to read from and "make sense" of online course listings. In XML, each document is an object and each element of the document is an object, too. Being XML documents containing both data and code, lecturelets can be manipulated as objects.

Once defined, templates can be applied to the objects in XML documents. Based on user defined templates, lecturelets will be re-organized during the learning process and displayed accordingly, and may even trigger events on their own. For example, they will be able to find an application by using the searching, classifying, and routing mechanisms. They will have learning status self-contained for users to set and interrogate. Lecturelets can either run independently, or interact with each other through standard XML messages.

IDEAL is able to use the many search tools that are being adapted for XML. The lecturelet framework will allow for the search of educational materials in various ways. In addition to the keyword-based search, the objects in XML documents allow for more intelligent searches such as content-based search. There are already SGML query languages that are similar to SQL in power. With standardized DTDs for different applications one could retrieve information accurately. The relationships in the document structures can be used as well as the objects themselves in the query. The

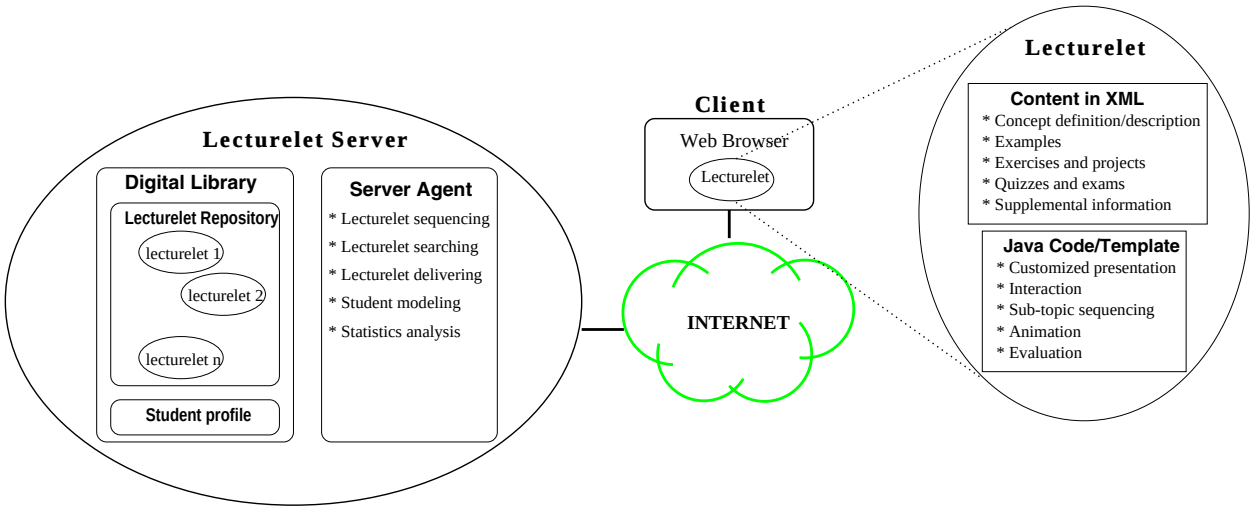

Figure 2: ***The framework for intelligent delivery of lecturelets and the internal elements of lecturelets.***

DTD allows for precise relational searches of the XML documents either in the local repositories or on the Web.

IDEAL provides an interactive learning environment that combines the visual presentation of course information, class notes, and executable components of learning materials. In learning a subject, the lecturelet agent is essentially a teaching agent. The lecturelet agent obtains the student model from the lecturelet server and updates the model by observing the student's learning process. It teaches the materials in the most appropriate form and pace based on the background and learning capability of the student.

The lecturelet agent contains a student model, a simple subject-unit sequencing module and an assessment component. The student model represents the student's learning style and knowledge levels. The subject-unit sequencing module is responsible for selecting the most appropriate basic subject units to be presented to the student based on the performance of the student on previous subject units, the student models, and the dependency relationships between the subject units. A lecturelet on a subject usually contains a collection of basic subject units. The assessment component records the usage of the lecturelet, the performance of students on the exercises and quizzes, and the comments from the students. The assessment data will be uploaded to the server when the current session is finished. The student model and profile will then be revised on the lecturelet server by the server agent accordingly. The lecturelet agent may also adopt various cognitive skills such as natural language understanding, conversation, natural language generation, learning, and social aspects. These make it easier for students to interact with the agent through natural forms of conversation and expression.

A course is composed of a collection of lecturelets. The server agent on the lecturelet server manages lecturelets and their relationships. It configures the lecturelets into a coherent course sequence based on the course objective and the target audience. It uses the pedagogical modeling technique to deliver the appropriate subsequent lecturelets to individual students based on their performance and interests. It collects student performance data from lecturelet agents and performs data analysis and data mining to extract useful information for improving teaching in the future. The agent on the lecturelet server also manages the student profiles and is responsible for updating student models.

7. IMPLEMENTATION DETAILS OF THE PROTOTYPE

A prototype of IDEAL is implemented using Java so that the system can run on heterogeneous platforms. It consists of interface agents, personal agents, teaching agents, and course agents. The software agents communicate with each other in XML messages through a variety of communication channels, including peer-to-peer, multicasting, and broadcasting. Real-time communication among users has been implemented in the forms of chat room, white board, and streaming audio and video. The hardware environment consists of high-performance workstations and storage devices connected via high-bandwidth, low-latency networks. The agents run concurrently on the servers and workstations in the distributed environment with operating system level support for software agents. The courseware and student profile database are stored in RAIDs and accessed through the storage area network (SAN) consisting of RAIDs and servers connected via a Fibre Channel switch. The multimedia courseware is in the form of XML documents and organized as a flexible, extensible, and scalable digital library. The Web-based interface acts as a bridge between the student and IDEAL. This interface interacts with the HTTP Web server for adaptive delivery of electronic courses over the WWW, assignment submissions, and student learning assessment. The Web-based interface with a rigorous authentication process is implemented using new technologies, including Java Servlets and JSP (JavaServer Pages).

We have experimented with several distributed object-oriented environments in developing multi-agent systems [24, 25, 26], including Java Remote Method Invocation (RMI), JATLite [16], and JavaSpace [15]. Among these Java technologies, Java RMI provides an intermediate network layer that allows Java objects residing at distributed sites to communicate using normal method calls. It is reliable, has good performance, and works on many platforms. JATLite allows users to quickly create software agents that communicate robustly over the Internet and supports mobile agents. It

provides a variety of agent functionalities including registering with an Agent Message Router (AMR) using a name and password, connecting/disconnecting from the Internet, sending/receiving message asynchronously, and transferring files with FTP. A problem with the current implementation is that the software is not very stable and sometimes hangs completely. JavaSpace supports robust distributed communication and data interchange and provides a simple, expressive, and powerful tool that eases the traditional burden of building distributed applications. JavaSpace is very new, has not been fully developed, and is slow.

A prototype of the Web-based lecturelet management and delivery system together with a sample set of lecturelets on Web and agent technologies is also developed in the CECS department at MU. The implementation consists of the server side and the client side connected through the Internet. The server side consists of the intelligent agents, the student profiles, and the lecturelets that can be sent dynamically to the client and can be dynamically updated. The server agent is responsible for delivering these lecturelets to the client and handling all kinds of requests from the client. The lecturelet agent is responsible for teaching the lecturelets. Prototypes of the agents are implemented on top of distributed object-oriented software environments including Java RMI, JavaSpace, and JATLite.

The client side consists of a browser that has support for XML and Java applets. Applets are used for dynamic processing on the client side thus reducing the load on the server as well as on the network. The contents in XML are presented either using XSL (eXtended Stylesheet Language) or Java applets depending on the level of processing that needs to be done on the client side. As a simple example of client-side processing, we take a look at the quiz applet. The quiz applet allows the user to take the quiz at the client side. The "quiz.xml" file looks like:

```
<quiz>
  <query>
  <subject>Which of the following is a programming
           language?
  </subject>
    <choices>
      <item>English</item>
      <item>Emacs</item>
      <item choice="correct">Java</item>
    </choices>
  </query>
  <query>
    <subject>Who invented the Internet?</subject>
    <choices>
      <item>Al Gore</item>
      <item>George Bush</item>
      <item choice="correct">Hard to identify
      </item>
    </choices>
  </query>
</quiz>
```

The document consists of the content structured by a set of tags. The questions and the answer choices are the content, and the tags associated with this content are used to present this document. The attribute "correct" plays a vital role in dynamically validating the student's choices against the correct choice. The XML document is parsed to obtain an object model for presentation in an applet. The nodes in the object model are extracted using a number of methods of the DOM API. Once the student finishes the quiz, the student's choices are dynamically validated against the correct solution on the client side.

The presentation of the lecturelets is quite simple. The XML file and its corresponding XSL file are placed on the Web server. According to the student's request, the Web server responds appropriately. The response is generally a method invocation on the servlet object. This method of handling requests is far more efficient than CGI.

There are a number of aspects that are essential to build a reliable learning system. They include security on the client side to prevent the student from printing off the test/quiz and others, effective lecturelet search techniques, effective student modeling to model student performance, and a dynamic question set for each quiz/exam every time it is retaken by the student. In addition, we need to do more research to further increase the intelligence of the learning system.

8. REFERENCES

[1] J. R. Anderson, A. T. Corbett, K. Koedinger, and R. Pelletier. Cognitive tutors: lessons learning. *The Journal of the Learning Sciences*, 4(2):167–207, 1995.

[2] T. A. Angelo. The campus as learning community: Seven promising shifts and seven powerful levers. *AAHE Bulletin*, 49(9):3–6, 1997.

[3] L. Barnett, J. Kent, J. Casp, and D. Green. Design and implementation of an interactive tutorial framework. *SIGCSE Bulletin*, 30(1):87–91, March 1998.

[4] M. Bauer. A Dempster-Shafer approach to modeling agent references for plan recognition. *User Modeling and User-Adapted Interaction*, 5:317–348, 1996.

[5] J. E. Beck and B. P. Woolf. Using a learning agent with a student model. In B. P. Goettl, H. M. Halff, C. L. Redfield, and V. J. Shute, editors, *Intelligence Tutoring System (Proc. 4th Int'l Conf. ITS'98)*, pages 6–15. Springer, 1998.

[6] C. Bonwell. Building a supportive climate for active learning. *The National Teaching and Learning Forum*, 6(1):4–7, 1996.

[7] C. Buron, M. Grinder, and R. Ross. Tying it all together: Creating self-contained, animated, interactive, web-based resources for computer science education. *SIGCSE Bulletin*, 31(1):7–11, March 1999.

[8] C. A. Carver, R. A. Howard, and W. D. Lane. Enhancing student learning through hypermedia courseware and incorporation of student learning styles. *IEEE Trans. on Education*, 42(1):33–38, February 1999.

[9] C. Chou. Developing hypertext-based learning courseware for computer networks: The macro and micro stages. *IEEE Trans. on Education*, 42(1):39–44, February 1999.

[10] K. P. Cross. Why learning communities? why now? *About Campus*, 3(3):4–11, 1998.

[11] A. Davidovic and E. Trichina. Open learning environment and instruction system (OLEIS). *SIGCSE Bulletin*, 30(3):69–72, September 1998.

[12] V. F. Hartman. Teaching and learning style

preferences: Transitions through technology. *VCCA Journal*, 9(2):18–20, 1995.

[13] F. Hattori, T. Ohguro, M. Yokoo, S. Matsubara, and S. Yoshida. Socialware: Multiagent systems for supporting network communities. *Communications of the ACM*, 42(3):55–61, March 1999.

[14] L. W. Hawkes, S. J. Derry, and E. A. Rundensteiner. Individualized tutoring using an intelligent fuzzy temporal relational database. *Int'l Journal of Man-Machine Studies*, 33:409–429, 1990.

[15] E. Freeman S. Hupfer and K. Arnold. *JavaSpaces Principles, Patterns, and Practice*. Addison-Wesley, 1999.

[16] JATLite. http://java.stanford.edu/java_agent/html.

[17] N. R. Jennings and M. J. Wooldridge. *Agent technology: Foundations, applications, and Markets*. Springer, Berlin, 1998.

[18] H.A. Latchman, C. Salzmann, D. Giblet, and H. Bouzekri. Information technology enhanced learning in distance and conventional education. *IEEE Trans. on Education*, 42(4):247–254, November 1999.

[19] D. McArthur, C. Stasz, J. Hotta, O. Peter, and C. Burdorf. Skill-oriented task sequencing in an intelligent tutor for basic algebra. *Instructional Science*, 17(4):281–307, 1988.

[20] W. R. Murray. A practical approach to Bayesian student modeling. In B. P. Goettl, H. M. Halff, C. L. Redfield, and V. J. Shute, editors, *Intelligence Tutoring System (Proc. 4th Int'l Conf. ITS'98)*, pages 424–433. Springer, 1998.

[21] V. A. Petrushin and K. M. Sinista. Using probabilistic reasoning techniques for learner modeling. In *World Conf. on AI in Education*, pages 418–425, Edinburgh, 1993.

[22] L. G. Richards. Promoting active learning with cases and instructional modules. *Journal of Engineering Education*, 84(4):375–381, 1995.

[23] L. Rubin and C. Hebert. Model for active learning: Collaborative peer teaching. *College Teaching*, 46(1):26–30, 1998.

[24] Y. Shang, C. Sapp, and H. Shi. An intelligent web representative. *Information*, 3(2):253-262, 2000.

[25] Y. Shang and H. Shi. A web-based multi-agent system for interpreting medical images. *World Wide Web*, 2(4):209–218, 1999.

[26] H. Shi, Y. Shang, A. Joshi, and M. Jurczyk. Laboratory-oriented teaching in web and distributed computing. In *Proc. 2000 ASEE Annual Conference & Exposition*, St. Louis, June 2000.

[27] M. K. Stern and B. P. Woolf. Curriculum sequencing in a Web-based tutor. In B. P. Goettl, H. M. Halff, C. L. Redfield, and V. J. Shute, editors, *Intelligence Tutoring System (Proc. 4th Int'l Conf. ITS'98)*, pages 584–593. Springer, 1998.

[28] V. Tinto. Universities as learning organizations. *About Campus*, 1(6):2–4, 1997.

[29] M. Villano. Probabilistic students models: Bayesian belief networks and knowledge space theory. In *Intelligence Tutoring System (Proc. 2nd Int'l Conf. ITS'92)*, pages 491–498. Springer, 1992.

[30] G. Weber. Individual selection of examples in an intelligent learning environment. *Journal of Artificial Intelligence in Education*, 7(1):3–31, 1996.

[31] G. Weiss, editor. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. The MIT Press, Cambridge, MA, 1999.

[32] xml.com. XML.COM online, 2001. http://www.xml.com.